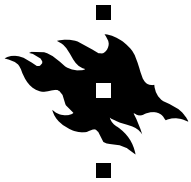
Department of Computer Science
Series of Publications C
Report C-1997-8

Levelwise search and borders of theories in knowledge discovery

Heikki Mannila and Hannu Toivonen



University of Helsinki
Finland

Levelwise search and borders of theories in knowledge discovery

Heikki Mannila and Hannu Toivonen

University of Helsinki,Department of Computer Science
PO Box 26, FIN-00014 University of Helsinki, Finland
Email: Heikki.Mannila, Hannu.Toivonen@cs.helsinki.fi
Report C-1997-8, January 1997, 26 pages

Abstract

One of the basic problems in knowledge discovery in databases (KDD) is the following: given a data set $\mathbf{r}$, a class $\mathcal{L}$ of sentences for defining subgroups of $\mathbf{r}$, and a selection predicate, find all sentences of $\mathcal{L}$ deemed interesting by the selection predicate. We analyze the simple levelwise algorithm for finding all such descriptions. We give bounds for the number of database accesses that the algorithm makes. For this, we introduce the concept of the border of a theory, a notion that turns out to be surprisingly powerful in analyzing the algorithm. We also consider the verification problem of a KDD process: given $\mathbf{r}$ and a set of sentences $\mathcal{S} \subseteq \mathcal{L}$, determine whether $\mathcal{S}$ is exactly the set of interesting statements about $\mathbf{r}$. We show strong connections between the verification problem and the hypergraph transversal problem. The verification problem arises in a natural way when using sampling to speed up the pattern discovery step in KDD.

Computing Reviews Categories and Subject Descriptors:
I.2.8 Problem Solving, Control Methods, and Search
F.2.2 Nonnumerical Algorithms and Problems
H.3.1 Content Analysis and Indexing
I.2.6 Learning

General Terms:
Algorithms, Theory

Additional Key Words and Phrases:
Knowledge Discovery, Data Mining, Borders of Theories, Association Rules, Episodes, Integrity Constraints, Hypergraph Transversals

1 Introduction

Knowledge discovery in databases (KDD), also called data mining, has recently received wide attention from practitioners and researchers. There are several attractive application areas for KDD, and it seems that techniques from machine learning, statistics, and databases can be profitably combined to obtain useful methods and systems for KDD. See, e.g., [9, 34] for general descriptions of the area.

The KDD area is and should be largely guided by (successful) applications. Still, theoretical work in the area is needed. In this paper we take some steps towards theoretical KDD. We consider a KDD process in which the analyzer first produces lots of potentially interesting rules, subgroup descriptions, patterns, etc., and then interactively selects the truly interesting ones from these. In this paper we analyze the first stage of this process: how to find all the potentially interesting rules in the database.

We study a simple breadth-first or levelwise algorithm for this task that has been studied in various forms in machine learning literature. We demonstrate how instantiations of this algorithm have been and can be used in several KDD applications. We analyze the computational complexity of this algorithm, and show that the notion of the border of interesting statements is quite important in analyzing the complexity of algorithms for this task. Our results are not technically difficult, but they show some interesting connections between KDD algorithms for various tasks.

The rest of this paper is organized as follows. The levelwise algorithm is presented in Section 2. Section 3 gives examples of the applicability of the algorithm in various KDD tasks: association rules, episode rules, and integrity constraints in relational databases. The computational complexity of the algorithm is studied in Section 4, where we also introduce the concept of border.

In Section 5 we consider using an initial guess for the collection of interesting sentences, e.g., from a sample, to speed up the discovery. This leads to

the verification problem addressed in Section 6: given a data collection and a set $\mathcal{S}$ of sentences, determine whether $\mathcal{S}$ is exactly the set of interesting sentences. We show strong connections between the verification problem and the hypergraph transversal problem. Section 7 is a short conclusion.

2 The levelwise algorithm for finding all potentially interesting sentences

Formally, the task of finding all potentially interesting sentences can be described as follows. Assume a database $\mathbf{r}$, a language $\mathcal{L}$ for expressing properties or defining subgroups of the data, and a *selection predicate* q are given. The predicate q is used for evaluating whether a sentence $\varphi \in \mathcal{L}$ defines a potentially interesting subclass of $\mathbf{r}$. The task is to find the theory of $\mathbf{r}$ with respect to $\mathcal{L}$ and q, i.e., the set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q) = \{\varphi \in \mathcal{L} \mid q(\mathbf{r}, \varphi) \text{ is true}\}$.

Example 1 Given a relation r with n rows over binary-valued attributes R, an *association rule* [1] is an expression of the form $X \Rightarrow A$, where $X \subseteq R$ and $A \in R$. Denoting $t(X) = 1$ if and only if row $t \in r$ has a 1 in each column $A \in X$, we define the *frequency* $fr(X)$ of X to be $|\{t \in r \mid t(X) = 1\}|/n$. The frequency of the rule $X \Rightarrow A$ is $fr(X \cup \{A\})$, and the confidence of the rule is $fr(X \cup \{A\})/fr(X)$.

All rules with frequency higher than a given threshold can be found effectively by using a simple algorithm for finding all *frequent* sets. A set $X \subseteq R$ is frequent, if $fr(X)$ exceeds the given threshold. Several algorithms for finding frequent sets have been presented [1, 2, 11, 14, 15, 16, 31, 35, 36, 37, 38].

The problem of finding all frequent sets can be described in our framework as follows. The description language $\mathcal{L}$ consists of all subsets X of elements of R. The selection predicate $q(\mathbf{r}, X)$ is true if and only if $fr(X) \geq min_fr$, where min_fr is the frequency threshold given by the user. □

In the above example the language $\mathcal{L}$ is a very limited slice of all potential descriptions of subsets of the original data set: we can only define subsets on

the basis of positive information.

Note that in our general framework we are not specifying any satisfaction relation for the sentences of $\mathcal{L}$ in $\mathbf{r}$: this task is taken care of by the selection predicate q. For some applications, $q(\mathbf{r}, \varphi)$ could mean that φ is true or almost true in $\mathbf{r}$, or that φ defines (in some way) an interesting subgroup of $\mathbf{r}$. The roots of this approach are in the use of *diagrams* of models in model theory (see, e.g., [5]). The approach has been used in various forms, for example in [2, 6, 7, 18, 20, 23]. One should note that in contrast with, e.g., [6], our emphasis is on very simple representation languages.

