Data Mining and Knowledge Discovery 1, 241–258 (1997)

Levelwise Search and Borders of Theories in Knowledge Discovery

HEIKKI MANNILA heikki.mannila@cs.helsinki.fi
HANNU TOIVONEN hannu.toivonen@cs.helsinki.fi
Department of Computer Science, P.O. Box 26, FIN–00014 University of Helsinki, Finland

Editor: Usama Fayyad

Received January 28, 1997; Revised July 10, 1997; Accepted July 11, 1997

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.

Keywords: theory of knowledge discovery, 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., Fayyad et al. (1996), and Piatetsky-Shapiro and Frawley (1991) 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.

The intuitive idea behind this work is as follows. A lot of work in data mining can be formulated in terms of finding all rules of the form

if φ then θ,

where φ and θ are possibly quite complex conditions from a predefined and potentially large set of patterns $\mathcal{L}$. The interpretation of such a rule is that in most cases where φ applies to the data, also θ applies. Examples of cases where this formulation applies are the discovery of association rules, episode rules, and integrity constraints in relational databases.

The crucial observation is that typically in these data mining tasks the rule can only be interesting if there are sufficiently many occasions in which the rule applies: we are not interested in rules that talk about only a few cases. Then the task of finding all rules of the above form can be solved by locating all conditions from the class $\mathcal{L}$ that apply frequently enough; we call such conditions *frequent*. Once frequent conditions are known, computing the confidences of the rules is simple.

How can we, then, find all frequent conditions? Suppose that the conditions in the class $\mathcal{L}$ are built from some primitive conditions using conjunction. Then the frequency of occurrence is non-increasing with the number of conjuncts. We can therefore first search for frequent conditions with one conjunct, then build two-conjunct conditions from these, etc.; that is, we can proceed in the partial order of the conditions in a levelwise manner.

This argumentation can be generalized as follows. We want to find all rules of the form "if φ then θ", where φ and θ are from $\mathcal{L}$. Additionally, we require that the condition φ must define an interesting subgroup of the data or otherwise be relevant. We assume there is a way of determining whether φ satisfies this demand, which might depend on the frequency of the condition, as above, or on some additional factors. Assume additionally that there is a partial order $\preceq$ between conditions such that if $\theta \preceq \varphi$ and the condition φ defines an interesting subgroup, then θ also defines an interesting subgroup. If such a partial order exists, then we can simply start from the minimal elements of this order. If any conditions are frequent, then the minimal ones must be. We can then work our way up levelwise in the partial ordering relation, evaluating the frequency of a condition only if all elements below it are indeed frequent.

This simple breadth-first or levelwise algorithm has actually been studied in various forms in machine learning literature. From a machine learning viewpoint, the algorithm might be called trivial. Our emphasis is, however, on investigating the problem of finding all rules, not just a single rule with high predictive power. The algorithm can be shown to be optimal in this respect under some conditions, and it performs quite well in practice.

In this paper we give a framework for the discovery of all patterns from a given class of conditions. We demonstrate how instantiations of the levelwise algorithm have been and can be used in several KDD applications. We also analyze the computational complexity of this algorithm. Given a collection $\mathcal{S}$ of frequent conditions, the border of $\mathcal{S}$ is the set of all conditions that are "just off" $\mathcal{S}$ or "just in" $\mathcal{S}$, i.e., conditions such that all the conditions preceding them in the partial order are indeed frequent, and such that those succeeding them are not. The concept of border turns out to be useful in analyzing the levelwise algorithm, and it also has some nice connections to hypergraph transversals. We also discuss a general algorithm for the use of sampling in locating all frequent rules. 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. Given a relation r with n rows over binary-valued attributes R, an *association rule* (Agrawal et al., 1993) is an expression of the form $X \Rightarrow A$, where $X \subseteq R$ and $A \in R$. The interpretation of the rule is that those rows in r that have value 1 for the attributes in X, also tend to have value 1 for the attribute A. Formally, 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 (Agrawal et al., 1993, 1996; Fukuda et al., 1996; Han and Fu, 1995; Holsheimer et al., 1995; Houtsma and Swami, 1993; Park et al., 1995; Savasere et al., 1995; Srikant and Agrawal, 1995, 1996; Toivonen, 1996).

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.

