Data Mining and Knowledge Discovery, 3, 7–36 (1999)

Discovery of frequent DATALOG patterns

LUC DEHASPE luc.dehaspe@cs.kuleuven.ac.be
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium

HANNU TOIVONEN hannu.toivonen@rni.helsinki.fi
Rolf Nevanlinna Institute & Department of Computer Science, P.O. Box 4, FIN-00014 University of Helsinki, Finland

Editor: Sašo Džeroski and Nada Lavrač

Abstract. Discovery of frequent patterns has been studied in a variety of data mining settings. In its simplest form, known from association rule mining, the task is to discover all frequent itemsets, i.e., all combinations of items that are found in a sufficient number of examples. The fundamental task of association rule and frequent set discovery has been extended in various directions, allowing more useful patterns to be discovered with special purpose algorithms. We present WARMR, a general purpose inductive logic programming algorithm that addresses *frequent query discovery*: a very general DATALOG formulation of the frequent pattern discovery problem.

The motivation for this novel approach is twofold. First, exploratory data mining is well supported: WARMR offers the flexibility required to experiment with standard and in particular novel settings not supported by special purpose algorithms. Also, application prototypes based on WARMR can be used as benchmarks in the comparison and evaluation of new special purpose algorithms. Second, the unified representation gives insight to the blurred picture of the frequent pattern discovery domain. Within the DATALOG formulation a number of dimensions appear that relink diverged settings.

We demonstrate the frequent query approach and its use on two applications, one in alarm analysis, and one in a chemical toxicology domain.

Keywords: frequent patterns, inductive logic programming, DATALOG queries, association rules, episodes, sequential patterns

1. Introduction

Discovery of recurrent patterns in large data collections has become one of the central topics in data mining. In tasks where the goal is to uncover structure in the data and where there is no preset target concept, the discovery of relatively simple but frequently occurring patterns has shown good promise.

Association rules (Agrawal et al, 1993) are a basic example of this kind of setting. A prototypical application example is in market basket analysis: first find out which items tend to be sold together, the frequent item set discovery phase, and next postprocess these frequent patterns into rules about the conditional probability that one set of items will be in the basket given another set already there, the association rule discovery phase. The motivation for such an application is the potentially high business value of the discovered patterns. At the heart of the task is the problem of determining all combinations of items that occur frequently together, where "frequent" is defined as "exceeding a user-specified frequency threshold". The use of a frequency threshold for filtering out non-interesting patterns is natural for a large number of data mining problems. Patterns that are rare, e.g., that concern only a couple of customers, are probably not reliable nor useful for the user.

A family of data mining problems can be specified as follows (Mannila and Toivonen, 1997). Given a database $\mathbf{r}$, a class $\mathcal{L}$ of sentences (patterns), and a selection predicate q, the task is to find the theory of $\mathbf{r}$ with respect to $\mathcal{L}$ and q, i.e., the set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q) = \{Q \in \mathcal{L} \mid q(\mathbf{r}, Q) \text{ is true}\}$. The selection predicate q is used for evaluating whether a sentence $Q \in \mathcal{L}$ defines a (potentially) interesting pattern in $\mathbf{r}$. For the discovery of frequent patterns, q is defined so that $q(\mathbf{r}, Q)$ is true if and only if the frequency of pattern Q in database $\mathbf{r}$ exceeds the frequency threshold. Problem settings that fit this description and that are close to the problem of discovering association rules include the use of item type hierarchies (Han and Fu, 1995, Holsheimer et al., 1995, Srikant and Agrawal, 1995), the discovery of episodes in event sequences (Mannila and Toivonen, 1996, Mannila et al., 1997), and the search of sequential patterns from series of transactions (Agrawal and Srikant, 1995, Srikant and Agrawal, 1996). For all these cases the pattern language $\mathcal{L}$ is different, and specialized algorithms exist for the tasks.

We present a powerful inductive logic programming algorithm, WARMR, for a large subfamily of this type of tasks. WARMR discovers frequent DATALOG queries that succeed with respect to a sufficient number of examples. In other words, the language $\mathcal{L}$ consists of DATALOG queries, and WARMR outputs those that are "frequent" in a given DATALOG (or relational) database $\mathbf{r}$. The DATALOG formulation is very general, as it allows the use of variables and multiple relations in patterns, and it thus significantly extends the expressive power of patterns that can be found.

The flexibility of WARMR is a strong advantage over previous algorithms for the discovery of frequent patterns. Each discovery task is specified to WARMR in terms of a declarative language bias definition. The language bias declarations determine which DATALOG queries are admissible, i.e., which subset of DATALOG constitutes the language $\mathcal{L}$ for the particular task. With different languages (and databases) WARMR can be adapted to diverse tasks, including the settings mentioned above and also more complex novel problems, without requiring changes to the implementation. WARMR thus supports truly explorative data mining: pattern types can be modified and experimented with very flexibly with a single tool.

This article is organized as follows. We start in Section 2 by describing the data mining task of discovering frequent DATALOG queries. In Section 3 we present WARMR, an algorithm that discovers frequent DATALOG queries. We show in Section 4 how WARMR and the described setting can be used to implement and extend some well-known data mining tasks. Two novel applications of frequent DATALOG queries and WARMR are discussed in Section 5, one in telecommunication alarm analysis and one in a chemical toxicology domain. Finally, in Sections 6 and 7, we touch upon related work, and conclude.

2. The frequent query discovery task

We use DATALOG (see, e.g., (Ullman, 1988)) to represent data and patterns. There is a straightforward and well-defined correspondence between DATALOG and both relational databases and first-order clausal logic. The use of DATALOG allows us to describe a number of data mining tasks in the area of frequent pattern discovery in a clear and uniform manner.

Relational algebra, the formal framework of relational databases, has the same expressive power as DATALOG without recursion. For instance, the recursive concept *ancestor* can be

defined in DATALOG but not in relational algebra. DATALOG, in turn, is a subset of clausal logic (and PROLOG) that is restricted to function-free definite clauses. In this paper, we restrict ourselves to DATALOG to simplify the discussion. It should be stressed however that the algorithms described below in practice operate in the more general PROLOG setting.

We now briefly review the DATALOG concepts used in this paper, and describe the data mining task of discovering frequent patterns or, in DATALOG terminology, frequent queries. Next, we introduce a formalism to specify a remaining essential parameter of the frequent query discovery task: a search space $\mathcal{L}$ of DATALOG queries.

2.1. DATALOG *concepts*

In DATALOG a *term* is defined as a constant symbol or a variable. To distinguish between them, we write variables with an initial upper case letter, while using names beginning with lower case letters for constants. An *atom* is an m-ary *predicate* symbol followed by a bracketed m-tuple of terms. A *definite clause* is a universally quantified formula of the form $B \leftarrow A_1, \ldots, A_n$ $(n \geq 0)$, where B and the A_i are atoms. This formula can be read as "B if A_1 and ... and A_n". If $n = 0$ a definite clause is also called a *fact*. *Ground* clauses are clauses that contain only constants as terms, no variables. A *substitution* θ is a set of bindings $\{X_1/a_1, \ldots, X_m/a_m\}$ of variables X_i to terms a_i. If we substitute a_i for each occurrence of X_i in a clause C, we obtain $C\theta$, the *instance* of C by substitution θ. If $C\theta$ is ground, it is called a *ground instance* of formula C, and θ is called a *grounding substitution*.

A deductive DATALOG *database* is a set of definite clauses. Often a distinction is made between the *extensional* database, which contains the predicates defined by means of ground facts only, and the remaining *intensional* part.

A DATALOG (and PROLOG) *query* is a logical expression of the form ?- A_1, ..., A_n. Submitting such a query to a DATALOG database corresponds to asking the question "does a grounding substitution exist such that conjunction A_1 and ... and A_n holds within the database". The (resolution based derivation of the) answer to a given query with variables $\{X_1, \ldots, X_m\}$ binds these variables to terms $\{a_1, \ldots, a_m\}$, such that the query *succeeds* if a_i is substituted for each X_i. This so-called *answering substitution* is denoted by $\{X_1/a_1, \ldots, X_m/a_m\}$. Due to the nondeterministic nature of the computation of answers, a single query Q may result in many answering substitutions. We will refer by *answerset* $(Q, \mathbf{r})$ to the set of all answering substitutions obtained by submitting query Q to a DATALOG database $\mathbf{r}$.

As already pointed out, if recursion is not allowed, these concepts correspond directly to relational database terminology. Predicates map to relations, facts to tuples of a relation, and a query such as

?- *window(Window_id, Alarmtype), is_a(Alarmtype, warning)*

can be written in SQL as

```
SELECT   WINDOW.window_id, WINDOW.alarmtype
FROM     WINDOW, IS_A
WHERE    WINDOW.alarmtype=IS_A.alarmtype
AND      IS_A.ancestor='warning'
```

An overview of strategies to make the relationship between DATALOG and relational databases operational can be found in (Ullman, 1988).