Obviously, if $\mathcal{L}$ is infinite and $q(\mathbf{r}, \varphi)$ is satisfied for infinitely many sentences, (an explicit representation of) $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ cannot be computed. For the above formulation to make sense, the language $\mathcal{L}$ has to be defined carefully.

In this paper we analyze a simple levelwise algorithm for computing the collection $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$. As already done by Mitchell [30], we use a specialization/generalization relation between sentences. (See, e.g., [22] for an overview of approaches to related problems.) A *specialization relation* is a partial order $\preceq$ on the sentences in $\mathcal{L}$. We say that φ is *more general* than θ, if $\varphi \preceq \theta$; we also say that θ is *more specific* than φ. The relation $\preceq$ is a *monotone specialization relation* with respect to q if the selection predicate q is monotone with respect to $\preceq$, i.e., for all $\mathbf{r}$ and φ we have the following: if $q(\mathbf{r}, \varphi)$ and $\gamma \preceq \varphi$, then $q(\mathbf{r}, \gamma)$. In other words, if a sentence φ satisfies q, then also all less special (i.e., more general) sentences γ satisfy q. In the sequel we assume that $\preceq$ is a monotone specialization relation. We write $\varphi \prec \theta$ for $\varphi \preceq \theta$ and not $\theta \preceq \varphi$.

Example 2 Continuing Example 1, consider two descriptions of frequent sets $\varphi = X$ and $\theta = Y$, where $X, Y \subseteq R$. Then we have $\varphi \preceq \theta$ if and only if $X \subseteq Y$. □

Typically, the relation $\preceq$ is (a restriction of) the semantic implication relation: if $\varphi \preceq \theta$, then $\theta \models \varphi$, i.e., for all databases $\mathbf{r}$, if $\mathbf{r} \models \theta$, then

$\mathbf{r} \models \varphi$. If the selection predicate q is defined in terms of statistical significance or something similar, then the semantic implication relation is not a monotone specialization relation with respect to q: a more specific statement can satisfy q, even when a general statement does not.

Consider a set $\mathcal{L}$ of sentences and a selection predicate q, for which there is a monotone specialization relation $\preceq$. Since q is monotone with respect to $\preceq$, we know that if any sentence γ more general than φ is not interesting, then φ cannot be interesting. One can base a simple but powerful generate-and-test algorithm on this idea. The central idea is to start from the most general sentences, and then to generate and evaluate more and more special sentences, but *not* to evaluate those sentences that cannot be interesting given all the information obtained in earlier iterations [2]; see also [22, 30].

The method is as follows.

Algorithm 3 The *levelwise algorithm* for finding all potentially interesting sentences.
Input: A database $\mathbf{r}$, a language $\mathcal{L}$ with specialization relation $\preceq$, and a selection predicate q.
Output: The set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$.
Method:

1. $\mathcal{C}_1 := \{\varphi \in \mathcal{L} \mid \text{there is no } \gamma \text{ in } \mathcal{L} \text{ such that } \gamma \prec \varphi\}$;
2. $i := 1$;
3. **while** $\mathcal{C}_i \neq \emptyset$ **do**
   // evaluation: find which sentences of $\mathcal{C}_i$ satisfy q:
4. $\quad \mathcal{F}_i := \{\varphi \in \mathcal{C}_i \mid q(\mathbf{r}, \varphi)\}$;
   // generation: compute $\mathcal{C}_{i+1} \subset \mathcal{L}$ using $\bigcup_{j \leq i} \mathcal{F}_j$:
5. $\quad \mathcal{C}_{i+1} := \{\varphi \in \mathcal{L} \mid \text{for all } \gamma \prec \varphi \text{ we have } \gamma \in \bigcup_{j \leq i} \mathcal{F}_j\} \setminus \bigcup_{j \leq i} \mathcal{C}_j$;
6. $\quad i := i + 1$;
7. **od**;
8. output $\bigcup_{j < i} \mathcal{F}_j$;

The algorithm works iteratively, alternating between *candidate generation* and *evaluation* phases. First, in the generation phase of an iteration i, a collection $\mathcal{C}_i$ of new *candidate* sentences is generated, using the information available from more general sentences. Then the selection predicate is evaluated on these candidate sentences. The collection $\mathcal{F}_i$ will consist of those sentences in $\mathcal{C}_i$ that satisfy q. In the next iteration $i+1$, candidate sentences in $\mathcal{C}_{i+1}$ are generated using the information about the sentences in $\bigcup_{j \leq i} \mathcal{F}_j$. The algorithm starts by constructing $\mathcal{C}_1$ to contain all the most general sentences, i.e., the sentences φ such that there is no sentence γ with $\gamma \prec \varphi$. The iteration stops when no more potentially interesting sentences can be found. In the end the potentially interesting sentences are output.

The algorithm aims at minimizing the amount of database processing, i.e., the number of evaluations of q (Step 4). Note that the computation to determine the candidate collection does not involve the database (Step 5). For example, in computations of frequent sets for association rules Step 5 uses only a negligible amount of time [2].

The following lemma is immediate.

Lemma 4 Assume $\mathcal{L}$ is a language, $\mathbf{r}$ is a database, q is a selection predicate, and $\preceq$ is a specialization relation. Then Algorithm 3 computes $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ correctly. □

For Algorithm 3 to be applicable, several conditions have to be fulfilled. The language $\mathcal{L}$ and the selection predicate have to be such that the size of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ is not too big. Recall that all sentences in $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ are probably not interesting to the user: $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ will be pruned further using, e.g., statistical significance or other criteria [19]. But $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ should not contain hundreds of thousands of useless sentences.

3 Examples

Next we look at the applicability of the algorithm by considering some examples of KDD problems.

Example 5 For association rules, the specialization relation was already given above. The algorithm will perform at most $k + 1$ iterations of the outermost loop, i.e., read the database $k + 1$ times, where k is the size of the largest subset X such that $fr(X)$ exceeds the given threshold. See [2, 14, 15, 31, 36, 38] for various implementation methods. □

Example 6 *Strong rules* [33] are database rules of the form `if` *expression* `then` *expression*, where the expressions are simple conditions on attribute values. For instance, the rule

$$\texttt{if } A < 40 \texttt{ then } B = 1$$

holds in a relation r, if on every row $t \in r$ the value of attribute B is 1 whenever the value of attribute A on t is less than 40. Such rules can be found using the above algorithm. Several choices of the specialization relation are possible, and the number of iterations in the outermost loop of the algorithm depends on that choice. □

Example 7 Consider the discovery of all inclusion dependencies that hold in a given database instance [17, 21, 24]. Given a database schema $\mathbf{R}$, an *inclusion dependency* over $\mathbf{R}$ is an expression $R[X] \subseteq S[Y]$, where R and S are relation schemas of $\mathbf{R}$, and X and Y are equal-length sequences of attributes of R and S, respectively, that do not contain duplicates.