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., Chang and Keisler, 1973). The approach has been used in various forms, for example in (Agrawal et al., 1996; De Raedt and Bruynooghe, 1993; De Raedt

and Džeroski, 1994; Kietz and Wrobel, 1992; Kloesgen, 1995; Mannila and Räihä, 1986). One should note that in contrast with, e.g., De Raedt and Bruynooghe (1993), 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 (1982), we use a specialization/generalization relation between sentences. (See, e.g., Langley (1996) 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 $\theta \preceq \varphi$. We assume that the minimal elements of $\mathcal{L}$ under $\preceq$ can be located efficiently.

Example. Continuing the previous example, consider two descriptions φ and θ of frequent sets, where $\varphi = X$ and $\theta = Y$, and $X, Y \subseteq R$. 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 possibly not a monotone specialization relation with respect to q: a more specific statement can satisfy q, even when a general statement does not. (It has been pointed out (Brin et al., 1997) that the χ^2 test statistic is monotone on binomial data.)

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 (Agrawal et al., 1996); see also Langley (1996), and Mitchell (1982). The method is given as Algorithm 1.

Algorithm 1. 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. $\mathcal{C}_1 := \{\varphi \in \mathcal{L} \mid$ there is no γ in $\mathcal{L}$ such that $\gamma \prec \varphi\}$;
2. $i := 1$;
3. *while* $\mathcal{C}_i \neq \emptyset$ *do*
4. // evaluation: find which sentences of $\mathcal{C}_i$ satisfy q:

5. $\mathcal{F}_i := \{\varphi \in \mathcal{C}_i \mid q(\mathbf{r}, \varphi)\}$;
6. // generation: compute $\mathcal{C}_{i+1} \subset \mathcal{L}$ using $\bigcup_{j \le i} \mathcal{F}_j$:
7. $\mathcal{C}_{i+1} := \left\{\varphi \in \mathcal{L} \mid \text{for all } \gamma \prec \varphi \text{ we have } \gamma \in \bigcup_{j \le i} \mathcal{F}_j\right\} \setminus \bigcup_{j \le i} \mathcal{C}_j$;
8. $i := i + 1$;
9. *od*;
10. 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 $\bigcup_{j \le i} \mathcal{F}_j$. 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 5). Note that the computation to determine the candidate collection (Step 7) does not involve the database. For example, in computations of frequent sets for association rules Step 7 uses only a negligible amount of time (Agrawal et al., 1996).

The following lemma is immediate.

Lemma 1. *Assume $\mathcal{L}$ is a language, $\mathbf{r}$ is a database, q is a selection predicate, and $\preceq$ is a specialization relation. Then Algorithm* 1 *computes $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ correctly.*

For Algorithm 1 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 (Klemettinen et al., 1994). 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. For association rules, the specialization relation was already given above. The algorithm performs at most $k+1$ iterations of the loop, i.e., it reads the database $k+1$ times, where k is the size of the largest subset X such that $fr(X)$ exceeds the given threshold. See Agrawal et al. (1996), Han and Fu (1995), Holsheimer et al. (1995), Park et al. (1995), Savasere et al. (1995), and Srikant and Agrawal (1995) for various implementation methods.

Example. *Strong rules* (Piatetsky-Shapiro, 1991) 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 depends on that choice.

Example. Consider the discovery of all inclusion dependencies that hold in a given database instance (Kantola et al., 1992; Knobbe and Adriaans, 1996; Mannila and Räihä, 1992a). 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 inclusion dependencies, and the selection predicate q is simply the satisfaction predicate. We could also 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 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. Consider the problem of discovering frequent episodes in sequences of events (Mannila et al., 1995, 1997). 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(a) depicts a sequence of events and figure 1(b) two episodes that occur several times in the sequence. Episode φ contains two events, A and B, but does not specify any order for

