BLOSOM: A Framework for Mining Arbitrary Boolean Expressions

Lizhuang Zhao[†], Mohammed J. Zaki[†], Naren Ramakrishnan[‡]
[†] CS Dept., Rensselaer Polytechnic Institute, Troy, NY 12180
[‡] CS Dept., Virginia Tech., Blacksburg, VA 24061
{zhaol2, zaki}@cs.rpi.edu, naren@cs.vt.edu

ABSTRACT

We introduce a novel framework, called BLOSOM, for mining (frequent) boolean expressions over binary-valued datasets. We organize the space of boolean expressions into four categories: pure conjunctions, pure disjunctions, conjunction of disjunctions, and disjunction of conjunctions. We focus on mining the simplest expressions (the *minimal generators*) for each class. We also propose a *closure* operator for each class that yields *closed* boolean expressions. BLOSOM efficiently mines frequent boolean expressions by utilizing a number of methodical pruning techniques. Experiments showcase the behavior of BLOSOM, and an application study on a real dataset is also given.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining

General Terms: Algorithms.

Keywords: Minimal Generator, Closed Itemsets, Boolean Expression, Data Mining

1. INTRODUCTION

One of the basic goals of data mining is to discover novel patterns that are potentially useful. For example, itemset mining [1] discovers patterns that are *pure conjunctions* of items. However, this covers a very restricted subset of the wider class of boolean patterns, which may consist of conjunctions, disjunctions, and negations of items. Mining such boolean patterns can lead to significant nuggets of knowledge, with many potential applications in market basket analysis, web usage mining, social network analysis, and bioinformatics.

Boolean expression mining can provide tremendous value, but there are two main challenges to contend with. The first deals with the problem of high computational complexity. With n items (or variables), there are 2^{2^n} possible distinct-valued boolean expressions, far too many to enumerate. To render the search tractable we focus on only the frequent boolean expressions. Also instead of mining all frequent boolean expressions, we focus on mining a lossless subset that retains complete frequency information, namely *closed* boolean expression. The second challenge relates to comprehension of the patterns, i.e., they may be complex and difficult to understand. Here we focus on mining the simplest or *minimal* expressions (which are in fact the *minimal generators* of the closed expressions) that still from a lossless representation of all possible boolean expressions.

In this paper, we present a novel framework, called BLOSOM (an anagram of the bold letters in **BOOL**ean expression **M**ining over attribute **S**ets), the first such approach to simultaneously mine closed boolean expressions over attribute sets and their minimal generators. Our main contributions are as follows: We organize boolean expressions into four categories: (i) pure conjunctions (AND-clauses), (ii) pure disjunctions (OR-clauses), (iii) conjunctive normal form (conjunction of disjunctions), and (iv) disjunctive normal form (disjunction of conjunctions). For each class of expressions, we propose a *closure* operator, and we give a characterization of the *minimal generators*. BLOSOM employs a number of effective pruning techniques for searching over the space of boolean expressions, yielding orders of magnitude in speedup. We also highlight some of the patterns found using BLOSOM on real datasets.

2. PRELIMINARY CONCEPTS

Lattice Theory: Let's first review a few facts from lattice theory [5], which will be useful in our discussion. Let $(P, \subseteq)$ be a partially ordered set (also called a *poset*). Let $X, Y \in P$ and let $f : P \rightarrow P$ be a function on P. f is called *monotone* if $X \subseteq Y \Rightarrow f(X) \subseteq f(Y)$. We say that f is *idempotent* if $f(X) = f(f(X))$. f is called *extensive* (or expansive) if $X \subseteq f(X)$. Finally, f is called *intensive* (or contractive) if $f(X) \subseteq X$. A *closure operator* on P is a function $\mathbb{C} : P \rightarrow P$ such that $\mathbb{C}$ is monotone, idempotent, and extensive. X is called *closed* if $\mathbb{C}(X) = X$. On the other hand, a *kernel operator* on P is a function $\mathbb{K} : P \rightarrow P$, which is monotone, idempotent, and intensive. X is called *open* if $\mathbb{K}(X) = X$. The set of all closed and open members of P form the fixed-point of the closure ($\mathbb{C}$) and kernel ($\mathbb{K}$) operators, respectively. Given two posets $(P, \subseteq)$ and $(Q, \leq)$, a *monotone Galois connection* between them consists of two *order-preserving* functions, $\phi : P \rightarrow Q$ and $\psi : Q \rightarrow P$, such that for all $X \in P$ and $Y \in Q$, we have: $X \subseteq \psi(Y) \iff \phi(X) \leq Y$. The composite function $\psi \circ \phi : P \rightarrow P$ is a closure operator, whereas the function $\phi \circ \psi : Q \rightarrow Q$ is a kernel operator, on P and Q, respectively [5]. Given posets $(P, \subseteq)$ and $(Q, \leq)$, a *anti-monotone Galois connection* [5] between them consists of two *order-reversing* functions, $\phi : P \rightarrow Q$ and $\psi : Q \rightarrow P$, such that for all $X \in P$ and $Y \in Q$, we have: $X \subseteq \psi(Y) \iff Y \leq \phi(X)$. The composite functions $\psi \circ \phi : P \rightarrow P$ and $\phi \circ \psi : Q \rightarrow Q$ are both closure operators on P and Q, respectively [5].

Boolean Expressions: Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of binary-valued attributes or *items*. Let AND, OR, and NOTdenote the usual logical operators. We denote a negated item (NOT i) as $\bar{i}$. We use the symbol $|$ to denote OR, and we simply omit the AND operator whenever there is no ambiguity. For example, (i_3 AND i_4) OR (i_1 AND (NOT i_7)) is rewritten as $(i_3 i_4)|(i_1 \overline{i_7})$. A *literal* is either an item i or its negation $\neg i$. A *clause* is either the logical AND or logical OR of one or more literals. An AND-clause is a clause that has only the $\wedge$ operator over all its literals, and an OR-clause is one that has only the $\vee$ operator over all its literals. We assume without loss of generality that a clause has all distinct literals (since a clause is either an AND- or an OR-clause, repeated literals are logically redundant). A *boolean expression* is the logical AND or OR of one or more clauses.

