Measures on Boolean polynomials and their applications in data mining

Szymon Jaroszewicz\textsuperscript{a}, Dan A. Simovici\textsuperscript{a,\ast}, Ivo Rosenberg\textsuperscript{b}

\textsuperscript{a}Department of Computer Science, University of Massachusetts at Boston, Boston, MA 02125, USA
\textsuperscript{b}Université de Montréal, C.P. 6128, succ. A, Montréal, P.Q. H3C 3J7, Canada

Received 30 April 2002; received in revised form 25 February 2004; accepted 1 June 2004

Abstract

We characterize measures on free Boolean algebras and we examine the relationships that exist between measures and binary tables in relational databases. It is shown that these measures are completely defined by their values on positive conjunctions, and a formula that yields this value is obtained using the method of indicators. An extension of the notion of support that is well suited for tables with missing values is presented. Finally, we obtain Bonferroni-type inequalities that allow for approximative evaluations of these measures for several types of queries. An approximation algorithm and an analysis of the results produced is also included.

Keywords: Free Boolean algebra; Measure; Bonferroni-type inequality; Inclusion–exclusion; Missing values; Frequent itemset

1. Introduction

The focus of this paper is a study of measures on free Boolean algebras with a finite number of generators (abbreviated as MFBAs). As we shall see, these measures play an important role in query optimization in relational databases, and also, in the study of frequent sets in data mining. We obtain general Bonferroni-type inequalities for sizes of arbitrary Boolean queries.

The origin of our investigation resides in a series of seminal papers by Mannila et al. \cite{12,13,15} in which the idea of using supports of attribute sets discovered with a data mining algorithm to obtain the size of a database query was introduced.

We start from the premise that Boolean algebra is the natural framework for estimating the size of queries applied to relational databases. Indeed, the set of conditions that specify a relational algebra selection (or a \texttt{where} clause in a \texttt{select} SQL phrase) is build from atomic conditions of the form $A \{a\} \{\leq\} B$, where $A,B$ are attributes, $a$ is a value of the domain of the attribute $A$, and $\{\leq\}$ is one of the relational operators $<, \leq, \geq, =, \neq$. Then, more complex conditions are built using the connective symbols \texttt{and}, \texttt{or}, \texttt{not}. As we shall see, this is precisely the free Boolean algebra over the set of atomic conditions. Thus, the size of the answer to a particular selection $p$ can be regarded as a measure over the free Boolean algebra generated by atomic conditions and the value of this measure on $p$ can be used in query processing in relational databases (see \cite{19}, for example).
After introducing basic definitions and notations concerning Boolean algebras and relational databases in Section 2, we give a representation result for measures on free Boolean algebras that show that any such functions can be induced by a table.

Section 3 presents an inclusion–exclusion principle for MFBAs using the method of indicator variables. The same section presents Bonferroni-type inequalities [5] that allow us to generate bounds on the value of measures of polynomials. These results are significant for estimations of the size of queries in relational databases, or for support estimations for item sets in data mining. Such issues are explored in Section 4. Also, we extend the notion of support for queries to tables with missing values. Our approach has the advantage of generating values that are probabilistically consistent, unlike the techniques used in [14, 17].

We conclude the paper by presenting an algorithm that computes bounds on support of an itemset based on a collection of itemsets with known supports.

2. Measures on the set of polynomials and tables

Let \( \mathcal{B} = (B, 0, 1, \neg, \vee, \wedge) \) be a Boolean algebra, where 0, 1 \( \in B \) are two distinguished elements of \( \mathcal{B} \), \( \neg \) is a unary operation, and \( \vee, \wedge \) are two binary associative, commutative, and idempotent operations that satisfy the usual axioms of Boolean algebras (see, for example [18]). Here 0 and 1 are the least and the largest element of the algebra, respectively.

We define \( x^b = x \) if \( b = 1 \) and \( x^b = \bar{x} \) if \( b = 0 \), for \( x \in B \) and \( b \in \{0, 1\} \).

It is a well-known fact that a Boolean algebra \( \mathcal{B} = (B, 0, 1, \neg, \vee, \wedge) \) defines a Boolean ring structure, \( \mathcal{B} = (B, 0, 1, \wedge, \oplus) \), where \( \wedge \) plays the role of the multiplication, and \( \oplus \) the role of addition, where

\[
x \oplus y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)
\]

for \( x, y \in B \). This ring is unitary, commutative, and has characteristic 2 (since \( x \oplus x = 0 \) for every \( x \)). Also, \( 1 \oplus x = \bar{x} \).

Let \( A = \{a_1, \ldots, a_n\} \) be a set of \( n \) variables. The set \( \text{pol}(A) \) of Boolean polynomials of the \( n \) variables in \( A \) is defined inductively by:

1. \( 0, 1, \) and each \( a_i \) belong to \( \text{pol}(A) \) for \( 1 \leq i \leq n \);
2. if \( p, q \) belong to \( \text{pol}(A) \), then \( \bar{p}, (p \vee q) \), and \( (p \wedge q) \) belong to \( \text{pol}(A) \).

If \( p, q \in \text{pol}(A) \), then we denote by \( (p \oplus q) \) the polynomial \( ((p \wedge \bar{q}) \vee (\bar{p} \wedge q)) \). A Boolean polynomial \( \cdots ((p_1 \wedge p_2) \wedge p_3) \wedge \cdots \wedge p_n \) is denoted by \( (p_1 \wedge p_2 \wedge \cdots p_n) \), where \( \omega \in \{\vee, \wedge, \oplus\} \).

Let \( \mathcal{B} = (B, 0, 1, \neg, \vee, \wedge) \) be a Boolean algebra and let \( A = \{a_1, \ldots, a_n\} \) be a set of \( n \) variables. The \( n \)-ary function \( f_p : B^n \rightarrow B \) generated by a polynomial \( p \in \text{pol}(A) \) is defined in the usual way. We write \( p = q \) for \( p, q \in \text{pol}(A) \) if \( f_p = f_q \).

Let \( \bar{b} = (b_1, \ldots, b_n) \) be a sequence of elements of the set \( \{0, 1\} \). An \( A \)-minterm is a Boolean polynomial

\[
p_{\bar{b}}^A = a_1^{b_1} \wedge \cdots \wedge a_n^{b_n}.
\]

The set of \( A \)-minterms is denoted by \( \text{mint}(A) \). Any Boolean polynomial in \( \text{pol}(A) \) can be uniquely written as a disjunction of some subset of \( A \)-minterms (up to the order of the disjuncts). This observation implies that the Boolean algebra \( \text{pol}(A) \) is isomorphic to the Boolean algebra of collections of subsets of the set \( A \); thus, \( \text{pol}(A) \) has \( 2^{|A|} \) elements.

For a set of polynomials \( M = \{p_1, \ldots, p_n\} \) and \( J = \{j_1, \ldots, j_m\} \subseteq \{1, \ldots, n\} \) we denote by \( p_J \) the conjunction \( p_{j_1} \wedge \cdots \wedge p_{j_m} \). For the special case, when \( J = \emptyset \) we write \( p_J = 1 \).

A measure on a Boolean algebra \( \mathcal{B} = (B, 0, 1, \neg, \vee, \wedge) \) is a non-negative, real-valued function \( \mu : B \rightarrow \mathbb{R} \) such that \( \mu(x \vee y) = \mu(x) + \mu(y) \) for every \( x, y \in B \) such that \( x \wedge y = 0 \).

