On an algorithm for finding all interesting sentences
Extended abstract

Heikki Mannila*
MPI Informatik
Im Stadtwaldt
D-66123 Saarbrücken, Germany
mannila@mpi-sb.mpg.de

Hannu Toivonen†
University of Helsinki
Department of Computer Science
FIN-00014 Helsinki, Finland
Hannu.Toivonen@Helsinki.FI

Abstract

Knowledge discovery in databases (KDD), also called data mining, has recently received wide attention from practitioners and researchers. One of the basic problems in KDD is the following: given a data set $\mathbf{r}$, a class $\mathcal{L}$ of sentences defining subgroups or properties of $\mathbf{r}$, and an interestingness predicate, find all sentences of $\mathcal{L}$ deemed interesting by the interestingness predicate. In this paper we analyze a simple and well-known levelwise algorithm for finding all such descriptions. We give bounds for the number of database accesses that the algorithm makes. We also consider the verification problem of a KDD process: given $\mathbf{r}$ and a set of sentences $T \subseteq \mathcal{L}$, determine whether T 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.

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; 27] for general descriptions of the area.

The KDD area is and should be largely guided by (succesful) applications. n this paper we take some steps towards theoretical KDD. Namely, we present a simple framework for KDD in which the task of knowledge discovery is defined to be finding all interesting statements from a set of sentences. We study a simple breath-first or levelwise algorithm for this task that has been used in various forms in machine learning and in several KDD appications, and we analyze the properties, especially the computational complexity, of this algorithm. Our results are not technically difficult, but they show some interesting connections between KDD algorithms for various tasks.

The model of knowledge discovery that we consider is the following. Given a database $\mathbf{r}$, a language $\mathcal{L}$ for expressing properties or defining subgroups of the data, and an interestingness predicate q for evaluating whether a sentence $\varphi \in \mathcal{L}$ defines an 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 $Th(\mathcal{L}, \mathbf{r}, q) = \{\varphi \in \mathcal{L} \mid q(\mathbf{r}, \varphi) \text{ is true}\}$.

Note that we are not specifying any satisfaction relation for the sentences of $\mathcal{L}$ in $\mathbf{r}$: this task is taken care of by the interestingness 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; 13; 15; 18] 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) $Th(\mathcal{L}, \mathbf{r}, q)$ cannot be computed. For the above formulation to make sense, the language $\mathcal{L}$ has to be defined carefully.

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 $t(X) = 1$ iff row $t \in r$ has a 1 in each column $A \in X$, we define the support $s(X)$ of X to be $|\{t \in r \mid t(X) = 1\}|/n$. The support of the rule $X \Rightarrow A$ is $s(X \cup \{A\})$, and the confidence of the rule is $s(X \cup \{A\})/s(X)$.

All rules with support higher than a given threshold can be effectively found by using a simple algorithm for finding *frequent* sets. A set $X \subseteq R$ is frequent, if

*On leave from the Department of Computer Science, University of Helsinki. Work supported by the Academy of Finland and the Alexander von Humboldt Stiftung.

†Work supported by the Academy of Finland.

$s(X)$ exceeds the given threshold; for variations of the algorithm, see [2] and references there.

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 interestingness predicate $q(\mathbf{r}, X)$ is true if and only if $s(X) > min\_s$, where min_s is a 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. The crucial property of $\mathcal{L}$ is that all frequent sets are interesting: even the most general descriptions in the language (i.e., the descriptions $\{A\}$ for attributes $A \in R$) are interesting.

In this paper we analyze a simple algorithm for computing the collection $Th(\mathcal{L}, \mathbf{r}, q)$. The algorithm is presented in Section 2. Section 3 gives examples of the applicability of the algorithm in various KDD tasks. The computational complexity of the algorithm is studied in Section 4.