A boolean expression is said to be in *negated normal form* (NNF) if all $\neg$ operators directly precede literals (any expression can be converted to NNF by pushing all negations into the clauses). An NNF boolean expression is said to be in *conjunctive normal form* (CNF) if it is an AND of OR-clauses. An NNF expression is said to be in *disjunctive normal form* (DNF) if it is an OR of AND-clauses. For example, $(i_3 \wedge i_4) \vee (i_1 \wedge i_5 \wedge i_7)$ is in DNF, whereas $(i_2 \vee i_3) \wedge (i_0 \vee i_1 \vee (\neg i_3))$ is in CNF. Note that by definition, single OR-clauses and single AND-clauses, are in both CNF and DNF. Furthermore, when considering negated literals, we disallow a tautology like the OR-clause containing $i|\overline{i}$ which is always true. Similarly, we disallow a contradiction like the AND-clause containing $i \wedge \overline{i}$ since this is always false. Note that a CNF expression is a tautology if and only if (iff) each one of its clauses contains both an item and its negation. Likewise, a DNF expression is a contradiction iff each one of its clauses contains both a variable and its negation. Thus by disallowing the tautologies (contradictions) in individual clauses, we disallow them in CNF (DNF) expressions, respectively.

$\mathcal{D}$

tid	set of items
1	ACD
2	BC
3	ABCD
4	ADE
5	E

$\mathcal{D}^T$

item	tidset
A	134
B	23
C	123
D	134
E	45

Figure 1: Dataset $\mathcal{D}$ and its transpose $\mathcal{D}^T$

Dataset: Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of items, let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be a set of transaction identifiers (tids). A dataset $\mathcal{D}$ is then a subset of $\mathcal{T} \times 2^{\mathcal{I}}$ (note that $2^{\mathcal{I}}$ denotes the power-set of $\mathcal{I}$, i.e., the set of all subsets of $\mathcal{I}$); in other words, the dataset $\mathcal{D}$ is a set of tuples of the form $(t, t.X)$, where $t \in \mathcal{T}$ is the tid of the transaction containing the set of items $t.X \subseteq \mathcal{I}$. Note that any categorical dataset can be transformed into this transactional form, by assigning a unique item for each attribute-value pair.

Given dataset $\mathcal{D}$, we denote by $\mathcal{D}^T$ the transposed dataset that consists of a set of tuples of the form $(i, i.Y)$, where $i \in \mathcal{I}$ and $i.Y \subseteq \mathcal{T}$ is the set of tids of transactions containing i. Fig. 1 shows a dataset and its transpose, which we will use as a running example in this paper. It has five items $I = \{A, B, C, D, E\}$ and five tids $\mathcal{T} = \{1, 2, 3, 4, 5\}$. Note that in $\mathcal{D}$, transaction t_1 contains the set of items $\{A, C, D\}$ (for convenience, we write it as ACD), and in $\mathcal{D}^T$, the set of tids of transactions that contain item A is $\{1, 3, 4\}$ (again, for convenience we write it as 134).

Tidset and Support: Given a transaction $(t, t.X) \in \mathcal{D}$, with $t \in \mathcal{T}$ and $t.X \subseteq \mathcal{I}$, we say that tid t *satisfies* an item/literal $i \in \mathcal{I}$ if $i \in t.X$, and t satisfies the literal $\overline{i}$ if $i \notin t.X$. For a literal l, the *truth value* of l in transaction t, denoted $V_t(l)$ is given as follows:

$$V_t(l) = \begin{cases} 1 & \text{if } t \text{ satisfies } l \\ 0 & \text{if } t \text{ does not satisfy } l \end{cases}$$

We say that a transaction $t \in \mathcal{T}$ *satisfies* a boolean expression E if the truth-value of E, denoted $V_t(E)$, evaluates to true when we replace every literal l in E with $V_t(l)$. For any boolean expression E, $\mathbf{t}(E) = \{t \in \mathcal{T} : V_t(E) = 1\}$ denotes the set of tids (also called a *tidset*), that satisfy E.

The *support* of a boolean expression E in dataset $\mathcal{D}$ is the number of transactions which satisfy E, i.e., $|\mathbf{t}(E)|$. An expression is *frequent* if its support is more than or equal to a user-specified *minimum support* (*min_sup*) value, i.e., if $|\mathbf{t}(E)| \geq$ *min_sup*. For disjunctive expressions, we also impose a *maximum support* threshold (*max_sup*) to disallow any expression with too high a support. Setting *min_sup* $=$ 1 and *max_sup* $= \infty$ allows mining all possible expressions.

Boolean Expression Mining Tasks: Given a dataset $\mathcal{D}$ and support thresholds *min_sup* and *max_sup*, the task is to mine minimal and closed frequent boolean expressions, such as AND-clauses, OR-clauses, CNF and DNF.

Before presenting the BLOSOM framework, we first study the structure and properties of four classes of boolean expressions; we consider each case separately – AND-clauses, OR-clauses, CNF and DNF. For simplicity of exposition, we restrict our examples to only positive literals, but our approach is applicable to negated literals as well.

3. MINING CLAUSES

3.1 Mining OR-clauses

Given dataset $\mathcal{D}$, and thresholds min_sup and max_sup, the goal here is to mine OR-clauses that occur in at least min_sup and in at most max_sup transactions. Let $\mathcal{E}^\vee$ be the set of all possible OR-clauses over the set of items $\mathcal{I}$; $\mathcal{T}$ is the set of all tids as before. For an OR-clause $X \in \mathcal{E}^\vee$, let $\mathcal{L}(X) = \{l | l \text{ is a literal in } X\}$ denote the set of its literals. Given $X, Y \in \mathcal{E}^\vee$, we define the relation $\subseteq$ between OR-clauses as follows: $X \subseteq Y$ iff $\mathcal{L}(X) \subseteq \mathcal{L}(Y)$. Then $\subseteq$ induces a partial order over $\mathcal{E}^\vee$. For example, $A|C \subseteq A|B|C$, since $\mathcal{L}(A|C) \subseteq \mathcal{L}(A|B|C)$. Let $X \in \mathcal{E}^\vee$ be an OR-clause, and let $l \in X$ be some literal in X. Then $\mathbf{t}(X) = \bigcup_{l \in X} \mathbf{t}(l)$. For example, $\mathbf{t}(A|B) = \mathbf{t}(A) \cup \mathbf{t}(B) = 134 \cup 23 = 1234$.

Closed OR-Clauses: For convenience let the OR-clause $l_1|l_2|\cdots|l_k$ be represented as $\bigvee\{l_1 l_2 \cdots l_k\}$. Let $(\mathcal{E}^\vee, \subseteq)$ be the partial order over OR-clauses, and let $(2^{\mathcal{T}}, \subseteq)$ be the partial order over the tidsets under the usual subset ($\subseteq$) relationship.