Let \( A = \{a_1, \ldots, a_n\} \) be a set of variables. In this context, we find it convenient to use the relational database terminology and we refer to the members of \( A \) as attributes. We attach a set \( \text{Dom}(a_i) \) to each attribute \( a_i \) such that \( |\text{Dom}(a_i)| \geq 2 \). The set \( \text{Dom}(a_i) \) is the domain of \( a_i \).

A table is a triple \( \tau = (T, A, \rho) \), where \( T \) is the name of the table, \( A = \{a_1, \ldots, a_n\} \) is the heading of the table and \( \rho = \{t_1, \ldots, t_m\} \) is a finite set of functions of the form \( t_i : A \rightarrow \bigcup_{a \in A} \text{Dom}(a) \) such that \( t_i(a) \in \text{Dom}(a) \) for every \( a \in A \). Following the relational database terminology we shall refer to these functions as \( A \)-tuples, or simply as tuples. If \( \text{Dom}(a_i) = \{0, 1\} \) for \( 1 \leq i \leq n \), then \( \tau \) is a binary table.

Let \( \tau = (T, A, \rho) \) be a binary table. A query on the table \( \tau \) is a Boolean polynomial in \( \text{pol}(A) \). This definition of queries is a formalization of the usual notion of queries in databases.
Theorem 2.2. A function $p$ is defined inductively as follows:

1. $\rho_0 = 0$ and $\rho_1 = \rho$;
2. if $p = a_i$, then $\rho_p = \{t \in \rho|t(a_i) = 1\}$;
3. if $p = \bar{q}$, then $\rho_p = \rho - \rho_q$;
4. if $p = (q_1 \lor q_2)$, then $\rho_p = \rho_{p_1} \cup \rho_{p_2}$ and,
5. if $p = (q_1 \land q_2)$, then $\rho_p = \rho_{p_1} \cap \rho_{p_2}$.

It is easy to see that for a conjunction

$$p = a_{i_1}^{b_{i_1}} \land \ldots \land a_{i_m}^{b_{i_m}},$$

where $b_i \in \{0, 1\}$ for $1 \leq i \leq m$, the set $\rho_p$ consists of those tuples $t$ such that $t(a_{i_\ell}) = b_\ell$ for $1 \leq \ell \leq m$.

Theorem 2.2. A function $\mu : \text{pol}(A) \rightarrow \mathbb{N}$ is a measure if and only if there exists a binary table $\tau = (T, A, \rho)$ such that $\mu(p) = |\rho_p|$ for all $p \in \text{pol}(A)$.

Proof. Suppose that $\tau = (T, A, \rho)$ is a table. Define the mapping $\mu_\tau : \text{pol}(A) \rightarrow \mathbb{R}$ by $\mu(p) = |\rho_p|$ for every $p \in \text{pol}(A)$. Let $p, q$ be two polynomials such that $(p \land q) = 0$. Then, $\mu_\tau(p \lor q) = |\rho_p \lor q| = |\rho_p \lor \rho_q|$. Since $p \land q = 0$ we have $\rho_p \lor \rho_q = 0$, so $\mu_\tau(p \lor q) = \mu_\tau(p) + \mu_\tau(q)$. Thus, $\mu_\tau$ is a measure on $\text{pol}(A)$.

Conversely, let $\mu$ be a measure on $\text{pol}(A)$, where $A = \{a_1, \ldots, a_n\}$. If $\vec{b} = (b_1, \ldots, b_n) \in \{0, 1\}^n$, $p^\vec{b} = a_1^{b_1} \land \ldots \land a_n^{b_n}$ is a minterm and $\mu(p^\vec{b}) = k$ consider a set $\sigma_{p^\vec{b}}$ of $k$ tuples $t^\vec{b}_1, \ldots, t^\vec{b}_k$, where $t^\vec{b}_j(a_i) = b_i$ for every $i, j$, $1 \leq j \leq k$, and $1 \leq i \leq n$. Define the table $\tau_{\vec{b}} = (T, A, \rho_\vec{b})$, where $\rho_\vec{b} = \bigcup_j \sigma_{p^\vec{b}_j} \in \text{mint}(A)$.

We claim that $\mu(p) = |\rho_p|$ for every polynomial $p \in \text{pol}(A)$. Suppose that $p$ can be expressed as a disjunction of minterms $p = p_{\vec{b}_1} \lor \ldots \lor p_{\vec{b}_k}$, where $\vec{b}_1, \ldots, \vec{b}_k \in \{0, 1\}^n$. Then, $\mu(p) = \sum_{j=1}^k \mu(p_{\vec{b}_j})$, because $p_{\vec{b}_j} \land p_{\vec{b}_h} = 0$ when $j \neq h$. On the other hand, $|\rho_p| = |\bigcup_j \rho_{p_{\vec{b}_j}}| = \sum_{j=1}^k |\rho_{p_{\vec{b}_j}}|$, so $\mu(p) = |\rho_p|$.

We shall refer to $\mu_\tau$ as the measure induced by the table $\tau$ on $\text{pol}(A)$.

Measures induced by tables are generated by pseudo-Boolean functions which range over the set $\mathbb{N}$ (see [7]). Namely, let $A = \{a_1, \ldots, a_n\}$ be a set of $n$ attributes. Define the pseudo-Boolean function $f : \{0, 1\}^n \rightarrow \mathbb{N}$ by $f(b_1, \ldots, b_n) = \mu_\tau(p_{b_1}, \ldots, b_n)$. Then, it is easy to verify that for every polynomial $p \in \text{pol}(A)$ we have

$$\mu_\tau(p) = \sum \{f(\vec{b}) | p^\vec{b} \in \text{mint}(A) \text{ and } p^\vec{b} \leq p\}. \quad (1)$$

Conversely, if $f : \{0, 1\}^n \rightarrow \mathbb{N}$ is an integer-valued, non-negative pseudo-Boolean function, then the function $\mu$ defined as in Equality (1) is clearly a measure on $\text{pol}(A)$.

In Section 3 we regard the set of minterms $\text{mint}(A)$ as a sample space and each polynomial $p \in \text{pol}(A)$ as an event on this sample space. The event $p$ occurs in $p_{\vec{b}}$ if $p^\vec{b} \leq p$. Thus, if $\mu$ is a measure on $\text{pol}(A)$, then the mapping $P_\mu : \text{pol}(A) \rightarrow \mathbb{R}$ given by $P_\mu(p) = \mu(p)/\mu(1)$ is a probability on $\text{pol}(A)$.

3. An inclusion–exclusion principle for MFBAs

Let $A = \{a_1, \ldots, a_n\}$ be a set of $n$ variables. If $I = \{i_1, \ldots, i_m\}$ is a subset of $\{1, \ldots, n\}$, then we denote the conjunction $a_{i_1} \land \cdots \land a_{i_m}$ by $a_I$. 

It is known that every polynomial \( p \in \text{pol}(A) \) can be uniquely written as

\[
p = \sum_I c_I \land a_I,
\]

where the summation \( \sum_I \) involves the “exclusive or” operation \( \oplus \) and is extended to all subsets \( I \) of \( \{1, \ldots, n\} \). The coefficients \( c_I \) belong to the set \( \{0, 1\} \). Thus, for a measure \( \mu \) on \( \text{pol}(A) \) it is interesting to evaluate \( \mu(p_1 \oplus p_2 \oplus \cdots \oplus p_m) \), where \( p_1, \ldots, p_m \) are polynomials in \( \text{pol}(A) \).