In Section 5 we consider the use of sampling to speed up the discovery. This leads to the verification problem addressed in Section 6: given $\mathbf{r}$ and a set of sentences $T \subseteq \mathcal{L}$, determine whether T is exactly the set of interesting statements about $\mathbf{r}$. We show strong connections between the verification problem and the hypergraph transversal problem.

2 The algorithm

In this section we present the simple algorithm for finding all interesting statements. As already considered by Mitchell [24], we use a specialization/generalization relation between sentences. (See, e.g., [17] 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 quality function 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 $\varphi' \preceq \varphi$, then $q(\mathbf{r}, \varphi')$. In other words, if a sentence φ is interesting according to the quality function q, then also all less special (i.e., more general) sentences $\varphi' \preceq \varphi$ are interesting. We write $\sigma \prec \tau$ if $\sigma \preceq \tau$ and not $\tau \preceq \sigma$.

Typically, the relation $\preceq$ is (a restriction of) the semantic implication relation: if $\sigma \preceq \tau$, then $\tau \models \sigma$, i.e., for all databases $\mathbf{r}$, if $\mathbf{r} \models \tau$, then $\mathbf{r} \models \sigma$. Note that if the interestingness 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 be interesting, even when a general statement is not.

Example 2 Consider two descriptions of frequent sets $\sigma = X$ and $\tau = Y$, where $X, Y \subseteq R$. Then we have $\sigma \preceq \tau$ if and only if $X \subseteq Y$. □

Consider a set $\mathcal{L}$ of sentences, and a quality function q, for which there is a monotone specialization relation $\preceq$. Since q is monotone with respect to $\preceq$, we know that if any sentence φ' 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 predicates, but *not* to generate those candidate sentences that cannot be interesting given all the information obtained in earlier iterations [2; 22].

The method is as follows.

Algorithm 3 The *levelwise algorithm* for finding all interesting statements.
Input: A database $\mathbf{r}$, a language $\mathcal{L}$ with specialization relation $\preceq$, and a quality function q.
Output: The set $Th(\mathcal{L}, \mathbf{r}, q)$.
Method:

1. $C_1 := \{\varphi \in \mathcal{L} \mid$ there is no φ' in $\mathcal{L}$ such that $\varphi' \prec \varphi\}$;
2. $i := 1$;
3. **while** $C_i \neq \emptyset$ **do**
4. $\quad L_i := \{\varphi \in C_i \mid q(\mathbf{r}, \varphi)\}$;
5. $\quad C_{i+1} := \{\varphi \in \mathcal{L} \mid$ for all $\varphi' \prec \varphi$ we have $\varphi' \in \bigcup_{j \leq i} L_j\} \setminus \bigcup_{j \leq i} C_j$;
6. $\quad i := i + 1$;
7. **od**;
8. output $\bigcup_{j<i} L_j$; □

The algorithm works iteratively, alternating between *candidate generation* and *evaluation* phases. First, in the generation phase of an iteration i, a collection C_i of new *candidate* sentences is generated, using the information available from more general sentences. Then the quality function is computed for these candidate sentences. The collection L_i will consist of the interesting sentences in C_i. In the next iteration $i + 1$, candidate sentences in C_{i+1} are generated using the information about the interesting sentences in $\bigcup L_j$.

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 Step 5 used only a negligible amount of time [2].

Lemma 4 Algorithm 3 computes $Th(\mathcal{L}, \mathbf{r}, q)$ correctly. □

For Algorithm 3 to be applicable, several conditions have to be fulfilled. The language $\mathcal{L}$ and the interestingness predicate have to be such that the size of $Th(\mathcal{L}, \mathbf{r}, q)$ is not too big. (It is not strictly necessary that all sentences in $Th(\mathcal{L}, \mathbf{r}, q)$ are truly interesting to the user: $Th(\mathcal{L}, \mathbf{r}, q)$ can be further pruned using, e.g., statistical significance or other criteria [14]. But $Th(\mathcal{L}, \mathbf{r}, q)$ should not contain hundreds of thousands of useless rules.)