2.2. Frequent query discovery

In terms of the generic formulation of the frequent pattern discovery problem by Mannila and Toivonen (1997) (see also above, Section 1), we consider the following data mining task.

Definition 1. Assume $\mathbf{r}$ is a DATALOG database, $\mathcal{L}$ is a set of DATALOG queries Q that all contain an atom *key*, and $q(\mathbf{r}, Q)$ is true if and only if the frequency of query $Q \in \mathcal{L}$ with respect to $\mathbf{r}$ given *key* is at least equal to the frequency threshold specified by the user. The *frequent query discovery task* is to find the set $\mathcal{T}h(\mathcal{L}, \mathbf{r}, q, \textit{key})$ of frequent queries.

Compared to the original formulation, we have added a fourth parameter, i.e., atom *key*, to the frequent pattern discovery task. In our framework this extra parameter is essential, as it determines *what* is counted. As we will shortly clarify in our definition of frequency, each binding of the variables in *key* uniquely identifies an entity.

In previous formulations of the task, the *key* unambiguously followed from the context. For instance, we count transactions in market basket analysis or windows in episode discovery. In frequent query discovery, with many predicates in the database, the focus of counting can be on any attribute and has to be specified by the user. The additional *key* parameter allows us to switch, for instance, from counting transactions, to customers, to supermarkets, to managers, to regions or to whatever relevant attribute in our market basket database.

We next define what frequency exactly means in this setting.

Definition 2. Assume $\mathbf{r}$ is a DATALOG database and $Q \in \mathcal{L}$ is a query that contains atom *key*. Then the relative *frequency* of query Q with respect to database $\mathbf{r}$ given *key* is

$$\textit{frq}\,(Q, \mathbf{r}, \textit{key}) = \frac{|\{\theta_k \in \textit{answerset}\,(\text{?- } \textit{key}, \mathbf{r}) \mid Q\theta_k \text{ succeeds w.r.t. } \mathbf{r}\}|}{|\{\theta_k \in \textit{answerset}\,(\text{?- } \textit{key}, \mathbf{r})\}|}$$

i.e., the fraction of substitutions of the key variables with which the query Q is true, or, more intuitively, the fraction of examples in which pattern Q occurs.

EXAMPLE: Consider the analysis of alarms from a telecommunication network (Klemettinen et al, 1998). The task is to discover episodes (Mannila et al., 1997), combinations of alarms that tend to occur close to each other and that are thus potentially causally related. Given a long sequence of alarms and a window width, one looks at the sequence by sliding a window of the given width on the sequence. A set of alarms that occurs frequently in the windows is called a parallel episode (Mannila et al., 1997).

For the episode discovery task, we store in the database $\mathbf{r}$ facts of the form *window(wid, atype)*$\leftarrow$ to indicate that window *wid* contains an alarm of type *atype*. Notice that in a practical implementation this information should probably be computed in an incremental manner at run time.

There are a large number of alarm types, and it is often useful to study episodes on different levels of abstraction. For instance, alarm type 1001 belongs to the class of switch alarms, but also to the class of BSC disturbances and the class of BSC alarms. In general, alarm types form an "is a" hierarchy (with the shape of a lattice). This information can be provided as background knowledge, in the form of facts *is_a(1001, switch)*←, *is_a(1001, BSC_disturbance)*←, *is_a(BSC_disturbance, BSC_alarm)*←, and so on. The transitive definition of *is_a* can be specified as a clause in the intensional part of the background database.

Discovery of frequent parallel episodes with an alarm type hierarchy can now be defined as a special case of frequent query discovery as follows. The database **r** contains, for each time window *wid* on the input alarm sequence, a fact *window_id(wid)*← and zero or more facts *window(wid, atype)*←. As we want to count windows, we set *key* to *window_id(Wid)*. The database **r** further contains, as domain knowledge, facts of the form *is_a(atype, parent_atype)*←. The patterns in $\mathcal{L}$ consist of the obligatory *key* atom *window_id(Wid)* together with one or more atoms or atom pairs of the form *window(Wid, atype$_i$)* and *window(Wid, AType$_j$), is_a(AType$_j$, atype$_k$)*.

Suppose we want to count the frequency of query

Q = ?- window_id(Wid), window(Wid, AType), is_a(AType, switch)

i.e., "the window contains an alarm type that belongs to the class of *switch* alarms". For each fact *window_id(wid$_k$)*← there is a substitution

$$\theta_k = \{\textit{Wid/wid}_k\} \in \textit{answerset}\,(\textit{?- window\_id(Wid)}, \mathbf{r})\ .$$

We then have to find the fraction of such substitutions θ_k for which

$$Q\theta_k = \textit{?- window\_id(wid}_k\textit{), window(wid}_k\textit{, AType), is\_a(AType, switch)}$$

succeeds.

This situation is actually identical to the discovery of frequent sets for association rules. For the supermarket basket analysis application, simply replace windows by transactions and alarm types by item types. The problem of discovering such frequent sets on multiple levels of an item type hierarchy has been considered, e.g., in (Han and Fu, 1995, Holsheimer et al., 1995, Srikant and Agrawal, 1995). □

To conclude the description of the frequent query discovery task, let us once more establish the link with relational database terminology. In SQL syntax the absolute frequency of Q can be obtained with the following query, inspired by (Lindner and Morik, 1995, Blockeel and De Raedt, 1996):

```
SELECT   count(distinct *)
FROM     SELECT   fields that correspond to the variables in key
         FROM     relations in Q
         WHERE    conditions expressed in Q
```

2.3. Declarative language bias

The representation of patterns as DATALOG queries requires a formalism to constrain the query language $\mathcal{L}$ to a set of meaningful and useful patterns. With association rules the

definition of $\mathcal{L}$ is straightforward: $\mathcal{L}$ is simply 2^I, the collection of all subsets of the set I of items. Srikant, Vu, and Agrawal (Srikant et al, 1997) describe a technique to impose and exploit user-defined constraints on combinations of items, but otherwise the definition of $\mathcal{L}$ has received little attention in the frequent pattern discovery literature. In inductive logic programming, on the other hand, this issue has been studied extensively in the subfield of declarative language bias. This is motivated by huge, often infinite, search spaces, that require a tight specification of patterns worth considering. Several formalisms have been proposed for adding language bias information in a declarative manner to the search process (for an overview, see (Adé et al, 1995, Nédellec et al., 1996)). Our formalism, WRMODE, is an adaptation to WARMR of the RMODE format developed for TILDE (Blockeel and De Raedt, 1998) which, in turn, is based on the formalism originally developed for PROGOL (Muggleton, 1995).

We will demonstrate later how the WRMODE notation can be used to constrain $\mathcal{L}$ to some interesting classes of patterns. The required subset of WRMODE is described below.

2.3.1. The WRMODE *basics* Let us first look at the simple case where $\mathcal{L}$ contains no variables, i.e., only ground queries are allowed. Under these circumstances, the WRMODE notation extends the straightforward $\mathcal{L} = 2^I$ bias to DATALOG queries: given a set $Atoms$ of ground atoms, the language $\mathcal{L}$ consists of 2^{Atoms}, i.e., of all possible combinations of the atoms. For example, $Atoms$ = {*p(a,b), q(c)*} defines $\mathcal{L} = 2^{Atoms}$ = {*?- true; ?- p(a,b); ?- q(c); ?- p(a,b),q(c)*}.

When variables are allowed in $\mathcal{L}$, the power set idea can be extended to a set of literals, as done, for instance, in (Weber, 1997) and (Dehaspe and De Raedt, 1997). However, this solution is inconvenient for two reasons. First, we might want to define infinite languages. For instance, in a graph represented as a set of DATALOG facts *edge(From,To)* we might want to allow queries *?- edge(X_1,X_2), edge(X_2,X_3), edge(X_3,X_4), ...* of arbitrary length. Thereto, an atom in $Atoms$ should be allowed several times in the query and not just once as in 2^{Atoms}. Second, we do not want to control the exact names of the variables in the query, as we do for constants, but rather the sharing of names between variables. For example, query *?- rectangle(Width,Height), Width<Height* is equivalent to query *?- rectangle(X,Y), X<Y* but not to query *?- rectangle(Width,Height),X<Y*.

In the WRMODE framework, non-ground atoms in $Atoms$ are allowed to occur multiple times in the query, as long as their variables obey the so-called *mode constraints*. These are declared for each variable argument of each atom by means of three *mode-labels* $+$, $-$, and $\pm$:

$+$	the variable is strictly *input*, i.e. bound before the atom is called
$-$	the variable is strictly *output*, i.e. bound by the atom
$\pm$	the variable can be both input and/or output, i.e. anything goes

In our approach the atoms of the query are evaluated one by one, following an ordering that is consistent with the mode declarations. The intuition is that the evaluation of some atoms, such as $X < Y$ in the example above, presupposes the binding of certain *input* variables to a constant. On the other hand some atoms, e.g. *rectangle(Width,Height)*, are allowed or required to introduce new *output* variables bound during evaluation of the atom.