THEOREM 1. *Given posets $(\mathcal{E}^\vee, \subseteq)$ and $(2^{\mathcal{T}}, \subseteq)$. Let $X \in \mathcal{E}^\vee$ and $Y \in 2^{\mathcal{T}}$. Then the following two mappings form a monotone Galois connection over $\mathcal{E}^\vee$ and $2^{\mathcal{T}}$:*

$$\phi = \mathbf{t} : \mathcal{E}^\vee \mapsto 2^{\mathcal{T}}, \quad \mathbf{t}(X) = \{t \in \mathcal{T} \mid \ t \text{ satisfies } X\}$$
$$\psi = \mathbf{i} : 2^{\mathcal{T}} \mapsto \mathcal{E}^\vee, \quad \mathbf{i}(Y) = \bigvee\{i \in \mathcal{I} \mid \ \mathbf{t}(i) \subseteq Y\} \ \blacksquare$$

Since $(\mathbf{t}, \mathbf{i})$ forms a monotone Galois connection, it follows immediately that $\mathbb{C} = \mathbf{i} \circ \mathbf{t} : \mathcal{E}^\vee \to \mathcal{E}^\vee$ is a closure operator and $\mathbb{K} = \mathbf{t} \circ \mathbf{i} : 2^{\mathcal{T}} \to 2^{\mathcal{T}}$ is a kernel operator for OR-clauses (see Sec. 2). For example, in our example dataset from Fig. 1, $\mathbb{C}(A|C) = \mathbf{i}(\mathbf{t}(A|C)) = \mathbf{i}(1234) = A|B|C|D$. Thus $A|B|C|D$ is a closed OR-clause. On the other hand $\mathbb{K}(234) = \mathbf{t}(\mathbf{i}(234)) = \mathbf{t}(B) = 23$. Thus 23 is an open tidset. It is also easy to see that the corresponding tidset for a closed OR-clause is always open. For example, the closed OR-clause $A|D$ has the open tidset 134. We use the notation $\mathbb{C}(\mathcal{E}^\vee)$ to denote the set of all closed OR-clauses.

Minimal OR-Clauses: Given any set $\mathbf{X}$ of subsets over a universe U, we denote by $\min_\subseteq(\mathbf{X})$ the set of all minimal members in $\mathbf{X}$, w.r.t. the subset operator $\subseteq$. Let $X \in \mathcal{E}^\vee$ be a closed OR-clause. We say that $Y \subseteq X, Y \neq \emptyset$ is a *generator* of X if $\mathbb{C}(Y) = X$. Y is called a *proper* generator if $Y \neq X$. By definition, a proper generator cannot be closed. Let $\mathcal{G}(X)$ be the set of all generators of X, including X. Then $\mathcal{M}(X) = \min_\subseteq\{Y \in \mathcal{G}(X)\}$, the set of all the minimal elements of $\mathcal{G}(X)$ are called the *minimal generators* of X. The unique maximal element of $\mathcal{G}(X)$ is X.

LEMMA 2. *Let $X = \mathbb{C}(X)$. If Y is a generator of X then* $\mathbf{t}(Y) = \mathbf{t}(X)$. ∎

By Lemma 2, $\mathbf{t}(Y) = \mathbf{t}(X) = T$ for all generators $Y \in \mathcal{G}(X)$. We conclude that X is the *unique* maximal OR-clause that describes the set of objects T. On the other hand, a minimal generator of X is the minimal or simplest OR-clause that still describes the same object set T. The set of all minimal generators of closed OR-clauses in $\mathcal{E}^\vee$ is given as $\mathcal{M}(\mathcal{E}^\vee) = \bigcup_{X\in\mathbb{C}(\mathcal{E}^\vee)} \mathcal{M}(X)$. The minimal generators and closed OR-clauses, and their corresponding tidsets, for our example database (from Fig. 1), are summarized in Table 1.

Tidset	Closed	Min Generators
23	B	B
45	E	E
123	$B\|C$	C
134	$A\|D$	A, D
1234	$A\|B\|C\|D$	$A\|B, A\|C, B\|D, C\|D$
1345	$A\|D\|E$	$A\|E, D\|E$
2345	$B\|E$	$B\|E$
12345	$A\|B\|C\|D\|E$	$A\|B\|E, B\|D\|E, C\|E$

Table 1: Closed (CO) and Minimal (MO) OR-Clauses

It should be apparent at this point that both closed OR-clauses (CO) and minimal OR-clauses (MO), individually, serve as a lossless representation of the set of all possible (frequent) OR-clauses. We are particularly interested in minimal OR-clauses, since they tend to be simpler to comprehend. We would like to gain further insight into the structure of these minimal generators.

We give two separate characterizations of the minimal generators, one based on the closed clauses and the other a direct one. They are both based on the notion of hitting sets. Let $\mathbf{X} = \{X_1, X_2, \cdots, X_k\}$ be a set of subsets over some universe U. The set $Z \subseteq U$ is called a *hitting set* (or *transversal*) of $\mathbf{X}$ iff $Z \cap X_i \neq \emptyset$ for all $i \in [1, k]$. Let $\mathcal{H}(\mathbf{X})$ denote the set of all hitting sets of $\mathbf{X}$. Z is called a *minimal hitting set* if there does not exist another hitting set Z', such that $Z' \subset Z$. Let X be a closed OR-clause, we define the *lower-shadow* of X as the set $\mathbf{X}^\ell = \{X_i\}_{i\in[1,k]}$ where for all $i \in [1, k]$, $X_i \subset X$, X_i is closed, and there doesn't exist any other closed OR-clause Y, such that $X_i \subset Y \subset X$. In other words, the lower shadow of X is the set of closed OR-clauses that are immediate subsets of X. We further define the *differential lower-shadow* of X as the set $\mathbf{X}^\delta = \{X - X_i\}_{i\in[1,k]}$, where $X_i \in \mathbf{X}^\ell$.