3 Examples

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

Example 5 For association rules, the specification ordering was already given above. The algorithm will perform k iterations of the outermost loop, i.e., read the database k times, where $k-1$ is the size of the largest subset X such that $s(X)$ exceeds the given threshold. See [2; 11; 22; 28] for various implementation methods. □

Example 6 *Strong rules* [26] are rules of the form `if` *expression* `then` *expression*, where the expressions are, e.g., of the form $A < 40$, $B = 1$, etc. 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 [12; 16; 19]. Given a database schema $\mathbf{R}$, an *inclusion dependency* (IND) 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.

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 we are interested in 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 quality 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 $\sigma = R[X] \subseteq S[Y]$ and $\tau = R'[X'] \subseteq S'[Y']$, we have $\sigma \preceq \tau$ only if $R = R'$, $S = S'$, and furthermore $X' = (A_1, \ldots, A_k), Y' = (B_1, \ldots, B_k)$, and for some $i_1, \ldots, i_h \in \{1, \ldots, k\}$ we have $X = (A_{i_1}, \ldots, A_{i_k}), Y = (B_{i_1}, \ldots, B_{i_k})$.

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

The next example shows a case which the levelwise algorithm does not suit particularly well.

Example 8 Given a relation r over attributes R, a functional dependency is an expression $X \rightarrow B$, 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; 19; 20; 21; 25]. Such dependencies can be found using the levelwise algorithm by considering the set of sentences $\{X \mid X \subseteq R\}$, and the interestingness predicate q: $q(r, X)$ iff $X \rightarrow B$ 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 interestingness predicate is monotone w.r.t. $\preceq$. 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 $1 + |R \setminus X|$, where X is the smallest set such that $X \rightarrow B$ holds in r. Note that in this case for a large R there will be many iterations, even though the answer might be representable succintly.

Note that one can avoid this problem by shifting 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 interesting sentences

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

The *border* $Bd(S)$ of S consists of those sentences σ such that all generalizations of σ are in S and none of the specializations of σ is in S. Those sentences σ in $Bd(S)$ that are in S are called the *positive border*[1] $Bd^+(S)$, and those sentences σ in $Bd(S)$ that are not in S are the *negative border* $Bd^-(S)$. In other words, $Bd(S) = Bd^+(S) \cup Bd^-(S)$, where $Bd^+(S) = \{\sigma \in S \mid \text{for all } \gamma \text{ with } \sigma \prec \gamma, \text{ we have } \gamma \notin S\}$ and $Bd^-(S) = \{\sigma \in \mathcal{L} \setminus S \mid \text{for all } \gamma \prec \sigma, \text{ we have } \gamma \in S\}$.

Note that $Bd(S)$ can be very small even for large S. Using this notation Step 5 of Algorithm 3 can be written as $C_{i+1} := Bd^-(\bigcup_{j \leq i} L_j) \setminus \bigcup_{j \leq i} C_j$.

[1] I.e., the positive border corresponds to the set "S" of [24].

Theorem 9 Algorithm 3 uses $|Th(\mathcal{L}, \mathbf{r}, q) \cup Bd^-(Th(\mathcal{L}, \mathbf{r}, q))|$ evaluations of the interestingness predicate q. □

Some straightforward lower bounds for the problem of finding all frequent sets are given in [2; 22]. Now we consider the problem of lower bounds in more realistic models of computation.

The main effort in finding interesting sets is in the step where the interestingness of subgroups are evaluated 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 interesting, i.e., does $q(\mathbf{r}, \varphi)$ hold?

Note that Algorithm 3 falls within this model of computation.