Table 1. Examples of WRMODE definitions and queries (dis)allowed in the corresponding languages.

WRMODE **definition** $Atoms$	**example pattern** $\in \mathcal{L}$	**example pattern** $\notin \mathcal{L}$
{*p*(−,−),*q*(−)}	*?- p(X,Y),q(Z)*	*?- p(X,Y),q(Y)*
{*p*(+,−),*q*(−)}	*?- q(X),p(X,Y),p(Y,Z)*	*?- p(X,Y)*
{*p*(+,±),*q*(−)}	*?- q(X),p(X,X),p(X,Y)*	*?- q(X),p(Y,X)*
{*p*(±,*a*),*q*(±)}	*?- p(X,a),q(X),q(Z),p(Z,a)*	*?- p(X,Y)*

A query is then mode conform if an ordering of atoms exists such that every input variable occurs in one of the previous atoms, and no output variable does. Some examples of queries that are consistent and inconsistent with mode declarations are listed in Table 1. Throughout the paper we use the notational convention that the atoms of a query are evaluated from left to right.

2.3.2. Typing in WRMODE Additional constraints on the sharing of variable names can be imposed via type declarations. The WRMODE convention is to append these to the mode declarations in $Atoms$. A query is then type conform if and only if arguments that share a variable name have identical types or at least one of them is untyped. For example, with mode and type declarations $Atoms$ = {*p*(−*s*,−*t*),*q*(+*t*,*a*)}, query *?- p(X,Y),q(Y,a)* is in $\mathcal{L}$, but not *?- p(X,Y),q(X,a)* since the first arguments of predicates p and q have incompatible types. Notice the difference between constants and types in declarations $Atoms$: types are preceded by a mode label and constants are not.

2.3.3. The WRMODE *key* As we have seen, the frequent query discovery task requires the specification of a *key* atom which is obligatory in all queries. Within WRMODE notation this is done with *key = KeyAtom*, where *KeyAtom* is a mode and type declaration as defined above. Obviously, the key atom declaration should not contain any + mode labels.

For an example of a non-ground language, consider *key = p(−,−)* and $Atoms$ = {*q*(+)}. These declarations together define $\mathcal{L}$ = {*?- p(X,Y); ?- p(X,Y),q(X); ?- p(X,Y),q(Y); ?- p(X,Y),q(X),q(X); ?- p(X,Y),q(X),q(Y);* ...}. Notice the presence of logically redundant queries such as *?- p(X,Y),q(X),q(X).*

2.3.4. Logical redundancy and WRMODE The WRMODE notation is only meant to capture application-specific constraints on $\mathcal{L}$. These are usually supplemented by a number of general constraints hardwired in the data mining algorithm in which WRMODE is embedded. Logical non-redundancy is such an application-neutral and algorithm-specific constraining principle. For instance, the last WRMODE definition in Table 1 allows *?- p(X,a), q(X), p(Y,a), q(Y), q(Y),* which is logically equivalent to the shorter *?- p(X,a),q(X).* Within WARMR the first pattern would be filtered away in the candidate generation phase, as explained below in Section 3.2.3.

2.3.5. WRMODE *extensions* The implementation of WRMODE contains many extra features that provide more expressive power and allow more condensed notations. One general mechanism is to make the presence of atoms conditional on the presence or absence of other atoms. For instance, in the case of alarm analysis, the query *?- window_id(Wid), window(Wid,AType)* is bound to succeed for all windows, assuming all windows contain at least one alarm. To avoid the evaluation of such uninteresting patterns we could require that *window(Wid,AType)* only occurs in combination with an atom *is_a(AType,atype)*. The details of how to achieve this with WRMODE are beyond the scope of this article, and we will make abstraction of this and similar extensions in the rest of the paper.

EXAMPLE: In the previous example we stated that for the parallel episode discovery task, the patterns in $\mathcal{L}$ consist of an atom *window_id(Wid)* together with atoms or atom pairs of the forms *window(Wid, atype$_i$)* and *window(Wid, AType$_j$), is_a(AType$_j$, atype$_k$)*.

For the WRMODE specification of this we first set the key atom by specifying

key = window_id(−w)

In words, we want to count windows, not alarm types or whatever. For the set of atoms that can be used in constructing queries we set

$$Atoms = \{\ \textit{window(+w,atype}_1\textit{)}, \ldots, \textit{window(+w,atype}_m\textit{)},\ \textit{window(+w,−a), is_a(+a,aclass}_1\textit{)}, \ldots, \textit{is_a(+a,aclass}_n\textit{)}\}$$

for all m alarm types and n alarm classes. According to these declarations, the first argument of *window* has to be an input variable of the same type w as the argument of *window_id*, and the second argument is either a constant from $\{$*atype*$_1$,...,*atype*$_m\}$ or an output variable different from the variable in *window_id*. The first argument of *is_a* is an input variable of the same type a as the second argument of *window*. Finally, the second argument of *is_a* is a constant from $\{$*aclass*$_1$,...,*aclass*$_n\}$. □

3. Query discovery with WARMR

Design of algorithms for frequent pattern discovery has turned out to be a popular topic in data mining (for a sample of algorithms, see (Agrawal et al, 1993, Agrawal et al, 1996, Lu et al, 1995, Savasere et al, 1995, Toivonen, 1996)). Almost all algorithms are on some level based on the same idea of levelwise search, known in data mining from the APRIORI algorithm (Agrawal et al, 1996). We first review the generic levelwise search method and its central properties and then introduce the algorithm WARMR (Dehaspe and De Raedt, 1997) for finding frequent queries. To conclude this section, we recall how this method fits in the two-phased discovery of frequent and confident rules.

3.1. The levelwise algorithm

The levelwise algorithm (Mannila and Toivonen, 1997) is based on a breadth-first search in the lattice spanned by a specialization relation $\preceq$ between patterns, cf. (Mitchell, 1982), where $p1 \preceq p2$ denotes "pattern $p1$ is more general than pattern $p2$", or "$p2$ is more specific than pattern $p1$".

Algorithm 1 *:* WARMR

`Inputs:` Database **r**; WRMODE language $\mathcal{L}$ and *key* ; threshold *minfreq*
`Outputs:` All queries $Q \in \mathcal{L}$ with *frq* $(Q, \mathbf{r}, \textit{key}) \geq \textit{minfreq}$

1. Initialize level $d := 1$
2. Initialize the set of candidate queries $\mathcal{Q}_1 := \{?\text{- } \textit{key}\}$
3. Initialize the set of infrequent queries $\mathcal{I} := \emptyset$
4. Initialize the set of frequent queries $\mathcal{F} := \emptyset$
5. While $\mathcal{Q}_d$ not empty
6. Find *frq* $(Q, \mathbf{r}, \textit{key})$ of all $Q \in \mathcal{Q}_d$ using WARMR-EVAL
7. Move the queries $\in \mathcal{Q}_d$ with frequency below *minfreq* to $\mathcal{I}$
8. Update $\mathcal{F} := \mathcal{F} \cup \mathcal{Q}_d$
9. Compute new candidates $\mathcal{Q}_{d+1}$ from $\mathcal{Q}_d$, $\mathcal{F}$ and $\mathcal{I}$ using WARMR-GEN
10. Increment d
11. Return $\mathcal{F}$

The method looks at a level of the lattice at a time, starting from the most general patterns. The method iterates between candidate generation and candidate evaluation phases: in *candidate generation*, the lattice structure is used for pruning non-frequent patterns from the next level; in the *candidate evaluation* phase, frequencies of candidates are computed with respect to the database. Pruning is based on monotonicity of $\preceq$ with respect to frequency: if a pattern is not frequent then none of its specializations is frequent. So while generating candidates for the next level, all the patterns that are specializations of infrequent patterns can be pruned. For instance, in the APRIORI algorithm for frequent itemsets, candidates are generated such that all their subsets (i.e., generalizations) are frequent.

The levelwise approach has two crucial useful properties:

- Assuming all candidates of a level are tested in single database pass, the database is scanned at most $d + 1$ times, where d is the maximum level (size) of a frequent pattern. This is an important factor when mining large databases.

- The time complexity is in practice linear in the product of the size of the result times the number of examples, assuming matching patterns against the data is fast.

3.2. The WARMR *algorithm*

The inputs of WARMR correspond to the four parameters of the frequent query discovery task as introduced in Definition 1. Algorithm 1, steps (5–10), shows WARMR's main loop as an iteration of candidate evaluation in step (6) and candidate generation in step (9). The manipulation of set $\mathcal{I}$ of infrequent queries in steps (3) and (7) is necessary for the generation phase. This is discussed below, together with some other features that distinguish WARMR from APRIORI.