THEOREM 3. *Let X be a closed OR-clause, and let $\mathbf{X}^\delta$ be the differential lower shadow of X. Then $\mathcal{M}(X) = \min_\subseteq\{Z \in \mathcal{E}^\vee \mid Z \in \mathcal{H}(\mathbf{X}^\delta)\}$.* ∎

The theorem states that the minimal generators of a closed OR-clause X are exactly the minimal hitting sets of the differential lower shadow of X. For example, consider the closed clause $X = A|B|C|D|E$. From Table 1 we obtain as its lower shadow the set of closed clauses $\mathbf{X}^\ell = \{A|B|C|D, A|D|E, B|E\}$. Thus the differential lower shadow of X is given as $\mathbf{X}^\delta = \{E, B|C, A|C|D\}$. The minimal hitting sets of $\mathbf{X}^\delta$ are given as $\min_\subseteq\{\mathcal{H}(\mathbf{X}^\delta)\} = \{C|E, A|B|E, B|D|E\}$. We can see from Table 1, that the minimal hitting sets are identical to the minimal generators of $A|B|C|D|E$.

The above characterization of minimal generators relies on knowing the closed sets and their lower shadows. There is in fact a direct structural description. We define a *union tidset* to be a tidset obtained by finite unions over the set of tidsets for single items, $\{\mathbf{t}(i)|i \in \mathcal{I}\}$. Let $\mathcal{U}$ be the set of all distinct union tidsets. For a union tidset $T \in \mathcal{U}$, we define the *transaction set* of T, as the set $\mathcal{R}(T) = \{t.X \subseteq \mathcal{I}|t \in T, (t, t.X) \in \mathcal{D}\}$, i.e., the set transactions in $\mathcal{D}$ with tids $t \in T$.

THEOREM 4. *Let $T \in \mathcal{U}$ be a union tidset. Then the set $\min_\subseteq\{Z \in \mathcal{E}^\vee \mid Z \in \mathcal{H}(\mathcal{R}(T))$ and $\mathbf{t}(Z) = T\}$ is identical to the minimal generators of the closed OR-clause X which has the corresponding open tidset $T = \mathbf{t}(X)$.* ∎

The above theorem states that every distinct tidset T obtained as a union of other tidsets produces minimal generators for the closed OR-clause associated with the tidset T. For example, let $T = 1345 = \mathbf{t}(A) \cup \mathbf{t}(E)$ in our example database in Fig. 1. Then $\mathcal{R}(T) = \{ACD, ABCD, ADE, E\}$. The hitting sets Z with $\mathbf{t}(Z) = T$ and that are minimal are given as follows $\{A|E, D|E\}$. Note that $C|E$ is a minimal hitting set of $\mathcal{R}(T)$, but $\mathbf{t}(C|E) = 12345 \neq T$, thus we reject it. We can see from Table 1 that these minimal hitting sets form the minimal generators of the closed OR-clause $A|D|E$ with tidset $T = 1345$.

3.2 Mining AND-clauses

Closed AND-clauses have been well studied in data mining as closed itemsets [4], as well as in the Formal Concept Analysis as concepts [6]. The notion of minimal generators for AND-clauses has also been previously proposed in [4]. We thus focus on our novel structural insights for AND-clauses.

Let $\mathcal{E}^\wedge$ be the set of all AND-clauses over the set of items $\mathcal{I}$, Given posets $(\mathcal{E}^\wedge, \subseteq)$ and $(2^\mathcal{T}, \subseteq)$, and $X \in \mathcal{E}^\wedge, Y \in 2^\mathcal{T}$, the following two mappings form an anti-monotone Galois connection [6]:

$$\phi = \mathbf{t} : \mathcal{E}^\wedge \mapsto 2^\mathcal{T}, \quad \mathbf{t}(X) = \{t \in \mathcal{T} \mid \ t \text{ satisfies } X\}$$
$$\psi = \mathbf{i} : 2^\mathcal{T} \mapsto \mathcal{E}^\wedge, \quad \mathbf{i}(Y) = \{i \in \mathcal{I} \mid \ Y \subseteq \mathbf{t}(i)\}$$

Notice the duality, in the definition of $\mathbf{i}$ and the Galois connection, for OR and AND clauses (see Theorem 1). Since $(\mathbf{t}, \mathbf{i})$ forms an anti-monotone Galois connection, it follows that $\mathbb{C} = \mathbf{i} \circ \mathbf{t} : \mathcal{E}^\wedge \to \mathcal{E}^\wedge$, and $\mathbf{t} \circ \mathbf{i} : 2^\mathcal{T} \to 2^\mathcal{T}$ both form a closure operator for AND-clauses [6]. We use the notation $\mathbb{C}(\mathcal{E}^\wedge)$ and $\mathcal{M}(\mathcal{E}^\wedge)$ to denote the set of all closed and minimal generators of AND-clauses, respectively.

In terms of the structure of minimal generators for AND-clauses, an analog of Theorem 3, describing the minimal generators of a closed AND-clause as the minimal hitting sets of its differential lower shadow is already known [9]. We focus instead on a novel characterization of the minimal generators for AND-clauses. We define an *intersection tidset* to be a tidset obtained by finite intersections over the set of tidsets for single items, $\{\mathbf{t}(i)|i \in \mathcal{I}\}$. Let $\mathcal{M}$ be the set of all distinct intersection tidsets. As before, for an intersection tidset $T \in \mathcal{M}$, we define the *transaction set* of T, denoted $\mathcal{R}(T)$, as the set of transactions with tids $t \in T$. For example, $T = 13$ is an intersection tidset since it can be obtained as the intersection of $\mathbf{t}(A)$ and $\mathbf{t}(C)$. The transaction set of T is the set of transactions with tids 1, and 3, given as $\mathcal{R}(T) = \{ACD, ABCD\}$.

THEOREM 5. *Let $T \in \mathcal{M}$ be an intersection tidset. Then the set $\min_\subseteq\{Z \in \mathcal{E}^\wedge \mid Z \in \mathcal{H}(\mathcal{R}(T))$ and $\mathbf{t}(Z) = T\}$ is identical to the minimal generators of the closed AND-clause X which has the corresponding closed tidset $T = \mathbf{t}(X)$.* ∎

Tidset	Closed	Min Generators
3	ABCD	AB, BD
4	ADE	AE, DE
13	ACD	AC, CD
23	BC	B
45	E	E
123	C	C
134	AD	A, D

Table 2: Closed (CA) and Minimal Generators (MA) for AND-clauses

This theorem gives a novel structural description of the minimal AND-clauses. It states that every distinct tidset T obtained as a finite intersection of other tidsets produces minimal generators for some AND-clause. For example, let

$T = 13 = \mathbf{t}(A) \cap \mathbf{t}(C)$ in our example database in Fig. 1. Then $\mathcal{R}(T) = \{ACD, ABCD\}$. The hitting sets Z of $\mathcal{R}(T)$ with $\mathbf{t}(Z) = T$ and that are minimal are given as follows $\{AC, CD\}$. Note that A is a minimal hitting set, but $\mathbf{t}(A) = 134$, thus we reject it. Likewise we reject hitting sets C, D, AB, and BD. Table 2 lists the set of all closed (CA) AND-clauses, the minimal generators (MA), and the corresponding tidsets. We can see that the minimal hitting sets are identical to the minimal generators of the closed AND-clause ACD with tidset $T = 13$.