Theorem 10 Any algorithm for computing $Th(\mathcal{L}, \mathbf{r}, q)$ that accesses the data using only Is-interesting queries must use at least $|Bd(Th(\mathcal{L}, \mathbf{r}, q))|$ queries. □

This result, simple as it seems, gives as a corollary a result about finding functional dependencies that in the more specific setting is not easy to find [19; 20]. 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 11 ([20]) Given a relation r over schema R, finding the minimal keys that hold in r requires at least MAX(r) evaluations of the predicate "Is X a key", where 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 10 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 consider the complexity of evaluation of the interestingness 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 $X \to Y$ 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 verify simulataneously the interestingness of several sets, whereas to verify the truth of a set of dependencies requires in general as many passes through the database as there are dependencies.

5 Sampling: the guess-and-correct algorithm

Algorithm 3 starts by evaluating the interestingness of 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 interesting statement will be overlooked. However, the approach can also be quite slow in the case there are interesting statements that are far from the bottom of the specialization hierarchy, i.e., if there are statements σ that turn out to be interesting, but which appear in the candidate set 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 $Th(\mathcal{L}, \mathbf{r}, q)$ from an initial guess $S \subseteq \mathcal{L}$, and then correcting the guess by looking at the database. The guess can be obtained, e.g., from computing the set $Th(\mathcal{L}, \mathbf{s}, q)$, where $\mathbf{s}$ is a sample of $\mathbf{r}$.

The *guess-and-correct algorithm* for computing $Th(\mathcal{L}, \mathbf{r}, q)$ is as follows. Given are a database $\mathbf{r}$, a language $\mathcal{L}$ with specialization relation $\preceq$, a quality function q, and an initial guess $S \subseteq \mathcal{L}$ for $Th(\mathcal{L}, \mathbf{r}, q)$. First, evaluate the sentences in the positive border $Bd^+(S)$ and remove from S those that are not interesting. Repeat the evaluation—removal step until the positive border only contains interesting sentences, i.e., $S \subseteq Th(\mathcal{L}, \mathbf{r}, q)$. Now expand S upwards, as in the original algorithm: evaluate such sentences in the negative border $Bd^-(S)$ that have not been evaluated yet, and add the interesting ones to S. Repeat the evaluation—addition step until there are no sentences to evaluate. Output $S = Th(\mathcal{L}, \mathbf{r}, q)$.

One can show that this algorithm uses $|(S \triangle Th) \cup Bd^-(S) \cup Bd^+(S \cap Th)|$ evaluations of q, where $Th = Th(\mathcal{L}, \mathbf{r}, q)$. Thus the better the estimate S for $Th(\mathcal{L}, \mathbf{r}, q)$ is, the faster the algorithm is.

For finding frequent sets and for finding functional dependencies one can show that sampling produces fairly good approximations. We omit the details.

Another method for computing an initial approximation can be derived from the algorithm of [28]. The idea is to divide $\mathbf{r}$ into small datasets $\mathbf{r}_i$ which can be handled in main memory and compute $S_i = Th(\mathcal{L}, \mathbf{r}_i, q)$. In the case of frequent sets, use as the guess S the union $\bigcup_i S_i$; in the case of functional dependencies, use as S the intersection $\bigcap_i S_i$. In both cases, the guess is a superset of $Th(\mathcal{L}, \mathbf{r}_i, q)$, 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 $S \subseteq \mathcal{L}$, and claims that $S = Th(\mathcal{L}, \mathbf{r}, q)$. How many evaluations of q do we have to do to check this claim?