The indicator random variable of a polynomial \( p \) (see [5]) is the variable \( I_p \) defined by

\[
I_p(p_B) = \begin{cases} 1 & \text{if } p_B \subseteq p, \\ 0 & \text{otherwise} \end{cases}
\]

for \( p_B \in \text{mint}(A) \). Note that the expected value \( E[I_p] \) of \( I_p \) equals \( P_\mu(p) \).

If \( M = \{p_1, \ldots, p_n\} \) is a set of polynomials and \( J = \{j_1, \ldots, j_m\} \subseteq \{1, \ldots, n\} \), then \( p_{M,J} \) is the polynomial \( p_{j_1} \land \cdots \land p_{j_m} \); it is easy to see that \( I_{p_{M,J}} = I_{p_{j_1}} \cdots I_{p_{j_m}} \).

For a set of polynomials \( M \) define \( S^{\mu}_{M,K} \) as

\[
S^{\mu}_{M,K} = \sum\{P_\mu(p_{M,K}) \mid |K| = k\}.
\]

The number of \( k \)-subsets \( K \) of \( M \) such that \( p_{M,K} \) holds is given by the random variable \( \sum\{I_{p_{M,K}} \mid |K| = k\} \). By the previous observation

\[
S^{\mu}_{M,K} = \sum\{E(I_{p_{M,K}}) \mid |K| = k\} = E\left[\sum\{I_{p_{M,K}} \mid |K| = k\}\right].
\]

Let \( \nu_M \) be the random variable on \( \text{mint}(A) \) such that \( \nu_M(p_B) = |\{p_i \in M \mid p_B \subseteq p_i\}| \). Note that \( \nu_M \) gives the number of events in \( M \) that hold and, therefore, the random variable \( \left( \begin{smallmatrix} \nu_M \end{smallmatrix} \right)_k \) gives the number of \( k \)-subsets \( Q \) of \( M \) such that \( p_{M,Q} \) holds, which means that

\[
\sum\left( \begin{smallmatrix} \nu_M \end{smallmatrix} \right)_k = \sum\{I_{p_{M,K}} \mid |K| = k\} = S^{\mu}_{M,K}.
\]

Equality (2) is the basis of the method of indicators, that is a method of proving probabilistic identities by taking expectations of their non-probabilistic counterparts, see [5] for details.

**Theorem 3.1.** Let \( \mu : \text{pol}(A) \to \mathbb{R} \) be a measure on the free Boolean algebra \( \text{pol}(A) \), where \( A = \{a_1, \ldots, a_n\} \). If \( M = \{p_1, \ldots, p_m\} \) is a set of \( m \) polynomials of \( \text{pol}(A) \), then

\[
\mu(p_1 \oplus \cdots \oplus p_m) = \mu(1) \cdot \sum_{k=1}^m (-2)^{k-1} \cdot S^{\mu}_{M,k}.
\]

**Proof.** Let \( a \in \mathbb{N} \), note that \((-1)^a = \sum_{k=0}^a (-2)^k \binom{a}{k}\), which yields, after elementary transformations

\[
\sum_{k=1}^a (-2)^{k-1} \binom{a}{k} = (-1)^a - 1 = \begin{cases} 0 & \text{if } a \text{ is even}, \\ 1 & \text{if } a \text{ is odd}. \end{cases}
\]

This implies

\[
\sum_{k=1}^M (-2)^{k-1} \left( \begin{smallmatrix} \nu_M \\ k \end{smallmatrix} \right) = \sum_{k=1}^M (-2)^{k-1} \left( \begin{smallmatrix} \nu_M \\ k \end{smallmatrix} \right) = \begin{cases} 0 & \text{if } \nu_M \text{ is even}, \\ 1 & \text{if } \nu_M \text{ is odd}. \end{cases}
\]

By taking expectations of both sides, and using equality (2) we get

\[
E\left[\sum_{k=1}^M (-2)^{k-1} \left( \begin{smallmatrix} \nu_M \\ k \end{smallmatrix} \right)\right] = \sum_{k=1}^M (-2)^{k-1} S^{\mu}_{M,k} = P_\mu(\nu_M \text{ is odd}) = P_\mu(p_1 \oplus \cdots \oplus p_m).
\]

which yields the desired equality. \( \Box \)
Corollary 3.2. Let \( \mu, \mu' : \text{pol}(A) \to \mathbb{R} \) be two measures on the free Boolean algebra \( \text{pol}(A) \), where \( A = \{a_1, \ldots, a_n\} \). If \( \mu(p) = \mu'(p) \) for every conjunction \( p \) of the form \( p = a_1 \wedge \cdots \wedge a_m \), then \( \mu = \mu' \).

Proof. The result follows immediately from Theorem 3.1. \( \square \)

Example 3.3. Consider the “majority polynomial” \( p_{\text{maj}} = (a_1 \wedge a_2) \vee (a_2 \wedge a_3) \vee (a_1 \wedge a_3) \). For \( p_{\text{maj}} \) we have \( f_{p_{\text{maj}}} (x_1, x_2, x_3) = 1 \) if and only if at least two of its arguments are equal to 1. Note that

\[
 p_{\text{maj}} = (a_1 \wedge a_2) \oplus (a_2 \wedge a_3) \oplus (a_1 \wedge a_3).
\]

Theorem 3.1 allows us to write

\[
 \mu(p_{\text{maj}}) = \mu(a_1 \wedge a_2) + \mu(a_2 \wedge a_3) + \mu(a_1 \wedge a_3) \\
 - 2 \mu((a_1 \wedge a_2) \wedge (a_2 \wedge a_3)) - 2 \mu((a_1 \wedge a_2) \wedge (a_1 \wedge a_3)) \\
 - 2 \mu((a_2 \wedge a_3) \wedge (a_1 \wedge a_3)) + 4 \mu((a_1 \wedge a_2) \wedge (a_2 \wedge a_3) \wedge (a_1 \wedge a_2)).
\]

Corollary 3.2 shows that the values of a measure on \( \text{pol}(A) \) are completely determined by its values on positive conjunctions of the form \( a_I \) for \( I \subseteq \{1, \ldots, n\} \). Note that the contribution of every tuple of a table \( \tau = (T, A, \rho) \) of the form \( (b_1, \ldots, b_n) \) to the value of \( \mu_{\tau}(I) \) equals 1 for every set \( I \) such that \( I \subseteq \{i \in \{1, \ldots, n\} \mid b_i = 1\} \).

Next, we obtain Bonferroni-type inequalities [5] that give bounds on the value of \( \mu(p_1 \oplus \cdots \oplus p_m) \). To this end we need the following technical result:

Define \( W^a_b \) for \( a, b \in \mathbb{N} \) and \( b \leq a \) as \( W^a_b = \sum_{k=b}^{a} (-2)^{k-1} \binom{a}{k} \). Alternatively, \( W^a_b \) can be written as

\[
 W^a_b = (-2)^{b-1} \sum_{k=b}^{a} (-2)^{k-b} \binom{a}{k} = (-2)^{b-1} \sum_{\ell=0}^{a-b} (-2)^{\ell} \binom{a}{b+\ell}.
\]

Lemma 3.4. The signs of the members of the sequence \( (W^a_b, W^a_{b+1}, \ldots, W^a_a) \) are alternating.