4. MINING NORMAL FORMS

4.1 Mining DNF Expressions

Let $\mathcal{E}^{\text{dnf}}$ denote the set of all boolean expression in DNF, i.e., each $X \in \mathcal{E}^{\text{dnf}}$ is an OR of AND-clauses. For convenience we denote a DNF-expression X as $X = \bigvee X_i$, where each X_i is an AND-clause. By definition $\mathcal{E}^{\wedge} \subseteq \mathcal{E}^{\text{dnf}}$. Also $\mathcal{E}^{\vee} \subseteq \mathcal{E}^{\text{dnf}}$, since an OR-clause is a DNF-expression over single literal (AND) clauses. We assume we have already computed the closed ($\mathbb{C}(\mathcal{E}^{\wedge})$) and minimal ($\mathcal{M}(\mathcal{E}^{\wedge})$) AND-clauses, and their corresponding tidsets.

Note that any DNF-expression $X = \bigvee X_i$ is equivalent (in terms of the database) to the DNF expression $X' = \bigvee \mathbb{C}(X_i)$, since any tidset that satisfies X_i must satisfy $\mathbb{C}(X_i)$ as well. Similarly X is also equivalent to the DNF expression $X'' = \bigvee \mathcal{M}(X_i)$, since $\mathbf{t}(X_i) = \mathbf{t}(\mathcal{M}(X_i))$. We say that $X = \bigvee X_i$ is a *min-DNF-expression* if for each AND-clause X_i there does not exist another X_j ($i \neq j$) such that $X_i \subseteq X_j$. Note that any DNF-expression can easily be made a min-DNF-expression by simply deleting the offending clauses. For example in the DNF-expression $(AD)|(ADE)$, we have $AD \subseteq ADE$; thus the expression is logically equivalent to its min-DNF form (AD). For any DNF-expression X, we use the notation $\min^{\text{dnf}}(X)$ to denote its min-DNF form. Given $X, Y \in \mathcal{E}^{\text{dnf}}$, with $X = \bigvee X_i$ and $Y = \bigvee Y_i$, we say that X is more general than Y, denoted $X \dot{\subseteq} Y$, if there exists a 1-1 mapping f that maps each $X_i \in X$ to $f(X_i) = Y_j \in Y$, such that $X_i \subseteq Y_j$.

Tidset	Closed (maximal min-DNF)	Min Generators
34	$(ABCD)\|(ADE)$	$(AB)\|(AE)$, $(AB)\|(DE)$, $(BD)\|(AE)$, $(BD)\|(DE)$
123	$(ACD)\|(BC)$	C
134	$(ACD)\|(ADE)$	A, D
234	$(ADE)\|(BC)$	$B\|(AE)$, $B\|(DE)$
345	$(ABCD)\|E$	$(AB)\|E$, $(BD)\|E$
1234	$(ACD)\|(ADE)\|(BC)$	$A\|B, A\|C, B\|D, C\|D$
1345	$(ACD)\|E$	$A\|E, D\|E$
2345	$(BC)\|E$	$B\|E$
12345	$(ACD)\|(BC)\|E$	$A\|B\|E, B\|D\|E, C\|E$

Table 3: Additional/changed Closed (CD) and Minimal Generators (MD) for DNF

Closed DNF: We now define a closure operator for DNF expressions. First, we consider DNF expressions consisting only of closed AND-clauses. Then if we treat each $X_i \in \mathbb{C}(\mathcal{E}^{\wedge})$ as a *composite item*, we can define two monotone mappings that form a monotone Galois connection as follows: Let $X = \bigvee X_i$ be a DNF expression, such that $X_i \in \mathbb{C}(\mathcal{E}^{\wedge})$, and let $Y \in 2^{\mathcal{T}}$. Define $\mathbf{t}(X) = \{t \in \mathcal{T} \mid t \text{ satisfies } X\}$, and $\mathbf{i}(Y) = \bigvee\{X_i \mid X_i \in \mathbb{C}(\mathcal{E}^{\wedge}) \wedge \mathbf{t}(X_i) \subseteq Y\}$. This implies that $\mathbb{C} = \mathbf{i} \circ \mathbf{t}$ is a closure operator. For example, consider each closed AND-clause in Table 2 as an "item". Consider $X = ACD|E$. $\mathbb{C}(X) = \mathbf{i}(\mathbf{t}(ACD|E)) = \mathbf{i}(1345) = ABCD|ADE|ACD|E|AD$, which is a closed DNF expression. However it is logically redundant. What we want is the maximal min-DNF expression equivalent to $\mathbb{C}(X)$, which is $ACD|BC|E$.

Minimal DNF: Let $T \in 2^{\mathcal{T}}$ be a union tidset obtained as the finite union of tidsets of closed AND-clauses. As before the transaction set of T, denoted $\mathcal{R}(T)$, as the set of transactions in $\mathcal{D}$ with tid $t \in T$. Analogous to Theorem 4 for OR-clauses, we can characterize the minimal DNF expressions as the set $\min_{\subseteq}\{Z \in \mathcal{E}^{\text{dnf}} \mid Z \in \mathcal{H}(\mathcal{R}(T)) \text{ and } \mathbf{t}(Z) = T\}$, which is the set of all minimal hitting sets of $\mathcal{R}(T)$ having the tidset T. For example, consider the union tidset $T = 34$ (which is the union of $\mathbf{t}(ABCD)$ and $\mathbf{t}(ADE)$. The minimal DNF hitting sets that hit only the tidset 34 are $AB|AE, AB|DE, BD|AE, BD|DE$. In fact minimal hitting sets can be obtained directly from minimal AND-clauses.