3.2.1. Specialization relation The subset specialization relation used in most frequent pattern discovery settings can in some restricted cases also be used for structuring a space of DATALOG queries, as done in (Weber, 1997, Weber, 1998) and (Dehaspe and De Raedt, 1997). In general however, the subset condition is too strong. For instance, we would like to consider query *?- p(Z,Z),q(Z)* as a specialization of query *?- p(X,Y),p(Y,X)*, although $\{$*p(X,Y),p(Y,X)*$\}$ is not a subset of $\{$*p(Z,Z),q(Z)*$\}$.

The most obvious general-purpose definition of the subsumption relation is based on logical implication: $Query1 \preceq Query2$ if and only if $Query2 \models Query1$. Logical implication could detect for instance that *?- p(X,Y),p(Y,X)* is a generalization of *?- p(Z,Z),q(Z)*. However, due to the high computational cost of the logical implication check, inductive logic programming algorithms often rely on a stronger variant coined θ-subsumption by Plotkin (Plotkin, 1970). $Query1$ θ-subsumes $Query2$ if and only if there exists a (possibly empty) substitution of the variables of $Query2$, such that every atom of the resulting query occurs in $Query1$, i.e., $Query1 \supseteq Query2\theta$. For instance, *?- p(Z,Z),q(Z)* θ-subsumes *?- p(X,Y),p(Y,X)*, with $\theta = \{X/Z, Y/Z\}$.

3.2.2. Candidate evaluation Algorithm WARMR-EVAL adapts Definition 2 of frequency of a single query Q_j to the levelwise approach, which matches a set of patterns against one example at a time. The example is here represented by θ_k, the substitution for the key variables obtained in step (2) by running *?- key* against the database. The algorithm, in step (2.b), applies a fixed substitution θ_k to the subsequent queries Q_j drawn from $\mathcal{Q}$, and increments an associated counter q_j in case $Q_j\theta_k$ succeeds with respect to the database.

If we execute the latter evaluation with respect to $\mathbf{r}$, we still need one pass through the database per query, instead of one pass per level. The solution adopted in WARMR-EVAL, step (2.a), is based on the assumption that there exists a relatively small subset $\mathbf{r}_k$ of $\mathbf{r}$, such that the evaluation of any $Q\theta_k$ only involves tuples from $\mathbf{r}_k$. Readers familiar with relational database technology might notice a similar assumption underlies the definition of a cluster index. In many cases $\{\mathbf{r}_k\}$ is a partition on $\mathbf{r}$. The algorithm then makes a single pass through the data in the sense that the key values θ_k are retrieved one by one, the subsequent subdatabases $\mathbf{r}_k$ are activated once in (2.a), and all queries are evaluated locally with respect to $\mathbf{r}_k$ in (2.b). An experimental evaluation of this localisation of information in a related data mining task can be found in (Blockeel et al, 1998a).

Consider as an example the alarm analysis database introduced in the previous section. Each subset $\mathbf{r}_k$ of this database would contain a fact *window_id(*wid_k*)*$\leftarrow$ and zero or more facts *window(*wid_k*,atype)*$\leftarrow$. This subset of the database indeed suffices for solving queries $Q\theta_k$ built with predicates *window_id* and *window*. But what about the facts of the form *is_a(atype, parent_atype)*$\leftarrow$? Clearly these are relevant for many keys wid_k and involved in solving queries in many examples θ_k. As a consequence, they cannot be assigned to one $\mathbf{r}_k$ exclusively. We will discuss efficient solutions in Section 3.2.4 below.

3.2.3. Candidate generation To generate candidates, WARMR-GEN employs at step (2) a classical specialization operator under θ-subsumption (Plotkin, 1970, Muggleton and De Raedt, 1994). A specialization operator ρ maps queries $\in \mathcal{L}$ onto sets of queries $\in 2^{\mathcal{L}}$, such that for any $Query1$ and $\forall\, Query2 \in \rho(Query1)$, $Query_1$ θ-subsumes $Query_2$. The

Algorithm 2 *:* WARMR-EVAL

`Inputs:` Database **r**; set of queries $\mathcal{Q}$; WRMODE *key*
`Outputs:` The frequencies of queries $\mathcal{Q}$

1. For each query $Q_j \in \mathcal{Q}$, initialize frequency counter $q_j := 0$
2. For each substitution $\theta_k \in$ *answerset(?- key,* **r***)*, do the following:
   (a) Isolate the relevant fraction of the database $\mathbf{r}_k \subseteq \mathbf{r}$
   (b) For each query $Q_j \in \mathcal{Q}$, do the following:
   If query $Q_j\theta_k$ succeeds w.r.t. $\mathbf{r}_k$, increment counter q_j
3. For each query $Q; \in \mathcal{Q}$, return frequency counter q_j

operator used in WARMR-GEN essentially adds one atom to the query at a time, as allowed by WRMODE declarations. For instance, given the WRMODE declarations:

key = window_id(−w)
$Atoms =$ { *window(+w,atype*$_1$*), . . . , window(+w,atype*$_m$*)* ,
window(+w,−a), is_a(+a,aclass$_1$*), . . . , is_a(+a,aclass*$_n$*)*}

and

$Q_j =$ *?- window_id(W), window(W,atype*$_1$*),window(W,AT1)*

WARMR-GEN builds specializations Q'_j of Q_j by adding an atom from set

{*window(W,atype*$_i$*), window(W,AT2), is_a(AT1,aclass*$_j$*)*}

with $1 \leq i \leq m$ and $1 \leq j \leq n$.

Mode and type declarations on variables may cause an atom to be added for the first time only deep down the lattice. For instance, in the example above, atom *is_a(AT1,aclass*$_j$*)* could not have been added if *window(W,AT1)* had not been in Q. This complicates pruning significantly. We can no longer require that all generalizations of a candidate are frequent as some of the generalizations, such as *?- window_id(W), window(W,atype*$_1$*),is_a(AT1,aclass*$_1$*)* in the example, might simply not be in $\mathcal{L}$. Instead, WARMR-GEN at step (2.i) requires candidates not to θ-subsume any infrequent query. In step (2.ii), we also require that candidates and frequent queries are mutually inequivalent under θ-subsumption. This way a potentially huge set of redundant solutions is eliminated.

3.2.4. Efficiency and complexity considerations WARMR, like any other inductive logic programming algorithm, has to cope with the theoretical result that both evaluation of a query and testing θ-subsumption are NP complete problems. In some practical cases however, as discussed in respectively (De Raedt and Džeroski, 1994) and (Kietz and Lübbe, 1994), both problems can be solved efficiently. We now localize these critical operations in the WARMR algorithm and discuss some implemented and possible optimizations.

The composition and the loading of $\mathbf{r}_k$ in WARMR-GEN step (2.a) can be optimized in two ways. First, if a fixed portion $\mathbf{r}^B$ reoccurs as a subset of many $\mathbf{r}_k$'s, we can load

Algorithm 3 *:* WARMR-GEN

`Inputs:` WRMODE language $\mathcal{L}$; infrequent queries $\mathcal{I}$; frequent queries $\mathcal{F}$; frequent queries $\mathcal{Q}_d$ for level d
`Outputs:` Candidate queries $\mathcal{Q}_{d+1}$ for level $d+1$

1. Initialize $\mathcal{Q}_{d+1} := \emptyset$
2. For each query $Q_j \in \mathcal{Q}_d$, and for each immediate specialization $Q'_j \in \mathcal{L}$ of Q_j :
   Add Q'_j to $\mathcal{Q}_{d+1}$, unless:
   (i) Q'_j is more specific than some query $\in \mathcal{I}$, or
   (ii) Q'_j is equivalent to some query $\in \mathcal{Q}_{d+1} \cup \mathcal{F}$
3. Return $\mathcal{Q}_{\lceil} + 1$

the common $\mathbf{r}^B$ once, and iteratively load only the specific $\mathbf{r}_k \setminus \mathbf{r}^B$. In inductive logic programming jargon, $\mathbf{r}^B$ typically corresponds to background knowledge. For instance, in the telecommunication domain, background knowledge $\mathbf{r}^B$ might consist of: (1) ground facts about alarm types, network elements, and network topology, and (2) clausal rules that capture general fault and network management principles and so forth. For instance, the *is_a(atype, parent_type)*$\leftarrow$ facts in the example above could be stored in $\mathbf{r}^B$. Second, in cases where the repeated composition of $\mathbf{r}_k$ is still too costly, e.g. if many facts have to be selected from many different predicates, a preprocessing step can be considered where all the $\mathbf{r}_k$'s are composed once and written to flat files, see (Blockeel et al, 1998a) for an experimental evaluation.