Proof. Define

\[
 U^a_b = \sum_{\ell=0}^{a-b} (-2)^{\ell} \binom{a}{b+\ell}
\]

for \( a, b \in \mathbb{N} \) and \( b \leq a \). Since \( W^a_b = (-2)^{b-1} U^a_b \) it suffices to prove that the numbers \( U^a_b \) have all the same sign.

Note that \( U^a_b = 1 \) for \( b \in \mathbb{N} \). We can write

\[
 U^a_b = \sum_{\ell=0}^{a-b} (-2)^{\ell} \binom{a}{b+\ell} \\
 = \binom{a-1}{b} + \binom{a-1}{b-1} - 2 \binom{a-1}{b+1} - 2 \binom{a-1}{b} \\
 + 2^2 \binom{a-1}{b+2} + 2^2 \binom{a-1}{b+1} - 2^3 \binom{a-1}{b+3} - 2^3 \binom{a-1}{b+2} + \cdots \\
 : \\
 + (-2)^{a-b-1} \binom{a-1}{a-1} + (-2)^{a-b-1} \binom{a-1}{a-2} + (-2)^{a-b} \binom{a-1}{a-1} \\
 = \binom{a-1}{a-1} - U^{a-1}_b.
\]

Thus, we obtain

\[
 U^a_b = \binom{a-1}{b-1} - U^{a-1}_b.
\]
We claim that $0 \leq U_{ab} \leq \left( \frac{a}{b-1} \right)$ for $0 \leq b \leq a$. This can be shown by induction on $a \geq b$. The basis step $a=b$ is immediate. Suppose that the double inequality holds for $a-1$, that is, $0 \leq U_{a-1,b-1} \leq \left( \frac{a-1}{b-1} \right)$. Then, it is clear that $U_{ab} \geq 0$. To show that $U_{ab} \leq \left( \frac{a}{b-1} \right)$ we need to verify that $\left( \frac{a}{b-1} \right) - U_{a-1,b-1} \leq \left( \frac{a}{b-1} \right)$. Since $\left( \frac{a}{b-1} \right) = \left( \frac{a-1}{b-1} \right) + \left( \frac{a-1}{b-2} \right)$ for $b \geq 2$ the last inequality follows. □

**Theorem 3.5.** For any $r,s \in \mathbb{N}$ we have

$$\mu(1) \cdot \sum_{k=1}^{2r} (-2)^{k-1} s_k^a \leq \mu(p_1 \oplus \cdots \oplus p_m) \leq \mu(1) \cdot \sum_{k=1}^{2s+1} (-2)^{k-1} s_k^a.$$

**Proof.** By equality (2) and Lemma 3.4 we get that for any $r,s \in \mathbb{N}$

$$\sum_{k=1}^{2r} (-2)^{k-1} \left( \frac{a}{k} \right) \leq \sum_{k=1}^{a} (-2)^{k-1} \left( \frac{a}{k} \right) \leq \sum_{k=1}^{2s+1} (-2)^{k-1} \left( \frac{a}{k} \right),$$

implying

$$\sum_{k=1}^{2r} (-2)^{k-1} \left( \frac{vM}{k} \right) \leq \sum_{k=1}^{\lfloor M \rfloor} (-2)^{k-1} \left( \frac{vM}{k} \right) \leq \sum_{k=1}^{2s+1} (-2)^{k-1} \left( \frac{vM}{k} \right).$$

By applying expectations and using equality (2) we get the desired result. □

**Example 3.6.** Consider a table $\tau$ given below

<table>
<thead>
<tr>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

and the majority polynomial $p_{maj}$ from Example 3.3. We have $\mu(a_1 \land a_2) = 4$, $\mu(a_1 \land a_3) = 3$, $\mu(a_2 \land a_3) = 3$, giving $\mu(p_{maj}) \leq 10$. Also $\mu((a_1 \land a_2) \land (a_1 \land a_3)) = \mu((a_1 \land a_2) \land (a_2 \land a_3)) = \mu((a_1 \land a_3) \land (a_2 \land a_3)) = 1$ giving $\mu(p_{maj}) \geq 4$. The true value of $\mu(p_{maj})$ is 8.

4. Applications in data mining and database query optimization

In this section we examine the accuracy of the computation of the size of a query using the inclusion–exclusion principle. Then, we extend the notion of support for queries that apply to tables with missing values.

In database query optimization and in data mining, it is often necessary to estimate the number of rows in a database table satisfying a given query. Unfortunately, in most cases, the exact number of rows satisfying a query cannot be computed exactly and has to be estimated (usually using the assumption of statistical independence between attributes). Following datamining terminology we will occasionally refer to sets of attributes as itemsets.
Let \( \tau = (T, A, \rho) \) be a binary table and let \( K = \{a_{k_1}, \ldots, a_{k_m}\} \) be a set of attributes, \( K \subseteq A \). The support of the set \( K \) relative to the table \( \tau \) is the value of the probability \( \overset{\mu_\tau}{\text{P}} (a_{k_1} \land \cdots \land a_{k_m}) \):

\[
\overset{\text{supp}}{\mu}_\tau(K) = \frac{\{t \in \rho \mid t(a) = 1 \text{ for all } a \in K\}}{|\rho|}.
\]

In other words, the support of an attribute set \( K \) in the table \( \tau \) is defined by the value of the measure induced by the table on the Boolean polynomial that describes the attribute set. By extension, we can regard the number \( \mu_\tau(q)/\mu_\tau(1) \) as the support of the query \( q \) and we denote this number by \( \overset{\text{supp}}{\mu}_\tau(q) \). Indeed, if \( q \in \text{pol}(A) \) is a query involving a table \( \tau = (T, A, \rho) \) such that \( q \) can be written as

\[
q = c \oplus \sum_{t \in \mathcal{S}} a_t,
\]

where \( c \in \{0, 1\} \) and \( \mathcal{S} \) is a collection of subsets of \( \{1, \ldots, n\} \), then \( \overset{\text{supp}}{\mu}_\tau(q) \) can be obtained from Theorem 3.1 using the numbers \( \overset{\text{supp}}{\mu}_\tau(a_t) \). Methods that obtain approximative estimations of query sizes been proposed [12], including the use of Maximum-Entropy Principle. An open problem raised was estimating the quality of such an approximation.

The computation of the size of the query using Theorem 3.1 can be often simplified if there is a known maximal number of 1 components in the tuples of the table. For example, in a store that sells 1000 items (corresponding to 1000 attributes in a table that contains the records of purchases) it is often the case that we can use an empirical limit of, say, 8 items per tuple. In this case, conjunctions that contain more than 8 conjuncts can be discarded and the estimation is considerably simplified. Even, if such an upper bound cannot be imposed a priori, it is often the case that we can discard large conjunctions (which have low support). However, there are some risks when approximations of this nature are performed due to the large values of coefficients that multiply the supports for large conjunctions.

Indeed, consider the tables \( \tau^n_{\text{odd}} = (T_o, A, \rho_{\text{odd}}) \), \( \tau^n_{\text{even}} = (T_e, A, \rho_{\text{even}}) \), where

\[
\rho_{\text{odd}} = \{t \in \text{Dom}(A) \mid n_1(t) \text{ is odd}\} \quad \text{and} \quad \rho_{\text{even}} = \{t \in \text{Dom}(A) \mid n_1(t) \text{ is even}\},
\]

where \( n_1(t) \) denotes the number of attributes equal to 1 in tuple \( t \) and \( |A| = n \).