THEOREM 6. *Let T be a union tidset, and let X be the closed DNF-expression with $\mathbf{t}(X) = T$. Then $\mathcal{M}(X) = \min_{\subseteq}\{Z = \bigvee Z_i | Z_i \in \mathcal{M}(\mathcal{E}^{\wedge})$ and $\mathbf{t}(Z) = T\}$.* ■

For example, for $T = 34$, we see in Table 2 that $\{AB, BD\}$ are the minimal generators with tidset 3, and $\{AE, DE\}$ have tidset 4. Taking the minimal OR expressions obtained from these two sets, we get all the minimal generators having tidset $T = 34$, namely $AB|AE, AB|DE, BD|AE, BD|DE$. Table 3 shows the closed and minimal DNF expressions, in addition to those shown in Tables 1 and 2. Some entries are repeated since the closed expressions in DNF have changed. Also the new union tidsets are marked in bold.

4.2 Mining CNF Expressions

Let $\mathcal{E}^{\text{cnf}}$ denote the set of all boolean expressions in CNF, i.e., each $X \in \mathcal{E}^{\text{cnf}}$ is an AND of OR-clauses. By definition $\mathcal{E}^{\vee} \subseteq \mathcal{E}^{\text{cnf}}$ and $\mathcal{E}^{\wedge} \subseteq \mathcal{E}^{\text{cnf}}$. For convenience we denote a CNF-expression X as $X = \bigwedge X_i$, where each X_i is an OR-clause. We say that X is a *min-CNF-expression* if for each OR-clause X_i there does not exist another X_j ($i \neq j$) such that $X_i \dot{\subseteq} X_j$. For a CNF expression X, we use the notation $\min^{\text{cnf}}(X)$ to denote its min-CNF form. Let $\mathcal{E}^{\text{cnf}}$ denote the set of all min-CNF-expressions. Given $X, Y \in \mathcal{E}^{\text{cnf}}$, with $X = \bigwedge X_i$ and $Y = \bigwedge Y_i$, we say that X is more general than Y, denoted $X \dot{\subseteq} Y$, if there exists a 1-1 mapping f that maps each $X_i \in X$ to $f(X_i) = Y_j \in Y$, such that $X_i \dot{\subseteq} Y_j$. Analogously to DNF expressions, we can define the closed and minimal CNF expressions directly from the set of all closed ($\mathbb{C}(\mathcal{E}^{\vee})$) and minimal ($\mathcal{M}(\mathcal{E}^{\vee})$) OR-clauses, and their corresponding tidsets.

Let's treat each $X_i \in \mathbb{C}(\mathcal{E}^{\vee})$ as a *composite item*, we can define two anti-monotone mappings that form an anti-monotone Galois connection as follows: Let $X = \bigwedge X_i$ be a CNF expression, such that $X_i \in \mathbb{C}(\mathcal{E}^{\vee})$, and let $Y \in 2^{\mathcal{T}}$. Define $\mathbf{t}(X) = \{t \in \mathcal{T} \mid t \text{ satisfies } X\}$, and $\mathbf{i}(Y) = \bigwedge\{X_i \mid X_i \in \mathbb{C}(\mathcal{E}^{\vee}) \wedge Y \subseteq \mathbf{t}(X)\}$. This implies that $\mathbb{C} = \mathbf{i} \circ \mathbf{t}$ is a closure operator. For example, consider each closed OR-clause in Table 1 as an "item". Consider $X = (A|D)(B|C)$. $\mathbb{C}(X) = \mathbf{i}(\mathbf{t}((A|D)(B|C))) = \mathbf{i}(13) = (B|C)(A|D)(A|B|C|D)(A|D|E)(A|B|C|D|E)$, which is closed. However it is logically redundant. What we want is the maximal min-CNF expression equivalent to $\mathbb{C}(X)$, which is $(A|D|E)(B|C)$.

THEOREM 7. *Let $T \in 2^{\mathcal{T}}$ be an intersection tidset obtained as the finite intersection of tidsets of closed OR-clauses. Let X be the closed CNF-expression with $\mathbf{t}(X) = T$. Then $\mathcal{M}(X) = \min_{\subseteq}\{Z = \bigwedge Z_i | Z_i \in \mathcal{M}(\mathcal{E}^{\vee})$ and $\mathbf{t}(Z) = T\}$.* ■

For example, let $T = 13$. We can obtain 13 as the intersection of several minimal OR-clauses' tidsets, e.g., the minimal OR-clauses C and $\{A|E, D|E\}$. However the minimal among all of these are C and $\{A, D\}$, giving the two minimal CNF expressions: AC and CD. Table 4 shows the closed CNF expressions and their minimal generators in addition to those already shown in Tables 1 and 2, or those that have changed.

Tidset	Closed (maximal min-CNF)	Min Generators
3	$B(A\mid D)$	AB, BD
13	$(B\mid C)(A\mid D\mid E)$	AC, CD
34	$(A\mid D)(B\mid E)$	$A(B\mid E), D(B\mid E)$
234	$(A\mid B\mid C\mid D)(B\mid E)$	$(A\mid B)(B\mid E)$, $(A\mid C)(B\mid E)$, $(B\mid D)(B\mid E)$, $(C\mid D)(B\mid E)$
345	$(A\mid D\mid E)(B\mid E)$	$(A\mid E)(B\mid E)$, $(D\mid E)(B\mid E)$

Table 4: Additional/changed Closed (CC) and Minimal Generators (MC) for CNF

5. THE BLOSOM FRAMEWORK

The BLOSOM framework supports the mining of arbitrary boolean expressions, including closed/minimal clauses (OR/AND) and normal forms (CNF/DNF). BLOSOM assumes that the input dataset is $\mathcal{D}$, and it then transforms it to work with the transposed dataset $\mathcal{D}^T$. Starting with the single items (literals) and their tidsets, BLOSOM performs a depth-first search (DFS) extending an existing expression by one more "item". BLOSOM employs a number of effective pruning techniques for searching over the space of boolean expressions, yielding orders of magnitude in speedup. These include: dynamic sibling reordering, parent-child pruning, sibling merging, threshold pruning, and fast subsumption checking. Further BLOSOM utilizes a novel *extraset* data structure for fast frequency computations, and to identify the corresponding transaction set for a given arbitrary boolean expression. Due to space limitations, we are not able to give a detailed account of these pruning methods; a very brief account follows.