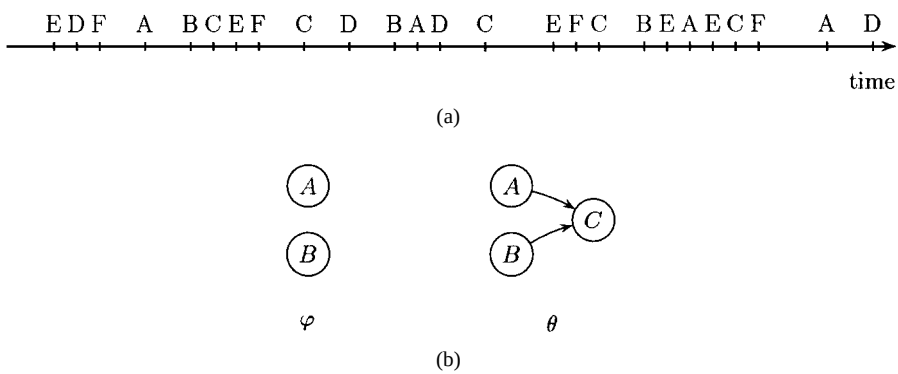

Figure 1. (a) An event sequence; (b) two episodes.

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 instance, for the episodes in figure 1(b).

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

Example. 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, then they have the same value for B. For various algorithms for finding such dependencies, see Bell (1995), Mannila and Räihä (1992a, 1992b, 1994), and Pfahringer and Kramer (1995).

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 a selection predicate q such that $q(r, X)$ 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 $Y \subseteq X$. 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 is at most $1 + |R \setminus X|$, where X is the smallest set such that $X \to B$ holds in r. In this case for a large R there will be many iterations, even though the answer might be representable succinctly.

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

Our basic task is to find the set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ of all sentences from $\mathcal{L}$ that satisfy the selection predicate q. We want to analyze the complexity of this task, concentrating especially on the number of evaluations of q. The size of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ is one indicator of the complexity of the task: one could assume that a big set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ would be more difficult to locate than a smaller one. To certain extent this is true, but it turns out that the size of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ is not the only influential factor. Also, the number of elements in the border of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ has an influence on the running time of the levelwise algorithm and also on the inherent complexity of the task.

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* $\mathcal{B}d^+(\mathcal{S})$, 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 $\mathcal{S}$ (denoted by "S" in Mitchell, 1982), 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 $\mathcal{S}$.

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. 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. Consider the class of totally ordered episodes, and denote an episode $\varphi = (V, <, g)$ by the corresponding sequence $\langle A_1, \ldots, A_{|V|}\rangle$ of events, i.e., for all x_i in V we have $g(x_i) = A_i$ and for all i, $1 \leq i < |V|$ we have $x_i < x_{i+1}$.

Assume that the 28 episodes that are subsequences of episodes

$$\{\langle C, F, A\rangle, \langle F, A, D\rangle, \langle A, D, B, C\rangle, \langle D, B, C, F\rangle\}$$

are frequent. In other words, the given episodes are the maximal frequent ones, and constitute the positive border of the collection of all frequent episodes. In the negative border we have the minimal non-frequent episodes:

$$\{\langle E\rangle, \langle A, F\rangle, \langle B, A\rangle, \langle B, D\rangle, \langle C, B\rangle, \langle C, D\rangle, \langle D, A\rangle, \langle F, B\rangle, \langle F, C\rangle\}.$$

Using the notion of negative border, step 7 of Algorithm 1 can be written as

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

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.

Example. 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. Note that the sizes of the theory and its border are determined by the nature of the data rather than by the number of rows.

The collection $\mathcal{B}d(\mathcal{T}h)$ 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

Table 1. Experimental results with random data sets.

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 2. Experimental results with a real data set.

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

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 of negative border is useful in analyzing the levelwise algorithm.

Theorem 1. *Algorithm* 1 *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. Next we consider the problem of lower bounds for the computation of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$. Some straightforward lower bounds for the problem of finding all frequent sets are given in Agrawal et al. (1996).

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 1 falls within this model of computation.

Theorem 2. *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 omit the proof, since in Theorem 4 we show 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 (Mannila and Räihä, 1992a, 1992b). 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 1 (Mannila and Räihä, 1992b). *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 2 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 $X \to B$ 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 integrity constraints requires in general as many passes through the database as there are constraints.

5. The guess-and-correct algorithm

Algorithm 1 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 $\mathbf{r}$. Algorithm 2 is the *guess-and-correct algorithm* for computing $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$.