Note that for proper subset \( K \) of \( A \), we have \( \overset{\text{supp}}{\mu}_{\text{odd}}(K) = \overset{\text{supp}}{\mu}_{\text{even}}(K) \), while

\[
\overset{\text{supp}}{\mu}_{\text{odd}}(A) = \begin{cases} 1 & \text{if } n \text{ is odd,} \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \overset{\text{supp}}{\mu}_{\text{even}}(A) = \begin{cases} 1 & \text{if } n \text{ is even,} \\ 0 & \text{otherwise.} \end{cases}
\]

Thus, from the point of view of the supports of any proper subset of the attribute set the tables \( \tau^n_{\text{odd}} \) and \( \tau^n_{\text{even}} \) are indiscernible. However, the support of certain queries can be vastly different on these tables. For example, consider the polynomial \( p = a_1 \oplus a_2 \oplus \cdots \oplus a_n \). We have \( \overset{\text{supp}}{\mu}_{\text{odd}}(p) = 1 \) and \( \overset{\text{supp}}{\mu}_{\text{even}}(p) = 0 \). So, ignoring the term that corresponds to the support for a single attribute set (note that this is also the attribute set with the smallest possible support) has a huge impact on \( \mu_\tau(p) \). Note that the result is consistent with (Theorem 3.1) which gives the set of attributes \( A \) a coefficient \( 2^{n-1} \). We stress however that the negative result above does not rule out practical applicability of approximating the values of \( \mu_\tau \) since the parity function query used above is by no means a typical database query.

Frequently, real-world datasets contain missing values; this makes important to adequately address this issue. Here we present a generalization of the notion of support which takes missing values into account. The idea is related to the hot deck imputation of missing values where each missing value is replaced by a value randomly drawn from some distribution (see [10]).

Suppose that \( \tau = (T, A, \rho) \) is a table such that \( A = \{a_1, \ldots, a_n\} \) and \( \text{Dom}(a_i) = \{0, u, 1\} \) for \( 1 \leq i \leq n \). The symbol \( u \) represents null values, that is, values that are missing or undefined. With every attribute \( a_i \in A \) we associate a real number \( z_i \in [0, 1] \). Intuitively, this number corresponds to the probability of \( a_i = 1 \), and can be obtained using the non-missing values for the attribute or based on background knowledge.

Let \( z \) be a non-negative number, and let \( b, c \in \{0, 1\} \). Define

\[
z^{(b,c)} = \begin{cases} z & \text{if } b = 1 \text{ and } c = 0 \\ 1 - z & \text{if } b = 0 \text{ and } c = 0 \\ 1 & \text{if } c = 1, \end{cases} \quad \text{and} \quad b^{(c)} = \begin{cases} b & \text{if } c = 1, \\ u & \text{if } c = 0, \end{cases}
\]
Theorem 4.3. Such that the exclusion principle automatically applies to a collection of sets of attributes, then their values of $\mu^d$ are probabilistically consistent. Other approaches to mining frequent itemsets in the presence of missing values can be found in [14,17]. However, both these approaches can produce probabilistically inconsistent results. Specifically, the technique used in [17] is to count the support of an itemset only on the portion of the table where it is
valid. For example, consider the table \( \tau = (T, a_1a_2, \rho) \), given by

<table>
<thead>
<tr>
<th>( a_1 )</th>
<th>( a_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>u</td>
</tr>
<tr>
<td>0</td>
<td>u</td>
</tr>
<tr>
<td>0</td>
<td>u</td>
</tr>
</tbody>
</table>

Using the method from [17] the support of attribute \( a_2 \) is counted only in the first row, giving \( \text{supp}_\tau(a_2) = 100\% \). Similarly \( \text{supp}_\tau(a_1) = 50\% \), and \( \text{supp}_\tau(a_1a_2) = 100\% \), but this means \( \text{supp}_\tau(a_1a_2) > \text{supp}_\tau(a_1) \), which is impossible. In the method proposed in [14] the probability for each attribute is estimated from the part of the data where the attribute is defined. When computing how much support does a row with a missing value contribute for an itemset, this probabilities are summed for each attribute (see [14] for details). In the table above this will give \( \text{supp}_\tau(a_1) = 50\% \), \( \text{supp}_\tau(a_2) = 100\% \), and \( \text{supp}_\tau(a_1a_2) = \left(0.5 \cdot 1 + 0.5 \cdot 1\right) + \left(0.5 \cdot 1 + 0.5 \cdot 1\right) + 2\left(0.5 \cdot 0 + 0.5 \cdot 1\right)/4 = 75\% \), and \( \text{supp}_\tau(a_1a_2) > \text{supp}_\tau(a_1) \). Using our \( \mu^g \) measure with \( x_2 = 1 \) gives consistent values of \( \text{supp}_\tau(a_1) = 50\% \), \( \text{supp}_\tau(a_2) = 100\% \), and \( \text{supp}_\tau(a_1a_2) = 50\% \).

5. Support approximations using Bonferroni-type inequalities

The question of estimating supports of general Boolean expressions based on supports of frequent itemsets discovered by a datamining algorithm was initiated in [13]. The accuracy of this estimation (using the inclusion–exclusion principle) is influenced by the supports of the frequent items set; when, for various reasons, some of these supports are missing this accuracy may be compromised. The problem has been addressed in [9,11] but the results presented there can be applied only for the case when we know supports of all itemsets up to a given size. This is usually not the case with datamining algorithms which compute supports of only some of the itemsets of a given size. A similar problem has been addressed in the area of statistical data protection, where it is important to assure that inferences about individual cases cannot be made from marginal totals (see [3,4] for an overview). Those methods concentrate on obtaining the most accurate bounds possible (in order to rule out information disclosure), computational efficiency being a secondary concern. Algorithms usually involve repeated iterations over full contingency tables [3], branch and bound search [4] or numerous applications of linear programming.

We use recursively Bonferroni inequalities to estimate supports of missing itemsets. In their original form the inequalities require that we know supports of all itemsets up to a given size. We address the problem by using the inequalities recursively to estimate supports of missing itemsets. The advantage of Bonferroni inequalities is that we can choose an arbitrary limit on the size of the marginals involved, thus allowing for trading off accuracy for speed. Our experiments revealed that it is possible to obtain good bounds even if only marginals of small size are used.