In some cases, $\mathbf{r}_k$ is very small compared to $\mathbf{r}$ and can be loaded in main memory even if $\mathbf{r}$ cannot. This has the crucial advantage that evaluation of candidates in WARMR-EVAL step (2.b) can be done more efficiently with respect to a cached fraction of the database. It is possible however to contrive a database and language such that each $\mathbf{r}_k \simeq \mathbf{r}$ and the evaluation of complex queries with respect to huge databases becomes impractical. Consider for instance the consequences of adding to the alarm analysis database facts of the form *follows(*wid_i*,*wid_j*)*$\leftarrow$ and allowing queries such as *?- window_id(W1), follows(W1,W2), window(W2,1001)* i.e., "the window follows a window that contains an alarm type *1001*". To solve such queries in an example θ_k we need all the facts *window(*wid_j*)*$\leftarrow$ with $j \leq k$. If we further add predicate *precedes*, $\mathbf{r}_k$ becomes roughly equal to $\mathbf{r}$. This example may stretch the notion of a *window* on the data, but it does illustrate that the isolation of $\mathbf{r}_k$ from $\mathbf{r}$ and "local" evaluation of $Q_j\theta$ in WARMR-EVAL step (2.b) is not guaranteed to be profitable. This approach does allow however to take advantage of situations where the bulk of the database is immaterial to that evaluation.

Some alternative strategies to boost WARMR-EVAL step (2.b) will be considered in future work. For instance, when in query *?- window_id(*wid_k*), window(*wid_k*,AType), is_a(AType,switch), window(*wid_k*,1001)* the atom *window(*wid_k*,1001)* fails, there is no point in looking for alternative bindings for *AType*. In PROLOG terminology, the *cut* operator

should be inserted after *is_a(AType,switch)* to suppress backtracking. Another possibility to reduce backtracking would be to reorganize set $\mathcal{Q}$ into some tree-like structure, similar to APRIORI's hash trees, and evaluate the queries collectively against the data.

To prune candidates Q'_j, WARMR-GEN in steps (2.i) and (2.ii) scans $\mathcal{I}$, $\mathcal{F}$, and $\mathcal{Q}_{d+1}$ until a query is found that is θ-subsumed by (and θ-subsumes, in the case of (2.ii)) Q'_j. As a straightforward optimization, a sorted list of predicates is associated with each query, and the expensive θ-subsumption test on a couple of queries is only applied after a positive subset test on the corresponding predicate lists. Planned improvements include the reorganization of the massively overlapping queries from $\mathcal{I}$, $\mathcal{F}$, and $\mathcal{Q}_{d+1}$ into a tree, as above, and verification of θ-subsumption against this structure.

One can also alleviate the candidate generation problem by using a declarative language bias formalism that is equipped with a refinement operator that is optimal in the sense that it generates each query at most once, as done, for instance, in (Dehaspe and De Raedt, 1996, Dehaspe and De Raedt, 1997, Weber, 1997, Weber, 1998, Wrobel, 1997).

3.3. Two-phased discovery of frequent and confident rules

Frequent patterns are commonly not considered useful for presentation to the user as such. Their popularity is mainly based on the fact that they can be efficiently post-processed into rules that exceed given confidence and frequency threshold values. The best known example of this two-phased strategy is the discovery of association rules (Agrawal et al, 1993), and closely related patterns include episodes (Mannila et al., 1997) and sequential patterns (Agrawal and Srikant, 1995). For all these patterns, the threshold values offer a natural way of pruning weak and rare rules.

We introduce the notion *query extension* to refer to the first-order equivalent of an association rule. In terms of the DATALOG concepts introduced in Section 2, a *query extension* E is an expression of the form $A_1, \ldots, A_k \Rightarrow A_{k+1}, \ldots, A_n$, where A_i are atoms. This formula should be read as "if query *?-* A_1,...,A_k succeeds then the extended query *?-* A_1,...,A_n succeeds also". The *confidence* of query extension E can be computed as the ratio of the frequencies of queries *?-* A_1,...,A_n and *?-* A_1,...,A_k. The *frequency* (or *support*) of query extension E is the frequency of query *?-* A_1,...,A_n.

As observed in (Agrawal et al, 1993) for association rules, confident and frequent query extensions can be found effectively in two steps. In the first step one determines the set of all frequent queries *?-* A_1,...,A_n, and in the second produces query extensions $A_1, \ldots, A_k \Rightarrow A_{k+1}, \ldots, A_n$ whose confidence exceeds the given threshold. If all frequent queries and their frequencies are known as a result of the first step, then this easy second step is guaranteed to output all frequent and confident query extensions.

EXAMPLE: Suppose we run WARMR on the alarm analysis database introduced in Section 2.2, with $\mathcal{L}$ defined by the WRMODE declarations at the end of Section 2.3, and obtain the two following queries with associated relative frequencies:

?- window_id(W),window(W,1001)	FREQ: 0.4
?- window_id(W),window(W,1001),window(W,A),is_a(A,switch)	FREQ: 0.3

These patterns can be processed without going back to the database into a query extension:

window_id(W),window(W,1001) ⇒ *window(W,A),is_a(A,switch)*
FREQ:0.3 ; CONF:0.75

i.e., with 30% frequency and 75% confidence:"**if** there is in the window an alarm of type *1001*, **then** the window will also contain an alarm that belongs to the class of *switch* alarms".

□

4. Cases of frequent query discovery

We now present more extensive problems in the discovery of frequent patterns in the domain of alarm analysis. These cases are inspired by previous data mining settings, but they contain elements that have not been considered before. We show how minor modifications to the language bias define increasingly complex tasks. We then briefly compare different settings and algorithms for solving them.

4.1. More complex parallel episodes

Consider the case where each occurrence of an alarm has a number of attributes associated with it. In the telecommunications domain such attributes include, in addition to the alarm type, e.g., the urgency and the sender of the alarm. The alarm database thus consists of a number of windows, each with a number of alarms, where each alarm has certain individual properties. Discovery of frequent combinations of alarms where the properties of alarms are also considered, is an interesting problem. However, mapping such a case to one of the existing frequent pattern discovery problems does not seem to be possible without loss of information.

Assume, for instance, an alarm database with facts *win_id(wid)*← of window identifiers, as before, and facts *win(wid, atype, urgency, sender)*← that denote for each alarm in a window the type, the urgency level, and the sending network element of the alarm. A sample of such a database is shown below. This setting has some similarity with the "multiple-instance problem" known from attribute-value learning (Dietterich et al, 1997).

win(1,1001,notice,367)←	*win(2,1001,notice,534)*←
win(1,1054,warning,367)←	*win(2,1005,notice,245)*←
...	...

Let us now look at two possible strategies for transforming this setting to simple sets of binary indicators, such as alarm types or items in the more simple examples considered in Section 2. First, one could blow up the number of indicators and introduce an item $atype\_urgency\_sender$ for all combinations that occur, e.g., the first fact shown above would be written as *win(1,1001_notice_367)*←. A first objection to this solution is that, especially with a high number of (many-valued) properties, this transformation will result in an exponential number of infrequent items. Moreover, even if this transformation is practicable, it would disallow the discovery of frequent combinations of the original properties.

As as second attempt we could add the individual properties as extra item types, as is done with item hierarchies and in (Klemettinen et al, 1998). This, indeed, allows the discovery of patterns such as

?- win_id(Wid),win(Wid,1001),win(Wid,warning),win(Wid,534)

i.e., "an alarm of type *1001*, a *warning*, and an alarm from *534*". However, we lose the facility to discover something about combinations of properties, such as "a *warning* of type *1001* from *534*": properties *warning* and *534* are tested independently and cannot be linked to the same alarm.

To summarize the problem, we would like alarms and their properties to occur both in isolation, and in any combination, e.g.

?- win_id(W),win(W,1001,notice,367),win(W,1054,warning,S),win(W,1034,U, T)

"a *notice* of type *1001* from *367*, a *warning* of type *1054*, and an alarm of type *1034*".

With WARMR such patterns could be discovered by choosing the language bias essentially as follows:

key = win_id(−w)
$Atoms$ = { *win(+w, −atype, −urgency, −sender),*
eq(+atype, 1001), eq(+atype, 1002), . . . ,
eq(+urgency, notice), eq(+urgency, warning), . . . ,
eq(+sender, 245), eq(+sender, 265), . . . }

where eq is an equality test. The corresponding language contains the pattern above in format:

?- win_id(Wid), win(Wid,A1,U1,S1), eq(A1,1001), eq(U1,notice), eq(S1,367),
win(Wid,A2,U2,S2), eq(A2,1054), eq(U2,warning),
win(Wid,A3,U3,S3), eq(A3,1034)

This bias could be easily extended to handle the case where a window as a whole may have properties in addition to the properties of alarms. For instance, whether the window falls on office time or not is a useful bit of information. One can then look for combinations of alarm and window properties, such as

?- win_id(Wid),win(Wid,1001,warning,S),office_hours(Wid,yes)

"*office time* windows containing a *warning* of type *1001*". To find such patterns with WARMR, we only have to add

$Atoms := Atoms \cup \{$*office_hours(+w,yes), office_hours(+w,no)*$\}$

to the language bias. Such an extension could be very interesting in the supermarket basket analysis domain, where the properties of a transaction may contain information about the context of shopping, about the customer, or aggregate information about the basket, such as the time or the location of shopping, or the total value or the number of items in the basket. Frequent patterns that could be discovered include, e.g., "baskets containing cigarettes and paid in cash" and "senior customers buying something promoted".