Algorithm 2. 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. $\mathcal{E} := \emptyset$;
2. // correct $\mathcal{S}$ downward:
3. $\mathcal{C} := \mathcal{B}d^+(\mathcal{S})$;
4. *while* $\mathcal{C} \neq \emptyset$ *do*
5. $\quad \mathcal{E} := \mathcal{E} \cup \mathcal{C}$;

6. $\mathcal{S} := \mathcal{S} \setminus \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi)$ is false$\}$;
7. $\mathcal{C} := \mathcal{B}d^+(\mathcal{S}) \setminus \mathcal{E}$;
8. *od*;
9. // now $\mathcal{S} \subseteq \mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$; expand $\mathcal{S}$ upwards:
10. $\mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{E}$;
11. *while* $\mathcal{C} \neq \emptyset$ *do*
12. $\mathcal{E} := \mathcal{E} \cup \mathcal{C}$;
13. $\mathcal{S} := \mathcal{S} \cup \{\varphi \in \mathcal{C} \mid q(\mathbf{r}, \varphi)$ is true$\}$;
14. $\mathcal{C} := \mathcal{B}d^-(\mathcal{S}) \setminus \mathcal{E}$;
15. *od*;
16. 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 2. *Algorithm* 2 *works correctly.*

Theorem 3. *Algorithm* 2 *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 $\mathcal{S}$. For finding frequent sets one can show that sampling produces very good approximations (Toivonen, 1996). In Toivonen (1996) sampling is applied in finding an upper approximation $\mathcal{S}$ of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$, i.e., a collection $\mathcal{S}$ such that $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q) \subset \mathcal{S}$ and such that $\mathcal{S}$ is not unnecessarily large. Given such $\mathcal{S}$, evaluating only the first half of Algorithm 2 suffices, and that can be implemented with a single pass over the database. There is, however, a small probability that the sample is skewed and $\mathcal{S}$ is not a superset of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$; then the second half of Algorithm 2 is executed in another database pass.

Another method for computing an initial approximation can be derived from the algorithm of Savasere et al. (1995). 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 4. *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 2.

Example. 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 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* (Berge, 1973), 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 5. $f^{-1}(\mathcal{T}r(\mathcal{H}(\mathcal{S}))) = \mathcal{B}d^-(\mathcal{S})$.

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 $f^{-1}(X) \notin \mathcal{S}$:

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

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. Recall the previous example, the problem of finding the minimal keys of a relation. We now compute the set $\mathcal{B}d^-(\mathcal{S})$ using the hypergraph formulation. 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

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

The advantage of Theorem 5 is that there is a wealth of material known about transversals of hypergraphs (see, e.g., Berge, 1973). The relevance of transversals to computing the theory of a model has long been known in the context of finding functional dependencies (Mannila and Räihä, 1994); see Eiter and Gottlob (1995) 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 Fredman and Khachiyan (1996), Gurvich and Khachiyan (1995), and Mishra and Pitt (1995) for recent breakthroughs.

7. Concluding remarks

We studied a simple levelwise algorithm for the rule discovery stage in KDD. We showed that this basically trivial 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.

This connection has recently been strengthened in Gunopulos et al. (1997), where some analogous concepts from the domain of PAC learning are also studied.

The framework and algorithm we have given are quite general. Their applicability is restricted chiefly by the requirement of a monotone specialization relation: such a relation does not necessarily exist if the criteria for interestingness or relevance are dependent on statistical significance or similar factors. Even in these cases the framework can possibly be used: the selection predicate q is defined in terms of, e.g., frequency of occurrence, and the statistically significant sentences are then selected from $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ by using a separate pruning step. Obviously, the problem here is that the size of $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q)$ might be much larger than the number of actually interesting sentences. Further experimentation is needed to study this issue.

Our framework is mainly conceptual: we have shown that existing algorithms can be viewed as instances of the framework. We have not evaluated the abstract algorithm empirically on new description languages $\mathcal{L}$. This is an important topic for further study.

We would like to point out especially the following possibly widely applicable approach. Consider a class $\mathcal{P}$ of primitive patterns, and build the description language $\mathcal{L}$ by forming all conjunctions of patterns from $\mathcal{P}$. Define the selection predicate q on the basis of occurrence frequency; then a monotone specialization relation exists automatically. It seems that this method might give quite good results; combining it with the use of sampling for the guess-and-correct methods is also worth studying. The method might give an approach to the KDD query compilation problem (Imielinski and Mannila, 1996): how to compile a specification of a KDD task to an efficient program for it.

Several open problems remain also in the technical side. 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 theoretical and practical efficiency of the guess-and-correct algorithm for various applications: the results with association rules indicate that this method can be very efficient.