Example 5.1. Consider a binary table \( \tau \) whose heading is \( A = abc \) and assume that the distribution of the values of the tuples in this table is given by

\[
\begin{array}{cccccccc}
   & a & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
   b & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
   c & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

Frequency

\[
\begin{array}{cccccc}
   & 0 & 0 & 0.1 & 0.25 & 0.1 & 0.25 & 0.05 & 0.25 \\
\end{array}
\]

A run of the Apriori algorithm [1] on a dataset conforming to that distribution, with the minimum support of 0.35 will yield the following itemsets:

<table>
<thead>
<tr>
<th>Itemset</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>ac</th>
<th>bc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>0.65</td>
<td>0.65</td>
<td>0.75</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>
To estimate the unknown support of the itemset abc we can use Bonferroni inequalities of the form

\[ \text{supp}_T(abc) \geq 1 - \text{supp}_T(\bar{a}) - \text{supp}_T(\bar{b}) - \text{supp}_T(\bar{c}), \]

(5)

\[ \text{supp}_T(abc) \leq 1 - \text{supp}_T(\bar{a}) - \text{supp}_T(\bar{b}) - \text{supp}_T(\bar{c}) + \supp_T(\bar{ab}) + \supp_T(\bar{ac}) + \supp_T(\bar{bc}). \]

(6)

Note that since the support of ab is below the minimum support its value is not returned by the Apriori algorithm and this creates a problem for this estimation. All the itemset supports, except for \( \supp_T(\bar{ab}) \), in the previous expression can be determined from known itemset supports using inclusion–exclusion principle. For example, we have

\[ \supp_T(\bar{ac}) = 1 - \supp_T(a) - \supp_T(c) + \supp_T(ac) = 0.1. \]

Since all needed probabilities are known exactly, the lower bound (5) is easy to compute giving

\[ \supp_T(abc) \geq 1 - 0.35 - 0.35 - 0.25 = 0.05. \]

To compute the upper bound we proceed as follows. Since \( \supp_T(\bar{ab}) \) is not known, we apply Bonferroni inequalities recursively to get an upper bound for it. We have

\[ \supp_T(\bar{ab}) = 1 - \supp_T(a) - \supp_T(b) + \supp_T(ab), \]

and, since \( ab \) is not frequent, we know that its support is less than the 0.35 minimum support, giving

\[ \supp_T(\bar{ab}) < 1 - \supp_T(a) - \supp_T(b) + \minsupp = 0.05. \]

Substituting into (7) we get

\[ \supp_T(abc) < 1 - \supp_T(\bar{a}) - \supp_T(\bar{b}) - \supp_T(\bar{c}) + 0.05 + \supp_T(\bar{ac}) + \supp_T(\bar{bc}) = 1 - 0.35 - 0.35 - 0.25 + 0.05 + 0.1 + 0.1 = 0.3. \]

Note that both bounds are not trivial since the lower bound is greater than 0, and the upper bound is less than the minimum support.

5.1 A recursive procedure for computing Bonferroni bounds from frequent itemsets

Since the Apriori algorithm only discovers supports of itemsets (as opposed to other types of queries), we need to express all inequalities in terms of supports of itemsets.

**Theorem 5.2.** Let \( q_1, \ldots, q_m \) be \( m \) queries in \( \text{pol}(A) \). The following inequalities hold for any \( t \in \mathbb{N} \):

\[
\sum_{k=0}^{2t+1} (-1)^k \sum_{r < i_1 < \cdots < i_k \leq m} \supp_T(q_1 \land \cdots \land q_r \land q_{i_1} \land \cdots \land q_{i_k}) \leq \supp_T(q_1 \land \cdots \land q_r \land \bar{q}_r+1 \land \cdots \land \bar{q}_m) \leq \sum_{k=0}^{2t} (-1)^k \sum_{r < i_1 < \cdots < i_k \leq m} \supp_T(q_1 \land \cdots \land q_r \land q_{i_1} \land \cdots \land q_{i_k}).
\]

**Proof.** By Rényi’s Theorem [16] it suffices to prove the claim for \( q_i \in \{1, 0\} \) for all \( 1 \leq i \leq m \). When \( q_i = 0 \) for some \( 1 \leq i \leq r \), then both sides of the inequalities reduce to 0 and the result is immediate. For the case \( q_i = 1 \) for all \( 1 \leq i \leq r \) we have \( \supp_T(q_1 \land \cdots \land q_r \land \bar{q}_r+1 \land \cdots \land \bar{q}_m) = \supp_T(q_r+1 \cdots \bar{q}_m) \), and for all \( k \) and for all \( r < i_1 < \cdots < i_k \leq m, \supp_T(q_1 \land \cdots \land q_r \land q_{i_1} \land \cdots \land q_{i_k}) = \supp_T(q_1 \land \cdots \land q_{i_k}). \) The result now follows from Bonferroni inequalities. \( \square \)
Corollary 5.3. Let \( a_1 \wedge a_2 \wedge \cdots \wedge a_r \wedge \bar{a}_{r+1} \wedge \bar{a}_{r+2} \wedge \cdots \wedge \bar{a}_m \) be a minterm. The following inequalities hold for any natural number \( t \):

\[
\sum_{k=0}^{2t+1} (-1)^k \sum_{r<i_1<\cdots<i_k \leq m} \text{supp}_\tau (a_1 \wedge \cdots \wedge a_r \wedge a_{i_1} \wedge \cdots \wedge a_{i_k}) \\
\leq \text{supp}_\tau (a_1 \wedge \cdots \wedge a_r \wedge \bar{a}_{r+1} \wedge \cdots \wedge \bar{a}_m) \\
\leq \sum_{k=0}^{2t} (-1)^k \sum_{r<i_1<\cdots<i_k \leq m} \text{supp}_\tau (a_1 \wedge \cdots \wedge a_r \wedge \bar{a}_{r+1} \wedge \cdots \wedge \bar{a}_m).
\]

Proof. This statement follows immediately from Theorem 5.2. □

Below we present results which form the basis of our algorithm for approximative computations of supports of itemsets. The binomial symbol \( \binom{n}{k} \) will allow negative values of \( n \), in which case its value is defined by the usual formula

\[
\binom{n}{k} = \frac{n(n-1) \cdots (n-k+1)}{k!}.
\]

Lemma 5.4. For \( m, k, h, s \in \mathbb{N} \) such that \( m > s \) we have

\[
\sum_{k=0}^{s} (-1)^{s-k} \binom{m-k-1}{s-k} \binom{h}{k} = \binom{h-m+s}{s}.
\]

Proof. In [6, p. 169], it is shown that for every \( a, b, c, d \in \mathbb{N} \) we have

\[
\sum_{k=0}^{a} (-1)^k \binom{a-k}{b} \binom{c}{k-d} = (-1)^{a+b} \binom{c-b-1}{a-b-d}. \tag{7}
\]

By using the complimentary combinations in Equality (7) we can write

\[
\sum_{k=0}^{s} (-1)^{s-k} \binom{m-k-1}{s-k} \binom{h}{k} = \sum_{k=0}^{s} (-1)^{s+k} \binom{m-k-1}{m-s-1} \binom{h}{k} \\
= (-1)^s \cdot \sum_{k=0}^{s} (-1)^k \binom{m-k-1}{m-s-1} \binom{h}{k} \\
= (-1)^s \cdot (-1)^{2m-2-s} \binom{h-m+s}{s} \\
= \binom{h-m+s}{s}.
\]

Note that the application of the formula

\[
\binom{m-k-1}{s-k} = \binom{m-k-1}{m-s-1}
\]

in the above chain of equalities is justified because \( m-k-1 \leq m-s-1 \geq 0 \). □

Note that if \( h = m \), the previous lemma implies

\[
\sum_{k=0}^{s} (-1)^{s-k} \binom{m-k-1}{s-k} \binom{m}{k} = 1.
\]

Our method of obtaining bounds is based on the following theorem.
Theorem 5.5. Let \( \tau = (T, A, \rho) \) be a table and let \( a_1, \ldots, a_m \) be attributes in \( A \). Then, if \( m > 2t \) we have
\[
\supp_\tau(a_1 \land a_2 \land \cdots \land a_m) \leq \sum_{k=0}^{2t} (-1)^k \binom{m-k-1}{2t-k} S_k,
\]
and if \( m > 2t + 1 \)
\[
\supp_\tau(a_1 \land a_2 \land \cdots \land a_m) \geq \sum_{k=0}^{2t+1} (-1)^{k+1} \binom{m-k-1}{2t+1-k} S_k,
\]
where
\[
S_k = \sum_{1 \leq i_1 < \cdots < i_k \leq m} \supp_\tau(a_{i_1} \land \cdots \land a_{i_k})
\]
and \( S_0 = 1 \).