Allowing the presence of arbitrary relations between alarm properties or arbitrary background knowledge makes the setting even more interesting. For example, we can slightly modify one of the patterns shown above:

?- win(W,1001,notice,367),win(W,1054,warning,S),win(W,1034,U,S)

"a *notice 1001* from *367*, and a *warning 1054* and an alarm *1034 from the same sender*". Notice the shared S variable—such patterns where the sender is shared can be found if WARMR's bias is extended with

$Atoms := Atoms \cup$ *win(+w, −atype, −urgency, +sender)*

All the facts mentioned above are relevant for a single window win_k only and can be stored in mutually exclusive $\mathbf{r}_k$ subdatabases to improve query evaluation (cf. Section 3.2.2). Common background knowledge $\mathbf{r}^B$ can be used to specify, e.g, neighborhood relations in the network of telecommunication equipment. The database and language then look as follows:

$\mathbf{r} := \mathbf{r} \cup \{$*neighbor(245,265)*$\leftarrow, \ldots\}$
$Atoms := Atoms \cup \{$*neighbor(+sender,+sender)*$\}$

An example of a query admitted by this extension is:

?- win_id(W),win(W,1001,notice,S1),win(W,1054,warning,S2),neighbor(S1,S2)

This formulation is close to the one of general episodes given in (Mannila and Toivonen, 1996). They allow binary relations on alarms, such as for instance the *neighbor* relation above, or an order relation based on a time stamp associated with the alarms. Their setting has not been implemented in a full scale before.

4.2. Dimensions of the frequent pattern discovery task

Different frequent pattern discovery tasks can be characterized in terms of their support for a fairly small number of features. In Table 2 we present an overview of different tasks. Since most of the work has been presented in the context of association rules, we use primarily terms from association rules and market basket analysis (and secondarily the alarm vocabulary).

"Itemsets" (IS) stands for the discovery of frequent sets of items, as it is done for the discovery of association rules in the very basic setting (Agrawal et al, 1993) (or parallel episodes of Section 2, without a class hierarchy on alarm types (Mannila et al., 1997)). "Item hierarchies" (IH) is the basic setting extended with a hierarchy on the items (Han and Fu, 1995, Holsheimer et al, 1995, Srikant and Agrawal, 1995) (or the case described as parallel episodes in the examples of Section 2). "Sequential patterns" (SP) refers to the case in basket analysis where a number of transactions are observed for each customer, and patterns relating items in different transactions are searched for (Agrawal and Srikant, 1995, Srikant and Agrawal, 1996). "Parallel episodes with alarm properties" (PE) stands for the more complex parallel episodes described in this section: items in a transaction have individual properties (individual occurrences of alarms have properties). "General episodes" (GE) is the setting where items have properties and patterns contain unary and binary relations on items within a transaction (Mannila and Toivonen, 1996). Finally, "DATALOG patterns" (DP) stands for the possibilities of frequent DATALOG queries.

Table 2. Dimensions of frequent pattern types. Legend: IS = itemsets, IH = itemsets with item hierarchies, SP = sequential patterns, PE = parallel episodes with alarm properties, GE = general episodes, DP = (full) DATALOG patterns.

	IS	IH	SP	PE	GE	DP
Many items per transaction	+	+	+	+	+	+
Item type properties		+	+	+	+	+
Many (ordered) transactions per example			+			+
Item instance and transaction properties				+	+	+
Binary item properties (besides order)					+	+
Arbitrary DATALOG queries						+

Table 3. Dimensions of pattern discovery algorithms. Legend: IS = itemsets, IH = itemsets with item hierarchies, SP = sequential patterns, PE = parallel episodes with alarm properties, GE = general episodes, DP = WARMR.

	IS	IH	SP	PE	GE	DP
Levelwise search	+	+	+	+	+	+
Bindings can be stored	+	+	+	+	+	
All backtracking suppressed	+	+				
Subset relation between item types only	+					
Incremental candidate evaluation				+	+	

The table lists six of the properties where the tasks differ. These properties are directly reflected by the existence of different types of atoms in the language $\mathcal{L}$. A cell contains a plus if the pattern type can deal or can easily be extended to deal with the given feature. Note that the table is coarse: for instance, "item type properties" means a concept hierarchy for most of the cases, and only some can handle other properties associated with item types.

According to Table 2, the most obvious gaps to fill are to either extend sequential patterns to include item and transaction attributes and binary properties, or to extend episodes to the case where there is another level of containment between items and examples (e.g., sets of alarms are sent as transactions, which then occur in windows). Finally, recall that episodes have not been implemented before in the extent described here.

4.3. Dimensions of the pattern discovery algorithms

We conclude this section with a summary of those dimensions that characterize and relate the different pattern discovery algorithms. Table 3 shares column labels with Table2: here a plus in a cell means that a specialized algorithm for the column can exploit the feature marked on the row.

All algorithms can use the levelwise search method. In all settings except WARMR, the use of variables is strongly limited, e.g., only to the window or transaction variable. As an effect, the management of variable bindings is very efficient and often the bindings can even be stored for later use with other patterns. The use of variables also affects the efficiency of the recognition of patterns. In some settings, the search can be organized so that there is essentially no backtracking within patterns. Some algorithms exploit the fact that their queries can be mapped to simple cases, in particular to testing the subset relation, which is

efficient when compared to θ-subsumption in the general case. This has an effect both on candidate generation and testing. Episode algorithms can take additional advantage of the fact that the sliding of the window can be handled in an incremental manner.

The relevant—though not very surprising—observation here is that Table 3 is roughly complementary with Table 2: columns with many plusses in one table tend have few plusses in the other. Thus, the combination of these two tables provides a fairly balanced picture of the obvious trade-off between expressivity and efficiency in the context of frequent pattern mining. It also demonstrates there is no dichotomy item sets vs. queries (APRIORI vs. WARMR), but rather a gradual and complex change in the trade-off between expressivity and efficiency, with a number of "intermediate" problems that have received considerable attention.

WARMR is a generic algorithm that does not take advantage of any special properties of particular problem instances. Therefore, for any specific setting, a more efficient algorithm can probably be devised. WARMR is useful, in particular, for exploring different settings, both existing and novel ones.

Finally, the two tables provide a blueprint for a single integrated system that uses Table 2 to determine the minimal level of expressivity required and Table 3 to fire the maximally efficient algorithm available within that setting. In such a system, WARMR would be the "catch-all" method.

5. Experiments

In this section we present experimental results with WARMR in the task of frequent query discovery. First, we round off the running example on alarm analysis. The actual inputs and a sample of the outputs are discussed in more detail. We then move on to a case where usefulness of the patterns discovered by WARMR has been confirmed by expert evaluation (Dehaspe et al, 1998). The task there is to identify substructures of chemical compounds that have a potential for inducing cancer in human beings.

5.1. Alarm analysis

The experimental data originates from a fault management database of a mobile communication network. The problem of discovering recurrent combinations of alarms from such databases has been considered in (Goodman and Latin, 1991, Hätönen et al., 1996, Klemettinen et al, 1998, Mannila et al., 1997). Closely related data mining problems have been considered, e.g., in (Bettini et al, 1996, Dousson et al, 1993, Morris et al, 1995, Oates and Cohen, 1996, Sasisekharan et al, 1996, Srikant and Agrawal, 1996, Padmanabhan and Tuzhilin, 1996, Wang et al, 1994).

The dataset consists of a sequence of 46662 alarms emitted by the network elements such as base stations and transmission devices during a period of one month. The time granularity of the data is one second. The average frequency of alarms is approximately 1500 alarms/day, or 1 alarm/minute, but since alarms tend to occur in bursts, the busiest second contains 50 alarms.

There are 180 different alarm types, which can be further classified into 10 overlapping classes. Each instance of an alarm has one of 4 urgency levels. The alarms in the dataset

have been received from 2012 network management objects of 9 different types. These objects represent units of different granularities, and they form a containment hierarchy. This hierarchy gives essential information about the nature of the relationships of the objects.

The discovery task we consider is to find those combinations of alarms that are frequent. This problem is the one considered in episode discovery, but here, to the best of our knowledge, we implement a much more expressive variation than has been done before. We consider alarms with different combinations of properties, and we also consider cases where the alarms are connected, e.g., in the object hierarchy or in some other way. We cannot see any way of transforming this task to episode or sequential pattern discovery task without losing information.

5.1.1. Database and background knowledge Following are some of the most important predicates used to represent the alarm data. The most obvious ones, such as *alarm_type(alarm, alarmtype)*←, relate each alarm instance to an alarm type, an occurrence time, several alarm classes, etc. A background predicate *precedes(alarm1, alarm2)* allows temporal order tests between alarms *alarm1* and *alarm2*. Some new clauses are defined in the background knowledge based on the occurrence time, to add potentially useful information such as *officehour(alarm)*←.