Suppose $\mathbf{r}$ is a database over $\mathbf{R}$, and let r and s be the relations corresponding to R and S, respectively. Consider the inclusion dependency $R[X] \subseteq S[Y]$, where $X = \langle A_1, \ldots, A_n \rangle$ and $Y = \langle B_1, \ldots, B_n \rangle$. The inclusion dependency *holds* in $\mathbf{r}$ if for every tuple $t \in r$ there exists a tuple $t' \in s$ such that $t[A_i] = t'[B_i]$ for $1 \leq i \leq n$. An inclusion dependency $R[X] \subseteq S[Y]$ is *trivial*, if $R = S$ and $X = Y$.

The problem of finding inclusion dependencies is the following. Given a database schema $\mathbf{R}$ and a database $\mathbf{r}$ over $\mathbf{R}$, find all nontrivial inclusion dependencies that hold in $\mathbf{r}$. Thus, the language $\mathcal{L}$ consists of all nontrivial

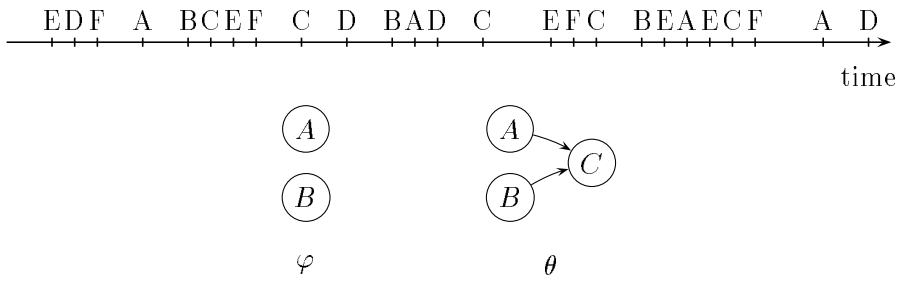


Figure 1: An event sequence and two episodes.

inclusion dependencies, and the selection predicate q is simply the satisfaction predicate. We could allow for small inconsistencies in the database by defining $q(\mathbf{r}, R[X] \subseteq S[Y])$ to be true if and only if for at least a fraction of c of the rows of r there exists a row of s with the desired properties.

This KDD task can be solved by using the levelwise algorithm. As the specialization relation we use the following: for $\varphi = R[X] \subseteq S[Y]$ and $\theta = R'[X'] \subseteq S'[Y']$, we have $\varphi \preceq \theta$ only if $R = R'$, $S = S'$, and furthermore $X' = (A_1, \ldots, A_k), Y' = (B_1, \ldots, B_k)$, and for some disjoint $i_1, \ldots, i_h \in \{1, \ldots, k\}$ with $h < k$ we have $X = (A_{i_1}, \ldots, A_{i_h}), Y = (B_{i_1}, \ldots, B_{i_h})$.

The number of iterations in the outermost loop in the algorithm will then be at most one plus the number of attributes in the attribute list of the longest nontrivial inclusion dependency that holds in the database. □

Example 8 Consider the problem of recognizing frequent episodes in sequences of events [27, 28]. An *episode* is a collection of events that occur within a time interval of a given size in a given partial order. Once such episodes are known, one can also produce rules for describing or predicting the behavior of the sequence.

Formally, an episode $\varphi = (V, \leq, g)$ is a set of nodes V, a partial order $\leq$ on V, and a mapping $g : V \to E$ associating each node with an event type in E. The interpretation of an episode is that the events in $g(V)$ have to occur in the order described by $\leq$. Figure 1 depicts a sequence of events and two episodes that occur several times in the sequence. Episode φ contains

two events, A and B, but does not specify any order for them. Episode θ contains additionally an event C, and states that C must occur after both A and B.

The task is to discover all episodes whose frequency exceeds a given threshold *min_fr*. Given a window width *win*, the frequency of φ in a given event sequence S is defined as the fraction of windows of length *win* on S that contain an instance of φ.

The discovery of all frequent episodes can be solved with the levelwise algorithm. For the specialization relation $\preceq$ we can use the following: for $\varphi = (V, \leq, g)$ and $\theta = (V', \leq', g')$ we have $\varphi \preceq \theta$, if and only if (1) $V \subseteq V'$, (2) for all $v \in V$, $g(v) = g'(v)$, and (3) for all $v, w \in V$ with $v \leq w$ also $v \leq' w$. The relation $\varphi \preceq \theta$ holds for the episodes in Figure 1. □

The next example shows a case in which the levelwise algorithm does not work very well.

Example 9 Given a relation r over attributes R, a *functional dependency* is an expression $X \to B$, where $X \subseteq R$ and $B \in R$. Such a dependency is true in the relation r, if for all pairs of rows $t, u \in r$ we have: if t and u have the same value for all attributes in X, they have the same value for B. For various algorithms for finding such dependencies, see [3, 24, 25, 26, 32].

Functional dependencies with a fixed right-hand side B can be found using the levelwise algorithm by considering the set of sentences $\{X \mid X \subseteq R\}$, and the selection predicate q: $q(r, X)$ is true if and only if $X \to B$ holds in r. The specialization relation is then the reverse of set inclusion: for X and Y we have $X \preceq Y$ if and only if $Y \subseteq X$. Then the selection predicate is monotone with respect to $\preceq$.

In applying the levelwise algorithm we now start with the sentences with no generalizations, i.e., from the sentence R, and the number of iterations in the outermost loop is at most $1 + |R \setminus X|$, where X is the smallest set such that $X \to B$ holds in r. Note that in this case for a large R there will be many iterations, even though the answer might be representable succinctly.

Note that one can avoid this problem by shifting the focus from the (minimal) left-hand sides of true functional dependencies to the maximal left-hand sides of *false* functional dependencies, and by searching for all of those, starting from the empty set. However, even in this case it can happen that many iterations are necessary, as there can be a large set of attributes that does not derive the given target attribute. □

4 Complexity of finding all sentences satisfied by q

To estimate the efficiency of the levelwise algorithm, we introduce the following notation. Consider a set $\mathcal{S}$ of sentences from $\mathcal{L}$ such that $\mathcal{S}$ is closed downwards under the relation $\preceq$, i.e., if $\varphi \in \mathcal{S}$ and $\gamma \preceq \varphi$, then $\gamma \in \mathcal{S}$.