Proof. We use the method of indicators previously discussed.

Let \( v_m \) be a random variable equal to the number of events \( A_1, \ldots, A_m \) that actually occur. By Lemma 5.4 we have
\[
\sum_{k=0}^{s} (-1)^{s-k} \binom{m-k-1}{s-k} \left( \begin{array}{c} v_m - m + s \cr s \end{array} \right) = \begin{cases} 1 & \text{if } v_m = m, \\ 0 & \text{if } v_m < m \text{ and } v_m \geq m - s, \\ v_m - m + s & \text{if } v_m < m - s. \end{cases}
\]
By taking expectations of the above equation we get
\[
\sum_{k=0}^{s} (-1)^{s-k} \binom{m-k-1}{s-k} S_k = \supp_\tau(v_m = m)
+ \sum \left\{ \left( \begin{array}{c} v_m(\omega) - m + s \cr s \end{array} \right) \supp_\tau(\omega) : \omega \in \Omega, v_m(\omega) < m - s \right\},
\]
where \( \Omega \) denotes the space of elementary events. Note that when \( v_m < m - s \) the sign of \( \left( \begin{array}{c} v_m - m + s \cr s \end{array} \right) \) is identical to that of \( (-1)^s \). Replacing \( s \) by \( 2t \) or \( 2t + 1 \) yields the result. \( \square \)

6. The estimation algorithm

The main problem in using Bonferroni-type inequalities on collections of frequent itemsets is that some of the probabilities in the \( S_k \) sums are not known. We solved this problem by estimating the missing probabilities using Theorem 5.5. In Fig. 1 we give an algorithm for computing bounds on support of an itemset based on a collection of itemsets with known supports.

Of course upper and lower bounds for itemsets are cached during computations to avoid repeated evaluations for the same itemset. The parameter \( r \) controls the maximum size of marginals (itemsets) used in the estimation.

The use of \( \minsupp \) in step 5 of function \( U \) requires some comment. Including the value of \( \minsupp \) in the minimum is possible only if we can determine that the estimated itemset \( I \) is not frequent. This can be done for example, if \( \mathcal{F} \) contains all frequent itemsets, or when \( \mathcal{F} \) contains all frequent itemsets up to a given size \( k \), and \( |I| \leq k \). If we do not know whether \( I \) is frequent or not, we have to drop \( \minsupp \) from the minimum.

6.1. Experimental results

In this section we present experimental evaluation of the bounds. Our algorithm works best on dense datasets, which are more difficult to mine for frequent itemsets than sparse ones. However, the algorithm was tested on both dense and sparse data (artificial market basket data was used). The rest of the paper is focused on experiments performed on dense databases.
As dense databases we used the mushroom database from the UCI Machine Learning Archive [2], and a census data of elderly people from the University of Massachusetts at Boston Gerontology Center available at http://www.cs.umb.edu/sj/datasets/census.arff.gz.

Since both datasets involve multivalued attributes, we replaced each attribute (including binary ones) with a number of Boolean attributes, one for each possible value of the original attribute.

Before we present a detailed experimental study of the quality of bounds, we present the results of applying the bounds to a practical task. Suppose that we did not have enough time or computational resources to run the Apriori (or similar) algorithm completely, and we decided to stop the algorithm after finding frequent itemsets of size less than or equal to 2. We then use lower bounds to find frequent itemsets of size greater than 2. The experimental results for mushroom and census databases are shown in Figs. 2 and 3 respectively.

The figures show, for various values of minimum support, the true number of frequent itemsets of sizes 3 and 4, the number of itemsets that we discovered to be frequent by using our bounds, and the ratio of the two numbers.

For large values of minimum support we are more likely to classify an itemset correctly than for smaller ones. The data shows that for itemsets with largest support the chances of actually being determined to be frequent without consulting the data can be as high as 80%.

We now present an experimental analysis of the bounds obtained. In what follows, by trivial bounds for the support of an itemset I we mean 0 for the lower bound, and for the upper bound: the minimum of the upper bounds of the supports of all proper

Algorithm 1
Input: Itemset I, natural number r, collection F of itemsets, and their supports
Output: Bounds L(I), U(I) on the support of I
The algorithm is implemented by functions L and U given below

Function L(I, F, r).
1. If I ∈ F
2. return supp_r(I)
3. else
4. return max_{-1≤2t+1≤r} \sum_{k=0}^{2t+1} S_L^{(t+1)} \binom{m-k-1}{2t+k} I, F, k)

Function U(I, F, r).
1. If I ∈ F
2. return supp_r(I)
3. else
4. U \rightarrow \min_{0≤2t≤r} \sum_{k=0}^{2t} S_U^{t} \binom{m-k-1}{2t-k} I, F, k)
5. U \rightarrow \min\{U, \min_{J⊆I} U(J)\}
6. return U

The functions S_L and S_U are defined below

Function S_L(c, itemset I = a_1 a_2 ... a_m, F, integer k)
1. If k = 0 return c
2. If c ≥ 0
3. return c \cdot \sum_{i_1 < ... < i_k ≤ m} L(a_i_1 a_i_2 ... a_i_k, F, k - 1)
4. else
5. return c \cdot \sum_{i_1 < ... < i_k ≤ m} U(a_i_1 a_i_2 ... a_i_k, F, k - 1)

Function S_U(c, itemset I = a_1 a_2 ... a_m, F, integer k)
1. If k = 0 return c
2. If c ≥ 0
3. return c \cdot \sum_{i_1 < ... < i_k ≤ m} U(a_i_1 a_i_2 ... a_i_k, F, k - 1)
4. else
5. return c \cdot \sum_{i_1 < ... < i_k ≤ m} L(a_i_1 a_i_2 ... a_i_k, F, k - 1)
subsets of $I$ and of the minimum support. As in the example above here too we mine frequent itemsets with at most two items, and compute bounds for larger ones.

Table 1 (a) contains the results for the census dataset with minimum support of 1.8%.

The parameter $r$ in Algorithm 1 was chosen for each itemset $I$ to be $|I| - 1$ for maximum accuracy. This causes an increase in estimation time for larger itemsets. Later in the section we present results showing that limiting the value of $r$ can give very fast estimates with a very small impact on the quality of the bounds. All experiments were run on a 100 MHz Pentium machine with 64 MB of memory.

The bounds obtained are fairly accurate. The width of the interval between the lower and upper bounds varied from 0.048 to 0.019 for itemsets of size 3. Note that the estimates become more and more accurate for larger itemsets. The reason is that the bulk of large itemsets will have subsets whose support is very small, thus giving better average trivial bounds. Nontrivial upper bounds occur slightly more frequently than nontrivial lower bounds; however, lower bounds give on average much better improvement over the trivial bounds (this is due to the fact that our trivial upper bounds are quite sophisticated, while the trivial lower bound is just assumed to be 0).