In a similar manner, the database contains clauses *sender(alarm, object)*← that indicate the sender of each alarm; background knowledge includes predicates such as *ancestor(object, ancestor)*, *sibling(object1, object2)*, and *same_object_type(object1, object2)*.

In the experiments we considered windows of width 120 seconds that start from an alarm. To represent and define windowing, we specified in the background knowledge a fact *win_id(wid)*← and a fact *start(wid,alarm)*← to identify the window and the alarm it starts from. Finally, we added a clause that derives *in_window(wid, interval, alarm)* if *alarm* occurs within time *interval* from the start of window *wid*.

5.1.2. Language bias The WRMODE language bias used in the experiments looks essentially as follows:

key = *win_id(−w)*
$Atoms$ =
{*in_window(+w,120,−a),start(+w,±a),sender(+a,−o), precedes(+a,+a),*
officehour(+a), weekend(+a), weekday(+a,mon), . . . , weekday(+a,fri),
urgency(+a,1),. . . ,urgency(+a,4), same_urgency(+a,+a),
alarm_type(+a,1001),. . . ,alarm_type(+a,2270), same_alarm_type(+a,+a),
alarm_class(+a,sw),. . . ,alarm_class(+a,tr), same_alarm_class(+a,+a),
sender_elem(+a,95),. . . ,sender_elem(+a,314), same_sender_elem(+a,+a),
object_type(+o,bcf),. . . ,object_type(+o,brx), same_object_type(+o,+o),
ancestor(+o,+o),. . . ,sibling(+o,+o) }

5.1.3. Results A specific task we considered was to describe the windows following alarms of a specific alarm type. Problems reported by this particular alarm are difficult to track; here the goal is to discover patterns of alarms from related objects that might help in

explaining the important alarms. We present one of the patterns, in the more informative query extension format:

win_id(W),start(W,A),in_window(W,120,B),alarm_class(B,bsc_message) $\Rightarrow$
in_window(W,120,C),alarm_class(C,trans),same_urgency(A,C),same_urgency(C,B)
FREQ:0.15 ; CONF:0.77

i.e., with 15% frequency and 77% confidence: "**if** a window *starts* with an alarm A and contains an alarm B of class *bsc-message* **then** it will also contain an alarm C of class *trans* such that all alarms referred to will have the *same urgency*".

In a more general setting, where the window could start on alarms of any type, the following pattern was discovered:

win_id(W),in_window(W,120,A),sender(A,O),object_type(O,bcf),ancestor(O,P),
object_type(P,bsc),in_window(W,120,B),alarm_class(B,bst_message) $\Rightarrow$
precedes(A,B),urgency(A,2),alarm_class(A,bst_message)
FREQ:0.27 ; CONF:0.68

i.e., with 68% confidence, and 27% frequency: "**if** there is in the window an alarm sent by an object of type *bcf*, and an ancestor of that object is of type *bsc*, and there is also an alarm of class *bst_message*, **then** the first alarm precedes the second one and has urgency 2 and class *bst_message*".

Observe that the natural language paraphrase is actually not exactly equivalent to the rule. A query extension $X \Rightarrow Y$ should be read as "**if** query *?- X* succeeds **then** query *?- X,Y* succeeds" (cf. Section 3.3). Within a window, the condition part might be met for a number of substitutions of alarms (A, B), whereas only some of them may meet the conclusion part. A more exact English formulation would have the form "if the condition part holds (with some substitution of the variables), then the condition and conclusion parts hold together (with possibly some other substitution)". The simpler formulation "**if** X succeeds **then** Y succeeds" is true if the intersection of condition and conclusion variables is empty, which is obviously the case with traditional association rules, or if that intersection only contains variables that can be bound in at most one way.

There are a large number of factors that affect the sending of alarms, such as the object relationships, network configuration and also the surrounding environment. Representing these factors and taking them into account in the pattern discovery is much easier with WARMR than with the previous episode formalisms. Network management experts have also found the pattern types discovered with WARMR more informative and useful.

5.2. Predictive toxicology

The goal of this second experiment is to discover frequent substructures of chemical compounds in relation to their possible carcinogenicity. A few raw statistics confirm this task is of clear scientific and medical interest: in western countries cancer is the second most common cause of death, one third of the population will get cancer, and one fourth of the population will die of cancer. An estimated 80% of these cancers are linked to environmental factors such as exposal to carcinogenic chemicals. At the same time, only fraction of chemicals are tested for carcinogenesis. This contradiction can be explained by the fact

that current methods are expensive and time consuming, hence the interest in cheaper and faster computer based methods.

The National Toxicology Program (NTP) of the U.S. National Institute for Environmental Health Sciences aims at safety testing of new chemicals (500 – 1000 every year) and identification of hazardous chemicals in use (nearly 100,000). Given a compound, they perform a range of tests that vary in expense, speed and accuracy to estimate the carcinogenic effect of the compound on humans. At the extreme cheap, fast, and relatively inaccurate end are biological tests that use bacteria. At the other end are long (about two years), expensive, and relatively reliable standardized bioassays on thousands of rodents. The urgent need for predictive toxicology models that identify hazardous chemical exposures more rapidly and at lower cost than current procedures is the driving force behind the Predictive Toxicology Evaluation (PTE) project (Bristol et al, 1996).

Within PTE they have collected and published a database of about 300 classified NTP chemical carcinogenesis bioassays, and a collection of 30 chemicals whose tests are to be completed by the end of 1998. The prediction of rodent chemical carcinogenesis of these compound was launched at IJCAI'97 as a research challenge for artificial intelligence (Srinisavan et al, 1997). Rather than competing with expert chemists in classifying chemicals to carcinogenic or otherwise, our goal was to discover frequent patterns that would aid chemists – and data miners seeking predictive theories – to identify useful substructures for carcinogenicity research, and so contribute to the scientific insight. This can be contrasted with previous machine learning research in this application, which has mainly concentrated on predicting the toxicity of unknown chemicals (Srinisavan et al, 1997, Kramer et al, 1997). We believe that a repository of frequent substructures and their frequencies would be valuable for chemical (machine learning) research. For example, once we know *all* frequent substructures, we can make stronger claims about the (non-)existence of high quality single rules than can usually be done with classifying approaches based on heuristic search.

The results of this experiment have been previously published in (Dehaspe et al, 1998). Related problems in structure discovery in molecular biology have been considered, e.g., in (Wang et al, 1997, Kramer et al, 1997, King et al, 1996, King and Srinivasan, 1996). Substructure discovery and the utilization of background knowledge have been discussed in (Djoko et al, 1995). Closely related data mining problems have recently arisen also in schema discovery in semi-structured data (Wang and Liu, 1997).

5.2.1. Database and background knowledge The database for the carcinogenesis problem was taken from *http://www.comlab.ox.ac.uk/oucl/groups/machlearn/PTE/*. The set we have used contains 337 compounds, 182 (54%) of which have been classified as carcinogenic and the remaining 155 (46%) otherwise.

Each compound is basically described as a set of atoms and their bond connectivities, as proposed in (King et al, 1996). The atoms of a compound are represented as DATALOG facts such as *atom(d1,d1_25,h,1,0.327)*← stating that compound *d1* contains atom *d1_25* of element *h* and type *1* with partial charge *0.327*. For convenience, we have defined additional view predicates *atomel*, *atomty*, and *atomch*; e.g., *atomel(d1,d1_25,h)*←. Bonds between atoms are defined with facts such as *bond(d1,d1_24,d1_25,1)*←, meaning that in compound *d1* there is a bond between atoms *d1_24* and *d1_25*, and the bond is of type *1*. There are

roughly 18500 of these atom/bond facts to represent the basic structure of the compounds. Notice we can define a partition $\{\mathbf{r}_k\}$ on these facts such that evaluation of candidates Q_i is done (relatively) efficiently with respect to a single compound cid_k at a time (cf. Section 3.2.2). For each member $\mathbf{r}_k$ of the partition, i.e., each compound, we have added a fact *comp_id(*cid_k*)*$\leftarrow$.

In addition, background knowledge contains around 7000 facts and some short DATALOG programs to define mutagenic compounds, genotoxicity properties of compounds, generic structural groups such as alcohols, connections between such chemical groups, tests to verify whether an atom is part of a chemical group, and a family of structural alerts called *Ashby* alerts (Ashby and Tennant, 1991).

We randomly split the set of 337 compounds into 2/3 for the discovery of frequent substructures, and 1/3 for the validation of derived query extensions about carcinogenicity.

5.2.2. Language bias The most extensive set of language bias WRMODE specifications used in the experiments is:

key = *comp_id(−c)*
$Atoms$ =
{*atomel(+c,±a,c),. . . ,atomel(+c,±a,h), atomty(+c,±a,1),. . . ,atomty(+c,±a,75),*
bond(+c,+a,+a,±btype),eq(+btype,1),. . . ,eq(+btype,7),
carcinogenic(+c),non_carcinogenic(+c),ames(+c),
has_property(+c,salmonella,p),. . . ,has_property(+c,chromex,n),
alcohol(+c,±struct),methyl(+c,±struct),. . . ,six_ring(+c,±struct),
ashby_alert(+c,cyanide,±struct),. . . ,ashby_alert(+c,methanol,±struct),
connected(+c,+struct,+struct),occurs_in(+c,+a,+struct)}

5.2.3. Results In order to investigate the usefulness of different types of information in the biochemical database, WARMR's language bias was varied to produce three sets of frequent patterns.