Theorem 12 Given $\mathcal{L}, \mathbf{r}, q$, and a set $S \subseteq \mathcal{L}$, determining whether $S = Th(\mathcal{L}, \mathbf{r}, q)$ requires in the worst case at least $|Bd(S)|$ evaluations of the predicate q, and it can be solved using exactly this number of evaluations of q. □

Example 13 Given a relation r over $\{A, B, C, D\}$, suppose a sample or some person tells us that $\{A, B\}$ and $\{A, C\}$ and their supersets are the only keys of r. Recall that for this case $X \preceq Y$ if and only if $Y \subset X$. To verify this, we have to check according to Theorem 12 the set $Bd(S)$ for $S = \{X \subseteq \{A, B, C, D\} \mid \{A, B\} \subseteq X \vee \{A, C\} \subseteq X\}$. The positive border of S is $\{\{A, B\}, \{A, C\}\}$, and $Bd^-(S) = \{\{B, C, D\}, \{A, D\}\}$, and we have to inspect the sets $\{A, B\}, \{A, C\}, \{B, C, D\}, \{A, D\}$ to determine whether $\{A, B\}$ and $\{A, C\}$ and their supersets really are the only keys of r. □

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 the f is one-to-one and surjective, f and its inverse are computable, and for all θ and φ we have $\theta \preceq \varphi$ if and only if $f(\theta) \subseteq f(\varphi)$.

Note that 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.[2]

A collection $\mathcal{H}$ of subsets of R is a (*simple*) *hypergraph*, 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 $\mathrm{Tr}(\mathcal{H})$. It is a hypergraph on R.

Now we return to the verification problem. Given $S \subseteq \mathcal{L}$, we have to determine whether $S = Th(\mathcal{L}, \mathbf{r}, q)$ holds using as few evaluations of the interestingness predicate as possible.

Given S, we can compute $Bd^+(S)$ without looking at the data $\mathbf{r}$ at all: simply find the most special sentences in S. The negative border $Bd^-(S)$ is also determined by S, but finding the most general sentences in $\mathcal{L} \setminus S$ can be difficult. We now show how minimal transversals can be used in the task. Assume that (f, R) represents $\mathcal{L}$ as sets, and consider the hypergraph $\mathcal{H}(S)$ on R containing as edges the complements of sets $f(\varphi)$ for $\varphi \in Bd^+(S)$: $\mathcal{H}(S) = \{R \setminus f(\varphi) \mid \varphi \in Bd^+(S)\}$. Then $\mathrm{Tr}(\mathcal{H}(S))$ is a hypergraph on R, and hence we can apply f^{-1} to it: $f^{-1}(\mathrm{Tr}(\mathcal{H}(S))) = \{f^{-1}(H) \mid H \in \mathrm{Tr}(\mathcal{H}(S))\}$. We have the following.

Theorem 14 $f^{-1}(\mathrm{Tr}(\mathcal{H}(S))) = Bd^-(S)$. □

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

Example 15 Continuing Example 13, we compute the set $Bd^-(S)$ using the hypergraph formulation. Now the representation of keys as sets is simple: $f(X) = R \setminus X$. Hence $\mathcal{H}(S) = \{R \setminus f(X) \mid X \in S\} = S$. Thus $\mathrm{Tr}(\mathcal{H}(S)) = \{\{A\}, \{B, C\}\}$, and $f^{-1}(\mathrm{Tr}(\mathcal{H}(S))) = \{\{B, C, D\}, \{A, D\}\}$. □

The advantage of Theorem 14 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 [21]; 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; 23] for recent breakthroughs.

[2]Actually, for every $\mathcal{L}$ one can devise a representation of $\mathcal{L}$ as sets by letting $f(\theta) = \{\gamma \in \mathcal{L} \mid \gamma \preceq \theta\}$. This representation, however, is not very useful.

Acknowledgements

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

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*. AAAI Press, Menlo Park, CA, 1996. To appear.

[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, 1989.

[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. To appear.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] A. J. Knobbe and P. W. Adriaans. Discovering foreign key relations in relational databases. In *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pages 94 – 99, Heraklion, Crete, Greece, Apr. 1995.

[17] P. Langley. *Elements of Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1995.

[18] 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.

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

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

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

[22] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In U. M. Fayyad and R. Uthurusamy, editors, *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192, Seattle, Washington, July 1994.

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

[24] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203 – 226, 1992.

[25] 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.

[26] 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.

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

[28] 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, 1995.