The *border* $\mathcal{B}d(\mathcal{S})$ of $\mathcal{S}$ consists of those sentences φ such that all generalizations of φ are in $\mathcal{S}$ and none of the specializations of φ is in $\mathcal{S}$. Those sentences φ in $\mathcal{B}d(\mathcal{S})$ that are in $\mathcal{S}$ are called the *positive border*[1] $\mathcal{B}d^+(\mathcal{S})$, and those sentences φ in $\mathcal{B}d(\mathcal{S})$ that are not in $\mathcal{S}$ are the *negative border* $\mathcal{B}d^-(\mathcal{S})$. In other words, the positive border consists of the most specific sentences in $\mathcal{S}$, and the negative border consists of the most general sentences that are not in $\mathcal{S}$: $\mathcal{B}d(\mathcal{S}) = \mathcal{B}d^+(\mathcal{S}) \cup \mathcal{B}d^-(\mathcal{S})$, where

$$\mathcal{B}d^+(\mathcal{S}) = \{\varphi \in \mathcal{S} \mid \text{for all } \theta \in \mathcal{L} \text{ with } \varphi \prec \theta, \text{ we have } \theta \notin \mathcal{S}\}$$

and

$$\mathcal{B}d^-(\mathcal{S}) = \{\varphi \in \mathcal{L} \setminus \mathcal{S} \mid \text{for all } \gamma \in \mathcal{L} \text{ with } \gamma \prec \varphi, \text{ we have } \gamma \in \mathcal{S}\}.$$

A set $\mathcal{S} \subseteq \mathcal{L}$ that is closed downwards can be described by giving just the positive or the negative border. Consider, e.g., the negative border: no pattern θ such that $\varphi \preceq \theta$ for some φ in the negative border is in $\mathcal{S}$, while all other patterns are in $\mathcal{S}$.

[1] I.e., the positive border corresponds to the set denoted by "S" in [30].

A theory $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ is always closed downwards with respect to a specialization relation, and the concept of border can be applied on the set of all patterns that satisfy q.

Example 10 Consider the discovery of frequent sets with attributes $R = \{A, \ldots, F\}$. Assume the collection $\mathcal{T}h$ of frequent sets is

$$\mathcal{T}h = \{\{A\}, \{B\}, \{C\}, \{F\}, \{A, B\}, \{A, C\}, \{A, F\}, \{C, F\}, \{A, C, F\}\}.$$

The positive border of this collection contains the maximal frequent sets, i.e.,

$$\mathcal{B}d^+(\mathcal{T}h) = \{\{A, B\}, \{A, C, F\}\}.$$

The negative border, in turn, contains sets that are not frequent, but whose all subsets are frequent, i.e., minimal non-frequent sets. The negative border is thus

$$\mathcal{B}d^-(\mathcal{T}h) = \{\{D\}, \{E\}, \{B, C\}, \{B, F\}\}.$$

□

Example 11 Consider the discovery of frequent episodes in a sequence over events $A, \ldots, D$. Assume that episodes in Figure 2 constitute the theory $\mathcal{T}h$ of frequent episodes in a given sequence.

There are only two maximally specific episodes, those presented in Figure 3. All other episodes in $\mathcal{T}h$ are included in either of these two that constitute the positive border $\mathcal{B}d^+(\mathcal{T}h)$. The negative border $\mathcal{B}d^-(\mathcal{T}h)$ is depicted in Figure 4. It consists of the maximally general episodes that are not frequent. □

Using the notation $\mathcal{B}d^-(\mathcal{S})$ Step 5 of Algorithm 3 can be written as

$$\mathcal{C}_{i+1} := \mathcal{B}d^-(\bigcup_{j \leq i} \mathcal{F}_j) \setminus \bigcup_{j \leq i} \mathcal{C}_j.$$

In other words, the candidate collection is the negative border of the interesting sentences found so far. However, those sentences of the negative border that have already been evaluated, i.e., the previous candidates, are excluded.

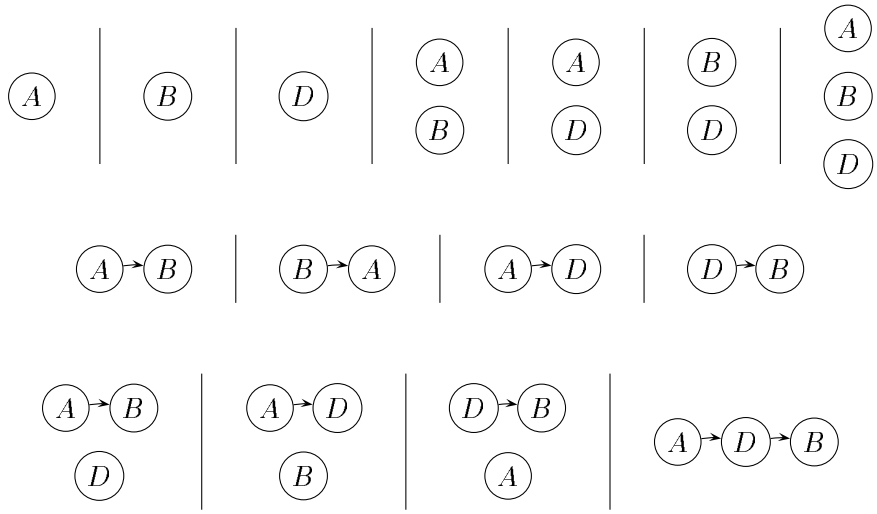


Figure 2: A collection $\mathcal{T}h$ of frequent episodes.

Figure 3: The positive border $\mathcal{B}d^+(\mathcal{T}h)$.

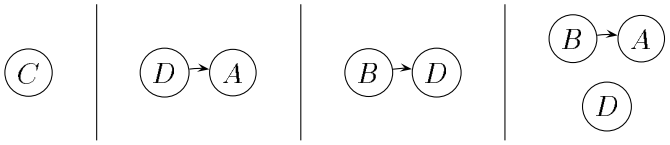


Figure 4: The negative border $\mathcal{B}d^-(\mathcal{T}h)$.

p	min_fr	$\lvert\mathcal{T}h\rvert$	$\lvert\mathcal{B}d^+(\mathcal{T}h)\rvert$	$\lvert\mathcal{B}d^-(\mathcal{T}h)\rvert$
0.2	0.01	469	273	938
0.2	0.005	1291	834	3027
0.5	0.1	1335	1125	4627
0.5	0.05	5782	4432	11531

Table 1: Experimental results with random data sets.

min_fr	$\lvert\mathcal{T}h\rvert$	$\lvert\mathcal{B}d^+(\mathcal{T}h)\rvert$	$\lvert\mathcal{B}d^-(\mathcal{T}h)\rvert$
0.08	96	35	201
0.06	270	61	271
0.04	1028	154	426
0.02	6875	328	759

Table 2: Experimental results with a real data set.