Acknowledgments

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

Part of this work was done while Heikki Mannila was visiting Max Planck Institut für Informatik, Saarbrücken, Germany.

This work is supported by the Academy of Finland.

References

Agrawal, R., Imielinski, T., and Swami, A. 1993. Mining association rules between sets of items in large databases. In Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD '93). Washington, D.C., pp. 207–216.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A.I. 1996. Fast discovery of association rules. In Advances in Knowledge Discovery and Data Mining, U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, (Eds.), Menlo Park, CA: AAAI Press. pp. 307–328.

Bell, S. 1995. Discovery and maintenance of functional dependencies by independencies. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD '95). Montréal, Canada, pp. 27–32.

Berge, C. 1973. Hypergraphs. Combinatorics of Finite Sets (third edition). Amsterdam: North-Holland.

Brin, S., Motwani, R., and Silverstein, S. 1997. Beyond market baskets: Generalizing association rules to correlations. In Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD '97). Tucson, AZ., pp. 265–276.

Chang, C.C. and Keisler, H.J. 1973. Model Theory (third edition, 1990). Amsterdam: North-Holland.

De Raedt, L. and Bruynooghe, M. 1993. A theory of clausal discovery. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI–93). Chambéry, France, pp. 1058–1053.

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

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

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (Eds.), 1996. Advances in Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI Press.

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

Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama, T. 1996. Mining optimized association rules for numeric attributes. In Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '96). Montréal, Canada, pp. 182–191.

Gunopulos, D., Khardon, R., Mannila, H., and Toivonen, H. 1997. Data mining, hypergraph transversals, and machine learning. In Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '97). Tucson, AZ., pp. 209–216.

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

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

Holsheimer, M., Kersten, M., Mannila, H., and Toivonen, H. 1995. A perspective on databases and data mining. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD '95). Montréal, Canada, pp. 150–155.

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

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

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

Kietz, J.-U. and Wrobel, S. 1992. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton (Ed.), Inductive Logic Programming. London: Academic Press, pp. 335–359.

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

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

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

Langley, P. 1996. Elements of Machine Learning. San Mateo, CA: Morgan Kaufmann.

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

Mannila, H. and Räihä, K.-J. 1992a. Design of Relational Databases. Wokingham, UK: Addison-Wesley.

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

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

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

Mannila, H., Toivonen, H., and Verkamo, A.I. 1997. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1(3):259–289.

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

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

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

Pfahringer, B. and Kramer, S. 1995. Compression-based evaluation of partial determinations. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD '95). Montréal, Canada, pp. 234–239.

Piatetsky-Shapiro, G. 1991. Discovery, analysis, and presentation of strong rules. In Knowledge Discovery in Databases, G. Piatetsky-Shapiro and W.J. Frawley (Eds.), Menlo Park, CA: AAAI Press, pp. 229–248.

Piatetsky-Shapiro, G. and Frawley, W.J. (Eds.), 1991. Knowledge Discovery in Databases. Menlo Park, CA: AAAI Press.

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

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

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

Toivonen, H. 1996. Sampling large databases for association rules. In Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB '96). Mumbay, India, pp. 134–145.

Heikki Mannila is a professor of Computer Science at the University of Helsinki, where he also obtained his Ph.D. in 1985. After that he has been an associate professor at the Universities of Tampere and Helsinki, a visiting professor at the Technical University of Vienna, and a guest researcher at the Max Planck Institute für Informatik in Saarbrücken. He has also worked at the National Public Health Institution in Helsinki, as well as a consultant in industry. His research interests include rule discovery from large databases, the use of Markov chain Monte Carlo techniques in data analysis, and the theory of data mining. He is one of the program chairmen of KDD-97.

Hannu Toivonen is an assistant professor at the University of Helsinki, Finland. Prior to joining the university, he was a research engineer at Nokia Research Center, where he was involved with knowledge-based systems and methods for telecommunication network management. Hannu Toivonen earned his Ph.D. in Computer Science from the University of Helsinki in 1996 on data mining, with a thesis titled "Discovery of frequent patterns in large data collections." He is one of the developers of the TASA knowledge discovery system and the implementor of the data mining algorithms. His current research interests are in data mining and in the use of Markov chain Monte Carlo methods for data analysis.