5.1 Mining OR-Clauses

BLOSOM-MO mines all the minimal OR-generators, and is broadly based on Charm [15]. However, Charm mines only the closed AND-clauses, whereas BLOSOM-MO mines the minimal OR-clauses. BLOSOM-MO takes as input the set of parameter values *min_sup*, *max_sup*, *max_item* and a dataset $\mathcal{D}$ (we implicitly convert it to $\mathcal{D}^T$). The *max_item* constraint is used to limit the maximum size of any boolean expression, if desired. BLOSOM-MO conceptually utilizes a DFS tree to search over the OR-clauses. Each OR-clause is stored as a set of items (the OR is implicitly assumed). Thus each node of the search tree is a pair of $(I \times T)$, where I is an item set denoting an OR-clause and T is a tidset. BLOSOM-MO systematically searches over this DFS tree, enumerating all the frequent minimal and closed OR-clauses. The main difference in mining closed OR-clauses is that instead of finding the *minimal* elements, we have to find the *maximal* elements corresponding to the given tidsets. Thus in BLOSOM-CO the logic of subsumption checking, as well relationship pruning, has to be reversed.

5.2 Mining Other Expressions

To mine the minimal and closed AND-clauses, we build upon BLOSOM-MO and BLOSOM-CO, respectively. Note that by DeMorgan's law, to mine the minimal AND-clauses, we can mine the minimal OR-clauses over the complemented tidsets. For example in our example dataset in Fig. 1, with $\mathcal{T} = 12345$, we have $\mathbf{t}(AB) = \mathbf{t}(A) \cap \mathbf{t}(B) = 134 \cap 23 = 3$. We can obtain the same results if we $\overline{\text{mine for } \overline{A}|\overline{B} \text{ in the complemented database}}$. For example $\overline{\mathbf{t}(\overline{A})|\mathbf{t}(\overline{B})} = \overline{\mathbf{t}(\overline{A}) \cup \mathbf{t}(\overline{B})} = \overline{25 \cup 145} = \overline{1245} = 3$.

Following the structural characterization of minimal DNF expressions in Sec. 4.1, BLOSOM-MD follows a two-phase approach. It first extracts all the minimal AND-clauses and then find the minimal DNF expressions using those. First we use BLOSOM-MA to get all MA generators ($\mathcal{M}_{MA}$) on the original dataset. Second, we generate tidsets for each entry in $\mathcal{M}_{MA}$ and assign each MA class a new item label. These then form a new dataset. Third, we call BLOSOM-MO to get all MO generators on the new dataset. Next, we combine the results from BLOSOM-MA and BLOSOM-MO to form MD candidates by replacing each item label in $\mathcal{M}_{MD}$ with the MA generators it represents. Finally, we delete the subsumed generators in $\mathcal{M}_{MD}$ to produce min-DNF forms. Finally, for mining the minimal DNF and CNF expressions, we can adopt BLOSOM-MO by using the closed expression methods BLOSOM-CO and BLOSOM-CA instead of the minimal generator methods.

6. EXPERIMENTS

All experiments were done on a Ubuntu virtual machine (over WindowsXP & VMware) with 448MB memory, and a 1.4GHz Pentium-M processor. We used both synthetic and real datasets to evaluate BLOSOM.

6.1 Performance Study

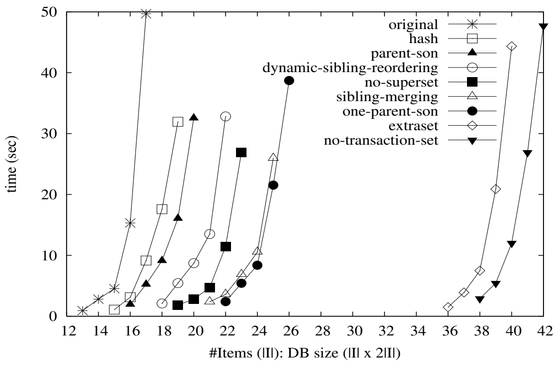

Figure 2: Speedup Optimizations

We study the effect of various optimizations proposed in on the performance of BLOSOM-MO, as shown in Fig. 2. The x-axis shows the number of items $|\mathcal{I}|$ in the synthetic datasets. The number of transactions were generated as $|\mathcal{T}| = 2|\mathcal{I}|$. Given the dataset density parameter $\delta = 0.5$, for each item i, the average size of its transaction set $\mathbf{t}(i)$, is given as $\delta \times |\mathcal{T}|$. Each curve in the figure shows the running time after applying the optimizations specified in succession. Thus the final curve for `no-transaction-set` includes all previous optimizations (`original` stands for the unoptimized version). We can see that the cumulative effect of the optimizations is substantial; BLOSOM-MO can process a dataset around 10 times $((38 \times 76)/(12 \times 24) = 10)$ larger than the base algorithm can in the same running time. Thus all the optimizations together deliver a speedup of over an order of magnitude compared to the base version.

We compared BLOSOM-MA with CHARM-L [14], which can also mine the minimal generators for AND-clauses (i.e., itemsets). We used the `chess` dataset, from the UCI machine learning repository, which has 3196 rows and 75 items. From Fig. 3 we can see that BLOSOM-MA can be about ten times faster than CHARM-L, and the gap is increasing with decreasing support. This is mainly because CHARM-L first finds all closed expressions and then uses their differential lower shadows to compute the minimal AND-clauses. In contrast, BLOSOM-MA directly mines the minimal generators, and uses effective optimizations to speed up the search.

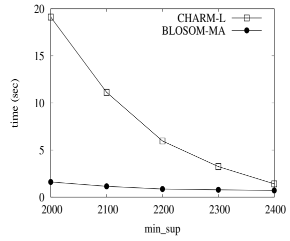

Figure 3: BLOSOM-MA **vs. CHARM-L**

6.2 Application: Gene Regulatory Networks

An application of BLOSOM is in finding complex gene regulatory networks, which can be represented in a simplified form, as boolean networks [2]. Consider the network involving 16 genes, taken from [2], shown in Fig. 4.