Example 12 Consider the discovery of frequent sets in a random relation over 20 attributes, where the probability that a cell has value 1 is $p = 0.2$ for all cells. In such relations the size of the border seems to be roughly 2 to 4 times the number of frequent sets. Table 1 presents the sizes of the theory and its positive and negative borders in experiments with two random relations, with $p = 0.2$ and $p = 0.5$, and with 1000 rows.

$\mathcal{B}d(\mathcal{T}h)$ can be small for a large theory $\mathcal{T}h$. An experiment with frequent sets in a real database gave the following results. We discovered all frequent sets in a course enrollment database where there is a row per student, a column per each course offered, and a row has value 1 in a column if the corresponding student took the corresponding course. There are 127 courses, and a student has taken 4.3 courses on average; the number of students is 4734. Table 2 shows that the size of the border behaves nicely with respect to the size of the theory. □

The concept $\mathcal{B}d(\mathcal{S})$ is very useful in analyzing the levelwise algorithm.

Theorem 13 Algorithm 3 uses $|\mathcal{T}h(\mathcal{L}, \mathbf{r}, q) \cup \mathcal{B}d^-(\mathcal{T}h(\mathcal{L}, \mathbf{r}, q))|$ evaluations

of the selection predicate q. □

The proof is again immediate.

Some straightforward lower bounds for the problem of finding all frequent sets are given in [2]. Now we consider the problem of lower bounds in more realistic models of computation.

The main effort in finding the theory $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ is in the evaluation of predicate q against the database. Thus we consider the following model of computation. Assume the only way of getting information from the database is by asking questions of the form

Is-interesting Is the sentence φ potentially interesting, i.e., does $q(\mathbf{r}, \varphi)$ hold?

Note that Algorithm 3 falls within this model of computation.

Theorem 14 Any algorithm that computes $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ and accesses the data using only Is-interesting queries must use at least $|\mathcal{B}d(\mathcal{T}h(\mathcal{L}, \mathbf{r}, q))|$ queries. □

We prove later that $|\mathcal{B}d(\mathcal{T}h(\mathcal{L}, \mathbf{r}, q))|$ queries are necessary already for the verification of the result.

This result gives as a corollary a result about finding functional dependencies that in the more specific setting was not easy to find [24, 25]. For simplicity, we present the result here for the case of finding keys of a relation. Given a relation r over schema R, a *key* of r is a subset X of R such that no two rows agree on X. Note that a superset of a key is always a key, and that $X \preceq Y$ if and only $Y \subseteq X$.

Corollary 15 ([25]) Given a relation r over schema R, finding the minimal keys that hold in r requires at least $|\mathrm{MAX}(r)|$ evaluations of the predicate "Is X a key", where $\mathrm{MAX}(r)$ is the set of all maximal subsets (w.r.t. set inclusion) of R that do not contain a key. □

The drawback of Theorem 14 is that the size of the border of a theory is not easy to determine. We return to this issue in Section 6, and show some connections between this problem and the hypergraph transversal problem.

Next we make some remarks about the complexity of evaluation of the selection predicate q. For finding whether a set $X \subseteq R$ is frequent, a linear pass through the database is sufficient. To verify whether an inclusion dependency $R[X] \subseteq S[Y]$ holds, one in general has to sort the relations corresponding to schemas R and S; thus the complexity is in the worst case of the order $O(n \log n)$ for relations of size n. Sorting of the relation r is also required for verifying whether a functional dependency $X \rightarrow Y$ holds in r.

The real difference between finding association rules and finding integrity constraints is, however, not the difference between linear and $O(n \log n)$ time complexities. In finding association rules one can in one pass through the database evaluate the selection predicate simultaneously on several sets, whereas to evaluate the truth of a set of dependencies requires in general as many passes through the database as there are dependencies.

5 The guess-and-correct algorithm

Algorithm 3 starts by evaluating the selection predicate q on the most general sentences, and moves gradually to more specific sentences. As the specialization relation is assumed to be monotone with respect to q, this approach is safe in the sense that no statement satisfying q will be overlooked. However, the approach can be quite slow, if there are interesting statements that are far from the bottom of the specialization hierarchy, i.e., if there are statements φ that turn out to be interesting, but which appear in the candidate set $\mathcal{C}_i$ only for a large i. As every iteration of the outermost loop requires an investigation of the database, this means that such sentences will be discovered slowly.

An alternative is to start the process of finding $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ from an initial guess $\mathcal{S} \subseteq \mathcal{L}$, and then correcting the guess by looking at the database. The

guess can be obtained, e.g., from computing the set $\mathcal{T}h(\mathcal{L}, \mathbf{s}, q)$, where $\mathbf{s} \subseteq \mathbf{r}$ is a sample of $\mathbf{r}$. The *guess-and-correct algorithm* for computing $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ is as follows.

Algorithm 16 The *guess-and-correct algorithm* for finding all potentially interesting sentences with an initial guess $\mathcal{S}$.
Input: A database $\mathbf{r}$, a language $\mathcal{L}$ with specialization relation $\preceq$, a selection predicate q, and an initial guess $\mathcal{S} \subseteq \mathcal{L}$ for $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$. We assume $\mathcal{S}$ is closed under generalizations.
Output: The set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$.
Method:

1. $\mathcal{E} := \emptyset$;
   // correct $\mathcal{S}$ downward:
2. $\mathcal{C} := \mathcal{B}d^+(\mathcal{S})$;
3. **while** $\mathcal{C} \neq \emptyset$ **do**
4. $\quad \mathcal{E} := \mathcal{E} \cup \mathcal{C}$;
5. $\quad \mathcal{S} := \mathcal{S} \setminus \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi) \text{ is false}\}$;
6. $\quad \mathcal{C} := \mathcal{B}d^+(\mathcal{S}) \setminus \mathcal{E}$;
7. **od**;
   // now $\mathcal{S} \subseteq \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$; expand $\mathcal{S}$ upwards:
8. $\mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{E}$;
9. **while** $\mathcal{C} \neq \emptyset$ **do**
10. $\quad \mathcal{E} := \mathcal{E} \cup \mathcal{C}$;
11. $\quad \mathcal{S} := \mathcal{S} \cup \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi) \text{ is true}\}$;
12. $\quad \mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{E}$;
13. **od**;
14. output $\mathcal{S}$;

The algorithm first evaluates the sentences in the positive border $\mathcal{B}d^+(\mathcal{S})$ and removes from $\mathcal{S}$ those that are not interesting. These evaluation and removal steps are repeated until the positive border only contains sentences

satisfying q, and thus we have $\mathcal{S} \subseteq \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$. The variable $\mathcal{E}$ is used to avoid evaluating sentences twice. Then the algorithm expands $\mathcal{S}$ upwards, as in the original algorithm: it evaluates such sentences in the negative border $\mathcal{B}d^-(\mathcal{S})$ that have not been evaluated yet, and adds those that satisfy q to $\mathcal{S}$. Again, the algorithm repeats the evaluation and addition steps until there are no sentences to evaluate. Finally, the output is $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$.