The percentage of itemsets having nontrivial bounds is quite small. However those itemsets who have high support (and thus are the most interesting) are more likely to get interesting nontrivial bounds. This can be seen in Tables 1(b) and (c), where up to 48% of itemsets have nontrivial bounds proving the usefulness of Theorem 5.5. Note that in this case the interval width increases.
Table 1
Results for the census dataset

<table>
<thead>
<tr>
<th>Itemset size</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 1.8% minimum support, all itemsets</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average interval width</td>
<td>0.0482797</td>
<td>0.0313103</td>
<td>0.0228579</td>
<td>0.0196316</td>
</tr>
<tr>
<td>Average upper bound</td>
<td>0.0568679</td>
<td>0.0319395</td>
<td>0.0228771</td>
<td>0.0196316</td>
</tr>
<tr>
<td>Average lower bound</td>
<td>0.00858817</td>
<td>0.000629199</td>
<td>1.925e-05</td>
<td>0</td>
</tr>
<tr>
<td>Itemsets with nontrivial bounds (%)</td>
<td>7.04</td>
<td>0.59</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>Itemsets with nontrivial lower bound (%)</td>
<td>4.06</td>
<td>0.39</td>
<td>0.07</td>
<td>—</td>
</tr>
<tr>
<td>Average lower improvement</td>
<td>0.211321</td>
<td>0.161151</td>
<td>0.0962518</td>
<td>—</td>
</tr>
<tr>
<td>Average upper improvement</td>
<td>0.0225656</td>
<td>0.00983444</td>
<td>0.00262454</td>
<td>—</td>
</tr>
<tr>
<td>Time (ms/itemset)</td>
<td>0.2</td>
<td>0.3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

(b) 1.8% minimum support, frequent itemsets only

| Average interval width | 0.102848 | 0.105024 | 0.106997 | 0.110767 |
| Average upper bound | 0.127438 | 0.109572 | 0.107491 | 0.110767 |
| Average lower bound | 0.0245896 | 0.00454846 | 0.00049354 | 0 |
| Itemsets with nontrivial bounds (%) | 20.17 | 4.25 | 0.58 | 0.02 |
| Itemsets with nontrivial lower bound (%) | 11.64 | 2.82 | 0.46 | — |
| Average lower improvement | 0.211321 | 0.161151 | 0.106164 | — |
| Average upper improvement | 0.0225656 | 0.00983444 | 0.00333985 | 0.00338427 |

(c) 9% minimum support, frequent itemsets only

| Average interval width | 0.171608 | 0.205194 | 0.222602 | 0.231362 |
| Average upper bound | 0.235004 | 0.223174 | 0.225491 | 0.231362 |
| Average lower bound | 0.0633963 | 0.0179804 | 0.00288882 | 0 |
| Itemsets with nontrivial bounds (%) | 48.55 | 16.79 | 3.40 | 0.14 |
| Itemsets with nontrivial lower bound (%) | 30.00 | 11.16 | 2.72 | — |
| Average lower improvement | 0.211321 | 0.161151 | 0.106164 | — |
| Average upper improvement | 0.0225656 | 0.00983444 | 0.00333985 | 0.00338427 |

with the size of the itemsets. This is due to the fact that for high supports we do not have large number of itemsets with low supports that would create trivial upper bounds.

The conclusions were analogous for the mushroom database.

Table 2 shows how the choice of the argument \( r \) in Algorithm 1 influences the computation speed and the quality of the bounds. The results when \( r \) is set to the highest possible value (size of the estimated itemset minus one) is given in Table 1(a).

The results show that limiting the value of \( r \) to 2 or 3 gives a large speedup at a negligible decrease in accuracy. This is the approach we recommend. Also note that the proportion of itemsets with nontrivial bounds is higher for lower values of \( r \).

Our last experimental result concerns estimating support of conjunctions allowing negated items using Corollary 5.3. Table 3 shows the results for the census dataset, with supports of all frequent 1- and 2-itemsets known (1.8% minimum support). In each of the itemsets exactly two of the items were negated. Again the inequalities gave fairly tight bounds.

7. Conclusions and open problems

We studied properties of measures defined on free Boolean algebras arising naturally in the evaluation of sizes of queries applied to binary tables in relational databases. A method of obtaining bounds for support of database queries based on supports of frequent itemsets discovered by a datamining algorithm was presented by generalizing the Bonferroni inequalities. Specialized bounds for estimating support of itemsets, itemsets with negated items, as well as bounds for arbitrary queries have been obtained.

An experimental evaluation of the bounds shows that the bounds are capable of providing useful approximations.

Various other specialized Bonferroni inequalities for other types of queries could be considered. General inequalities in Theorem 3.1 can be used for this but the bounds they give are not always tight. It has also been shown in [8] that for certain
Table 2
Influence of the order of inequalities on the bounds

<table>
<thead>
<tr>
<th>Itemset size</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>r = 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average interval width</td>
<td>0.0482797</td>
<td>0.0315442</td>
<td>0.022993</td>
<td>0.0196671</td>
</tr>
<tr>
<td>Average upper bound</td>
<td>0.0568679</td>
<td>0.0321734</td>
<td>0.0230122</td>
<td>0.0196671</td>
</tr>
<tr>
<td>Average lower bound</td>
<td>0.00858817</td>
<td>0.000629199</td>
<td>1.925e-05</td>
<td>0</td>
</tr>
<tr>
<td>Itemsets with nontrivial bounds (%)</td>
<td>7</td>
<td>1</td>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>Time (ms/itemset)</td>
<td>0.18</td>
<td>0.24</td>
<td>0.34</td>
<td>0.46</td>
</tr>
</tbody>
</table>

| r = 3       |          |          |          |          |
| Average interval width | 0.0482797 | 0.0313103 | 0.0228666 | 0.0196328 |
| Average upper bound   | 0.0568679 | 0.0319395 | 0.0228859 | 0.0196328 |
| Average lower bound   | 0.00858817| 0.000629199| 1.925e-05| 0         |
| Itemsets with nontrivial bounds (%) | 7 | 0.50 | 0 | 0 |
| Time (ms/itemset) | 0.18 | 0.50 | 0.53 | 0.92 |

Census data with 1.8% minimum support.

Table 3
Estimates for itemsets with negations

<table>
<thead>
<tr>
<th>Itemset size</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average interval width</td>
<td>0.040498</td>
<td>0.081989</td>
<td>0.0668155</td>
<td>0.0392651</td>
<td>0.0180174</td>
</tr>
<tr>
<td>Average upper bound</td>
<td>0.171319</td>
<td>0.120666</td>
<td>0.0685168</td>
<td>0.0392925</td>
<td>0.0180174</td>
</tr>
<tr>
<td>Average lower bound</td>
<td>0.130821</td>
<td>0.0386768</td>
<td>0.00170127</td>
<td>2.73405e-05</td>
<td>0</td>
</tr>
<tr>
<td>Time (ms/itemset)</td>
<td>0.24</td>
<td>0.46</td>
<td>0.96</td>
<td>2.54</td>
<td>5.12</td>
</tr>
</tbody>
</table>

Census data with 1.8% minimum support.

queries it is not possible to obtain tight bounds at all. Nevertheless, we believe that it is possible to obtain useful bounds for a large family of practically useful queries.

Another important direction of future research is the investigation of various other types of inequalities (e.g., sharp Bonferroni inequalities) to improve the tightness of currently available bounds. An interesting challenge would be applying other (besides the method of indicators) methods of proving inequalities presented in [5] like the method of polynomials or the geometric method.

References