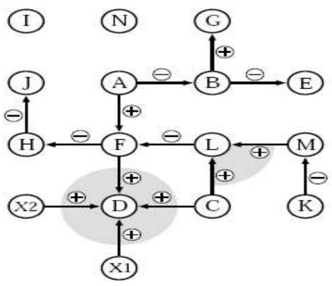

Figure 4: Gene Network

Here $\oplus$ and $\ominus$ denote gene *activation* and *deactivation*, respv. For example, genes B, E, H, J, and M are expressed if their parents are not expressed. On the other hand G, L, and D express if all of their parents express. For example, D depends on C, F, X_1 and X_2. Note that F expresses if A does, but not L. Finally A, C, I, K, N, X_1 and X_2 do not depend on anyone, and can thus be considered as *input* variables for the boolean network. We generated the truth table corresponding to the 7 input genes but BLOSOM was provided the values for all genes, without explicit instruction about which are inputs and which are outputs. This yields a dataset with 128 rows and 16 items (genes). We then ran BLOSOM to discover the boolean expression corresponding to this gene network; we used $min\_sup = 100\%$, since we want to find expressions that are true for the entire set of assignments. BLOSOM output 65 expressions in 0.36s, which hold true for the entire dataset. After simplification these can be reduced to the equivalent expression, as shown in Fig. 5. We verified that indeed this expression is true for all the rows in the dataset! It also allows us to reconstruct the boolean gene network shown in Fig. 4. For example, the first component of the expression in the first row $\overline{D} \mid (A\overline{B}CEF\overline{GH}JK\overline{LM}X_1X_2)$ can be converted into the implication $D \Rightarrow (A\overline{B}CEF\overline{GH}JK\overline{LM}X_1X_2)$, which means that D depends on the variables on the right hand side (RHS). If, at this point, we supply any partial knowledge about the input variables or of the maximum fan-out of the network, we could project the RHS only on those variables to obtain $(ACKX_1X_2)$, which happens to be precisely the relationship given in Fig. 4. The second row tells us that L depends on the activation of C and inactivation of K, i.e., $\overline{K}$, if we restrict ourselves to the input variables. Also C and $\overline{K}$ give the values for the remaining variables in the second row. Note that other dependencies in the boolean network are also included in the mined expression. For example, we find that B and A always have opposite values, and so do B and E, and K and M. G and B always have the same values, and so on. Thus this example shows the power of BLOSOM in mining gene regulatory networks.

$$\boxed{\begin{array}{l}
(\overline{D} \mid (A\ \overline{B}\ C\ E\ F\ \overline{G}\ \overline{H}\ J\ K\ \overline{L}\ \overline{M}\ X_1\ X_2))\ \text{AND} \\
(\overline{L} \mid (C\ \overline{F}\ H\ \overline{J}\ \overline{K}\ M))\ \text{AND} \\
((\overline{A}\ B\ \overline{E}\ G) \mid \overline{C} \mid D \mid L \mid \overline{X_1} \mid \overline{X_2})\ \text{AND} \\
((\overline{A}\ B\ \overline{E}\ G) \mid (C\ L) \mid (F\ \overline{H}\ J))\ \text{AND} \\
((\overline{F}\ H\ \overline{J}) \mid (A\ \overline{B}\ \overline{C}\ E\ \overline{G}) \mid (A\ \overline{B}\ E\ \overline{G}\ K\ \overline{M}))
\end{array}}$$

Figure 5: Boolean Network Expression

7. RELATED WORK

Mining frequent itemsets (i.e., pure conjunctions) has been extensively studied within the context of itemset mining [1]. The closure operator for itemsets (AND-clauses) was proposed in [6], and the notion of minimal generators for itemsets was introduced in [4]. Many algorithms for mining closed itemsets (see [7]), and a few to mine minimal generators [4, 14] have also been proposed in the past. The task of mining closed and minimal monotone DNF expressions was proposed in [12]. It gives a direct definition of the closed and minimal DNF expressions (i.e., a closed expression is one that doesn't have a superset with the same support and a minimal expression is one that doesn't have a subset with the same support). The authors further give a level-wise Apriori-style algorithm. In contrast, the novel contribution of our work is the structural characterization of the different classes of boolean expressions via the use of closure operators and minimal generators, as well as the framework for mining arbitrary expressions. Within the association rule context, there has been previous work on mining negative rules [11, 13, 3], as well as disjunctive rules [8]. Unlike these methods we are interested in characterizing such rules within the general framework of boolean expression mining. Also of relevance is the task of mining redescriptions. The CARTwheels algorithm [10] mines redescriptions only between length-limited boolean expressions in disjunctive normal form and CHARM-L [14] is restricted to redescriptions between conjunctions.

8. CONCLUSIONS

In this paper we present the first algorithm, BLOSOM, to simultaneously mine minimal generators of boolean expressions and their closures. Our four-category division of the space of boolean expressions yields a compositional approach to the mining of arbitrary expressions. The pruning operators employed here have resulted in orders of magnitude speedup, producing highly efficient implementations.

There are still many interesting issues to consider. The first one involves the effective handling of negative literals without being overwhelmed by dataset density. The second issue is to push tautological considerations deeper into the mining algorithm by designing new pruning operators. Finally, given a general propositional reasoning framework, we are interested in mining the simplest boolean expressions necessary for inference in that framework.

Acknowledgments: This work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, NSF grants EIA-0103708, EMT-0432098, EIA-0103660, IBN-0219332, and NIH/NIAID grant N01-AI-40035.

9. REFERENCES

[1] R. Agrawal, et al. Fast discovery of association rules. In *Advances in KDD*, pages 307–328. AAAI Press, 1996.
[2] T. Akutsu et al.. Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. In *Symposium on Discrete Algorithms*, 1998.
[3] M. Antonie and O. Zaiane. Mining positive and negative association rules: an approach for confined rules. In *PKDD Conf*, 2004.
[4] Y. Bastide et al. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2), Dec. 2000.
[5] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.
[6] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.
[7] B. Goethals and M. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explorations*, 6(1):109–117, June 2003.
[8] A. Nanavati et al. Association rule mining: Mining generalised disjunctive association rules. In *ACM CIKM Conf.*, 2001.
[9] J. Pfaltz and R. Jamison. Closure systems and their structure. *Information Sciences*, 139:275–286, 2001.
[10] N. Ramakrishnan et al. Turning cartwheels: An alternating algorithm for mining redescriptions. In *SIGKDD Conf.*, 2004.
[11] A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. In *IEEE ICDE Conf*, 1998.
[12] Y. Shima et al. Extracting minimal and closed monotone dnf formulas. In *Int'l Conf. on Discovery Science*, 2004.
[13] X. Wu, C. Zhang, and S. Zhang. Efficient mining of both positive and negative association rules. *ACM Trans. on Information Systems*, 22(3):381–405, 2004.
[14] M. Zaki and N. Ramakrishnan. Reasoning about sets using redescription mining. In *SIGKDD Conf.*, 2005.
[15] M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):462–478, Apr. 2005.