The following results are again straightforward, and the hence the proofs are omitted.

Lemma 17 Algorithm 16 works correctly. □

Theorem 18 Algorithm 16 uses at most

$$|(\mathcal{S} \triangle \mathcal{T}h) \cup \mathcal{B}d(\mathcal{T}h) \cup \mathcal{B}d^+(\mathcal{S} \cap \mathcal{T}h)|$$

evaluations of q, where $\mathcal{T}h = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$. □

How to obtain good original guesses $\mathcal{S}$? One fairly widely applicable method is sampling. Take a small sample $\mathbf{s}$ from $\mathbf{r}$, compute $\mathcal{T}h(\mathcal{L}, \mathbf{s}, q)$ and use it as $\mathcal{S}$. Applied to association rules this method produces extremely good results [38]. Basically, with a high probability one can discover the association rules holding in a database using only a single pass through the database.

Another method for computing an initial approximation can be derived from the algorithm of [35]. The idea is to divide $\mathbf{r}$ into small datasets $\mathbf{r}_i$ which can be handled in main memory, and to compute $\mathcal{S}_i = \mathcal{T}h(\mathcal{L}, \mathbf{r}_i, q)$. In the case of frequent sets, use as the guess $\mathcal{S}$ the union $\bigcup_i \mathcal{S}_i$; in the case of functional dependencies, use as $\mathcal{S}$ the intersection $\bigcap_i \mathcal{S}_i$. In both cases, the guess is a superset of $\mathcal{T}h(\mathcal{L}, \mathbf{r}_i, q)$, and executing the first half of the guess-and-correct algorithm suffices.

6 The verification problem

Consider the following idealized statement about the guess-and-correct method. Assume somebody gives us $\mathcal{L}$, $\mathbf{r}$, q, and a set $\mathcal{S} \subseteq \mathcal{L}$, and claims

that $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$. How many evaluations of q do we have to do to check this claim? The following result shows that the border concept is crucial also here.

Theorem 19 Given $\mathcal{L}$, $\mathbf{r}$, q, and a set $\mathcal{S} \subseteq \mathcal{L}$, determining whether $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ (1) requires in the worst case at least $|\mathcal{B}d(\mathcal{S})|$ evaluations of q, and (2) can be done using exactly this number of evaluations of q.

Proof We show that it is sufficient and in the worst case necessary to evaluate the border $\mathcal{B}d(\mathcal{S})$. The claims follow then from this.

First assume that the sentences in the border are evaluated. If and only if every sentence in $\mathcal{B}d^+(\mathcal{S})$ and no sentence in $\mathcal{B}d^-(\mathcal{S})$ is satisfied by q, then $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$, by the definition of the border. If $\mathcal{S}$ and $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ do not agree on the border, then clearly $\mathcal{S} \neq \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$.

Now assume that less than $|\mathcal{B}d(\mathcal{S})|$ evaluations have been made; then there is a sentence φ in the border $\mathcal{B}d(\mathcal{S})$ for which q has not been evaluated. If all evaluations have been consistent with the claim $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$, there is no way of knowing whether φ is in $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ or not. The satisfaction of other sentences gives no information about φ: since they were consistent with $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ and φ is in the border, all more general sentences satisfy q by the definition of border, none of the more special sentences does, and the rest are irrelevant with respect to φ. In other words, any sentence in the negative border can be swapped to the positive border, and vice versa, without changing the truth of q for any other set. □

The same argumentation can be used to prove Theorem 14.

Example 20 Continuing Example 10, consider the discovery of frequent sets in a relation over attributes $R = \{A, \ldots, F\}$. Assume a sample or some person tells us that the collection of frequent sets is

$$\mathcal{S} = \{\{A\}, \{B\}, \{C\}, \{F\}, \{A, B\}, \{A, C\}, \{A, F\}, \{C, F\}, \{A, C, F\}\}.$$

To verify whether the claim is true we have to evaluate, according to Theorem 19, the border $\mathcal{B}d(\mathcal{S})$. We thus have to evaluate $\mathcal{B}d^+(\mathcal{S}) =$

$\{\{A,B\},\{A,C,F\}\}$ and $\mathcal{B}d^-(\mathcal{S}) = \{\{D\},\{E\},\{B,C\},\{B,F\}\}$ to determine whether $\mathcal{S}$ is the collection of frequent sets. □

Example 21 Given a relation r over $\{A,B,C,D\}$, assume that $\{A,B\}$ and $\{A,C\}$ and their supersets are the only keys of r. To verify this we check the set $\mathcal{B}d(\mathcal{S})$ for

$$\mathcal{S} = \{X \subseteq \{A,B,C,D\} \mid \{A,B\} \subseteq X \text{ or } \{A,C\} \subseteq X\}.$$

(Recall that for this case $X \preceq Y$ if and only if $Y \subseteq X$.) The positive border consists of the given minimal keys, i.e.,

$$\mathcal{B}d^+(\mathcal{S}) = \{\{A,B\},\{A,C\}\},$$

and in the negative border we have sets such that all their proper supersets are keys, i.e.,

$$\mathcal{B}d^-(\mathcal{S}) = \{\{A,D\},\{B,C,D\}\}.$$

We thus have to check whether the sets $\{A,B\},\{A,C\},\{A,D\}$, and $\{B,C,D\}$ are keys of r. □

We next relate the verification problem to hypergraph transversals. For this, we need some definitions.

Let $\mathcal{L}$ be the language, $\preceq$ a specialization relation, and R a set; denote by $\mathcal{P}(R)$ the powerset of R. A function $f : \mathcal{L} \to \mathcal{P}(R)$ is a *representation of $\mathcal{L}$ (and $\preceq$) as sets*, if f is one-to-one and surjective, f and its inverse are computable, and for all φ and θ we have $\varphi \preceq \theta$ if and only if $f(\varphi) \subseteq f(\theta)$. Frequent sets, functional dependencies with a fixed right-hand sides, and inclusion dependencies are easily representable as sets; the same holds for (monotone) DNF or CNF formulae.

A collection $\mathcal{H}$ of subsets of R is a (*simple*) *hypergraph* [4], if no element of $\mathcal{H}$ is empty and if $X, Y \in \mathcal{H}$ and $X \subseteq Y$ imply $X = Y$. The elements of $\mathcal{H}$ are called the *edges* of the hypergraph, and the elements of R are the *vertices* of the hypergraph. Given a simple hypergraph $\mathcal{H}$ on R, a *transversal* T of

$\mathcal{H}$ is a subset of R intersecting all the edges of $\mathcal{H}$, that is, $T \cap E \neq \emptyset$ for all $E \in \mathcal{H}$. Transversals are also called *hitting sets*. A *minimal transversal* of $\mathcal{H}$ is a transversal T such that no $T' \subset T$ is a transversal. The collection of minimal transversals of $\mathcal{H}$ is denoted be $\mathcal{T}r(\mathcal{H})$. It is a hypergraph on R.

Now we return to the verification problem. Given $\mathcal{S} \subseteq \mathcal{L}$, we have to determine whether $\mathcal{S} = \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ holds using as few evaluations of the predicate q as possible. Given $\mathcal{S}$, we can compute $\mathcal{B}d^+(\mathcal{S})$ without looking at the data $\mathbf{r}$ at all: simply find the most special sentences in $\mathcal{S}$. The negative border $\mathcal{B}d^-(\mathcal{S})$ is also determined by $\mathcal{S}$, but finding the most general sentences in $\mathcal{L} \setminus \mathcal{S}$ can be difficult. We now show how minimal transversals can be used in the task.

Consider first the special case of frequent sets. Let the attributes in R be the vertices and the *complements* of the sets in the positive border be the edges of a simple hypergraph $\mathcal{H}$. So, for each set X in the positive border we have the set $R \setminus X$ as an edge in $\mathcal{H}$. Consider now a set $Y \subseteq R$. If there is an edge $R \setminus X$ such that $Y \cap (R \setminus X) = \emptyset$, then $Y \subseteq X$, and Y is frequent. On the other hand, if there is no such edge that the intersection is empty, then Y cannot be frequent. That is, Y is not frequent if and only if Y is a transversal of $\mathcal{H}$. Minimal transversals are now the minimal non-frequent sets, i.e., the negative border.

Generalizing, assume that (f, R) represents $\mathcal{L}$ as sets, and consider the hypergraph $\mathcal{H}(\mathcal{S})$ on R containing as edges the complements of sets $f(\varphi)$ for $\varphi \in \mathcal{B}d^+(\mathcal{S})$: $\mathcal{H}(\mathcal{S}) = \{R \setminus f(\varphi) \mid \varphi \in \mathcal{B}d^+(\mathcal{S})\}$. Then $\mathcal{T}r(\mathcal{H}(\mathcal{S}))$ is a hypergraph on R, and hence we can apply f^{-1} to it: $f^{-1}(\mathcal{T}r(\mathcal{H}(\mathcal{S}))) = \{f^{-1}(H) \mid H \in \mathcal{T}r(\mathcal{H}(\mathcal{S}))\}$. We have the following result.

Theorem 22 $f^{-1}(\mathcal{T}r(\mathcal{H}(\mathcal{S}))) = \mathcal{B}d^-(\mathcal{S})$.

Proof We prove the claim in two steps. First we show that a set $X \subseteq R$ is a transversal of $\mathcal{H}(\mathcal{S})$ if and only if $f^{-1}(X) \notin \mathcal{S}$:

$$\begin{aligned} & X \text{ is a transversal of } \mathcal{H}(\mathcal{S}) \\ \Leftrightarrow \quad & X \cap Y \neq \emptyset \text{ for all } Y \in \mathcal{H}(\mathcal{S}) \end{aligned}$$

$$
\begin{aligned}
&\Leftrightarrow \quad X \cap (R \setminus f(\varphi)) \neq \emptyset \text{ for all } \varphi \in \mathcal{B}d^+(\mathcal{S}) \\
&\Leftrightarrow \quad X \not\subseteq f(\varphi) \text{ for all } \varphi \in \mathcal{B}d^+(\mathcal{S}) \\
&\Leftrightarrow \quad f^{-1}(X) \not\preceq \varphi \text{ for all } \varphi \in \mathcal{B}d^+(\mathcal{S}) \\
&\Leftrightarrow \quad f^{-1}(X) \notin \mathcal{S}.
\end{aligned}
$$

Next we show that $\mathcal{T}r(\mathcal{H}(\mathcal{S})) = f(\mathcal{B}d^-(\mathcal{S}))$; the theorem then immediately follows.

$$
\begin{aligned}
& \mathcal{T}r(\mathcal{H}(\mathcal{S})) \\
= \quad & \{X \mid X \text{ is a minimal transversal of } \mathcal{H}(\mathcal{S})\} \\
= \quad & \{X \mid X \text{ is a minimal set such that } f^{-1}(X) \notin \mathcal{S}\} \\
= \quad & \{X \mid f^{-1}(X) \notin \mathcal{S} \text{ and } f^{-1}(Y) \in \mathcal{S} \text{ for all } Y \subset X\} \\
= \quad & \{f(\varphi) \mid \varphi \notin \mathcal{S} \text{ and } \gamma \in \mathcal{S} \text{ for all } \gamma \prec \varphi\} \\
= \quad & f(\mathcal{B}d^-(\mathcal{S})).
\end{aligned}
$$

□

Thus for languages representable as sets, the notions of negative border and the minimal transversals coincide.

Example 23 Continuing Example 20, the discovery of frequent sets in a relation with attributes $R = \{A, \ldots, F\}$, we now compute the set $\mathcal{B}d^-(\mathcal{S})$ using the hypergraph formulation. For frequent sets we simply let f be the identity function: $f(X) = X$. We are given the positive border $\mathcal{B}d^+(\mathcal{S}) = \{\{A, B\}, \{A, C, F\}\}$, so we have

$$\mathcal{H}(\mathcal{S}) = \{R \setminus f(X) \mid X \in \mathcal{B}d^+(\mathcal{S})\} = \{\{C, D, E, F\}, \{B, D, E\}\}.$$

The minimal transversals of $\mathcal{H}(\mathcal{S})$ are $\{D\}$, $\{E\}$, $\{B, C\}$, and $\{B, F\}$, thus

$$\mathcal{B}d^-(\mathcal{S}) = f^{-1}(\mathcal{T}r(\mathcal{H}(\mathcal{S}))) = \mathcal{T}r(\mathcal{H}(\mathcal{S})) = \{\{D\}, \{E\}, \{B, C\}, \{B, F\}\}.$$

□

Example 24 Recall Example 21, the problem of finding the minimal keys of a relation. We represent keys as their complements: $f(X) = R \setminus X$. Hence $\mathcal{H}(\mathcal{S}) = \{R \setminus f(X) \mid X \in \mathcal{B}d^+(\mathcal{S})\} = \mathcal{B}d^+(\mathcal{S}) = \{\{A, B\}, \{A, C\}\}$. Thus $\mathcal{T}r(\mathcal{H}(\mathcal{S})) = \{\{A\}, \{B, C\}\}$, and $f^{-1}(\mathcal{T}r(\mathcal{H}(\mathcal{S}))) = \{\{B, C, D\}, \{A, D\}\}$. □

The advantage of Theorem 22 is that there is a wealth of material known about transversals of hypergraphs (see, e.g., [4]). The relevance of transversals to computing the theory of a model has long been known in the context of finding functional dependencies [26]; see [8] for a variety of other problems where this concept turns up. The complexity of computing the transversal of a hypergraph has long been open: see [10, 13, 29] for recent breakthroughs.

7 Concluding remarks

We studied a simple levelwise algorithm for the rule discovery stage in KDD. We showed that this algorithm can be applied in various domains, including association rules, frequent episodes in sequences, and integrity constraints in relational databases. We defined the notion of the border of a set of sentences, and demonstrated how the size of the border is an important factor in the complexity of the levelwise algorithm. We also studied lower bounds for the pattern discovery task, and showed that the concept of border applies also there. We investigated the problem of computing the border, and showed that it is tightly connected to the well-known problem of computing transversals of hypergraphs.[2]

Several open problems remain. The applicability of the framework remains to be investigated for a larger class of domains. An interesting question would be to study the complexity of computing $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ as a function of the logical complexity of $\mathcal{L}$. Perhaps the most interesting question is investigating the sample-and-correct algorithm for various applications: the results with association rules indicate that this method can be very efficient.

[2] This connection has recently been strengthened [12].

Acknowledgements

Discussions with and comments from Willi Klösgen, Katarina Morik, Arno Siebes, and Inkeri Verkamo have been most useful.

Part of this work was done while Heikki Mannila was visiting Max Planck Institut für Informatik, Saarbrücken, Germany. Work of both Heikki Mannila and Hannu Toivonen is supported by the Academy of Finland.

References

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

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[3] S. Bell. Discovery and maintenance of functional dependencies by independencies. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 27 – 32, Montreal, Canada, Aug. 1995.

[4] C. Berge. *Hypergraphs. Combinatorics of Finite Sets.* North-Holland Publishing Company, Amsterdam, 3rd edition, 1973.

[5] C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, Amsterdam, 1973. 3rd ed., 1990.

[6] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the Thirteenth International Joint Conference on Artificial*

Intelligence (IJCAI–93), pages 1058 – 1053, Chambéry, France, 1993. Morgan Kaufmann.

[7] L. De Raedt and S. Džeroski. First-order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375 – 392, 1994.

[8] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278 – 1304, Dec. 1995.

[9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[10] M. L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, Nov. 1996.

[11] T. Fukuda et al. Mining optimized association rules for numeric attributes. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, 1996.

[12] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. Manuscript, 1996.

[13] V. Gurvich and L. Khachiyan. On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. Technical Report LCSR-TR-251, Rutgers University, 1995.

[14] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 420 – 431, Zurich, Swizerland, 1995.

[15] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen. A perspective on databases and data mining. In *Proceedings of the First International*

Conference on Knowledge Discovery and Data Mining (KDD'95), pages 150 – 155, Montreal, Canada, Aug. 1995.

[16] M. Houtsma and A. Swami. Set-oriented mining of association rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.

[17] M. Kantola, H. Mannila, K.-J. Räihä, and H. Siirtola. Discovering functional and inclusion dependencies in relational databases. *International Journal of Intelligent Systems*, 7(7):591 – 607, Sept. 1992.

[18] J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive Logic Programming*, pages 335 – 359. Academic Press, London, 1992.

[19] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401 – 407, Gaithersburg, MD, Nov. 1994. ACM.

[20] W. Kloesgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53 – 69, 1995.

[21] A. J. Knobbe and P. W. Adriaans. Discovering foreign key relations in relational databases. In *Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research*, pages 961 –966, Vienna, Austria, Apr. 1996.

[22] P. Langley. *Elements of Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1996.

[23] H. Mannila and K.-J. Räihä. Design by example: An application of Armstrong relations. *Journal of Computer and System Sciences*, 33(2):126 – 141, 1986.

[24] H. Mannila and K.-J. Räihä. *Design of Relational Databases.* Addison-Wesley Publishing Company, Wokingham, UK, 1992.

[25] H. Mannila and K.-J. Räihä. On the complexity of dependency inference. *Discrete Applied Mathematics*, 40:237 – 243, 1992.

[26] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies. *Data & Knowledge Engineering*, 12(1):83 – 99, Feb. 1994.

[27] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 146 – 151, Portland, Oregon, Aug. 1996. AAAI Press.

[28] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215, Montreal, Canada, Aug. 1995.

[29] N. Mishra and L. Pitt. On bounded-degree hypergraph transversal. Manuscript, 1995.

[30] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203 – 226, 1982.

[31] J. S. Park, M.-S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'95)*, pages 175 – 186, San Jose, California, May 1995.

[32] B. Pfahringer and S. Kramer. Compression-based evaluation of partial determinations. In *Proceedings of the First International Conference*

on Knowledge Discovery and Data Mining (KDD'95), pages 234 – 239, Montreal, Canada, Aug. 1995.

[33] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229 – 248. AAAI Press, Menlo Park, CA, 1991.

[34] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI Press, Menlo Park, CA, 1991.

[35] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 432 – 444, Zurich, Swizerland, 1995.

[36] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 407 – 419, Zurich, Swizerland, 1995.

[37] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'96)*, pages 1–12, Montreal, Canada, 1996.

[38] H. Toivonen. Sampling large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, pages 134 – 145, Mumbay, India, Sept. 1996. Morgan Kaufmann.