- *Experiment 1:* only atom element, atom type, and bond information. At level 6, WARMR generates substructure

  ?- atomel(C,A1,c), bond(C,A1,A2,BT), atomel(C,A2,c), atomty(C,A2,10),
  atomel(C,A3,h), bond(C,A2,A3,BT) FREQ:0.57

  i.e., "a carbon atom bound to a carbon atom of type 10 bound to a hydrogen atom, where the two bonds are of the same bond type".

- *Experiment 2:* everything except the atom/bond information. An example of a substructure discovered at level 4 is

  ?- six_ring(C,S1), alcohol(C,S2), ashby_alert(C,di10,S3), connected(S1,S3)
  FREQ:0.05

  i.e., "an alcohol and a six ring connected to a structure with Ashby alert di10".

- *Experiment 3:* the full database, except the Ashby alerts. At level 5, WARMR produces substructure

  ?- six_ring(C,S), atomel(C,A1,h), atomel(C,A2,c), bond(C,A1,A2,X), occurs_in(A2,S) FREQ:0.70

  i.e., "a hydrogen atom bound to a carbon atom in a six ring".

5.2.4. Query extensions As described in Section 3.3, our repository of frequent substructures can be exploited directly, i.e., without going back to the database, to produce query extensions about carcinogenicity. For instance, we can combine

?- cytogen_ca(C,n), sulfide(C,S) FREQ:0.07

and

?- non_carcinogenic(C), cytogen_ca(C,n), sulfide(C,S) FREQ:0.06

to generate the query extension

cytogen_ca(C,n), sulfide(C,S) $\Rightarrow$ *non_carcinogenic(C)* FREQ:0.06 ; CONF:0.86

To rank these rules we have applied a simple binomial test that verifies how "surprising" the confidence of query extension *substructure(C)*$\Rightarrow$*(non_)carcinogenic(C)* is, i.e., how much it deviates from the confidence of *true*$\Rightarrow$*(non_)carcinogenic(C)*. All rules with significance below $3 * \sigma$ were discarded, with σ an estimation of the standard deviation. For instance, the significance level of the above rule is $3.16 * \sigma$. The 215 rules that passed this test were further annotated with their significance level on the 1/3 validation set, and finally combined with human domain expertise provided by Ross Donald King (Dehaspe et al, 1998). The main findings are summarized below.

5.2.5. Discussion In Experiment 1, only using atom-bond information, no substructure described with less than 7 logical atoms is found to be related to carcinogenicity. This places a lower limit on the complexity of rules that are based exclusively on chemical structure.

For Experiments 2 and 3, validation on an independent test set showed that the rules identified as interesting in the training set were clearly useful in prediction: the estimated accuracies of the rules from the training data were optimistically biased, as expected.

The rules found in Experiments 2 and 3 are dominated by biological tests for carcinogenicity. It is very interesting that these tests appear broadly independent of each other, so that if a chemical is identified as a possible carcinogen by several of these tests, it is possible to predict with high probability that it is a carcinogen; unfortunately, such compounds are rare.

Inspection of the rules from Experiment 2 revealed that the Ashby alerts were not used by any rules. We believe this reflects the difficulty humans and machine have in discovering general chemical substructures associated with carcinogenicity. However, it is possible that the intuitive alerts used by Ashby were incorrectly interpreted and encoded in PROLOG by (King and Srinivasan, 1996).

Inspection of the rules from Experiment 3 revealed no interesting substantial chemical substructures (atoms connected by bonds) in the rules found.

Two particularly interesting rules that combine biological tests with chemical attributes were found. It is difficult to compare these directly with existing knowledge, because most work on identifying structural alerts has been based on alerts for carcinogenicity, while both rules identify alerts for non-carcinogenicity. It is reasonable to search for non-carcinogenicity alerts as there can be specific chemical mechanisms for this, e.g. cytochromes specifically neutralize harmful chemicals. The rule *?- cytogen_ca(C,n), ring(sulfide,A,B)* for identifying non-carcinogenic compounds is interesting. The combination of conditions in the rule seems to be crucial: the cytogen and sulfide tests in isolation seem to do worse. Within rule *?- atomch(C,A,X), X* $\leq$ *-0.215, salmonella(C,n)* the addition of the chemical test makes the biological test more accurate at the expense of less coverage. As the rule refers to charge this rule may be connected to transport across cell membranes.

It is interesting and significant that no atom-bond substructures described with less than 7 conditions were found to be related to carcinogenicity. This result is not inconsistent with the results obtained by (King and Srinivasan, 1996) and (Srinivasan et al, 1997) using PROGOL because most of the substructures there involve partial charges, and the rest do not meet the coverage requirements in Experiment 1.

Although the lack of significant atom-bond substructures found in Experiment 1 is disappointing, it is perhaps not too surprising. The causation of chemical carcinogenesis is highly complex with many separate mechanisms involved. Therefore predicting carcinogenicity differs from standard drug design problems, where there is normally only a single well defined mechanism. We consider that it is probable that the current database is not yet large enough to provide the necessary statistical evidence required to easily identify chemical mechanisms. Biological tests avoid this problem because they detect multiple molecular mechanisms; e.g., the Ames test for mutagenesis detects many different ways chemicals can interact with DNA and cause mutations; biological tests also detect whether the compound can cross cell membranes and not be destroyed before reaching DNA.

The ultimate goal of the work in predictive toxicology is to produce a program that can predict carcinogenicity *in humans* from just input chemical structure. Such a system would allow chemicals to be quickly and cheaply tested without harm to any animals. This goal is still distant. Our results suggest that an intermediate goal for data mining in this predictive toxicology problem is to identify the combinations of biological tests and chemical substructures that provides the most cost-effective tests for testing chemical carcinogenesis.

6. Related work

We restrict the discussion of related work to research not explicitly addressed elsewhere in the paper. For an overview of inductive logic programming work in the context of knowledge discovery in databases, we refer to (Džeroski, 1996).

6.1. Logical paradigm: learning from interpretations

The definition of frequent query discovery and the relatively efficient candidate evaluation in WARMR is rooted in the *learning from interpretations* paradigm, introduced by De Raedt and Džeroski (De Raedt and Džeroski, 1994) and related to other inductive logic programming settings in (De Raedt, 1996). Indeed, subset $\{\mathbf{r}_k\}$ of database $\mathbf{r}$ (see Section 3.2.2) can be formalized in first-order logic as a *Herbrand interpretation*. Every $\mathbf{r}_k$ in which a query succeeds is then a *Herbrand model* of that query.

The learning from interpretations paradigm has proven to be particularly suitable for the design of upgrades to popular attribute-value learning techniques. In that respect, APRIORI - WARMR is only one of the more recent additions to a sequence of similar upgrades (De Raedt et al, 1998): EXPLORA(Klösgen, 1996)–CLAUDIEN (De Raedt and Dehaspe, 1997), CN2 (Clark and Niblett, 1989)–ICL (De Raedt and Van Laer, 1995), C4.5 (Quinlan, 1986)–TILDE (Blockeel and De Raedt, 1998, Blockeel et al, 1998a), hierarchical clustering (Langley, 1996)-(Blockeel et al, 1998b), and reinforcement learning (Džeroski et al, 1998).

6.2. Clausal discovery

WARMR is the first algorithm that addresses the frequent query discovery task, but the query extensions derived from its output (see Section 3.3) can also be obtained via algorithms that search for good quality rules directly. The discovery of clauses is handled for instance by KNOWLEDGE MINER (Shen et al, 1996), CLAUDIEN (De Raedt and Dehaspe, 1997), MIDOS (Wrobel, 1997), RDT (Kietz and Wrobel, 1992), MOBAL (Lindner and Morik, 1995), LAUREL (Weber, 1997, Weber, 1998), and PROGOL in learning from positives only mode (Muggleton, 1996). We first clarify the link between query extensions and clauses and next relate WARMR to clausal discovery algorithms.

The frequency of a clause $H \leftarrow B$, where head H is a disjunction and body B a conjunction of atoms, can be defined in our framework as $\mathit{frc}\,(H \leftarrow B, \mathbf{r}, \mathit{key}) =$

$$\frac{|\{\theta_k \in \mathit{answerset}\,(?\text{- } \mathit{key}, \mathbf{r}) \mid (?\text{- } B)\theta_k \text{ succeeds and } (?\text{- } B, \neg H)\theta_k \text{ fails w.r.t. } \mathbf{r}\}|}{|\{\theta_k \in \mathit{answerset}\,(?\text{- } \mathit{key}, \mathbf{r})\}}$$

i.e., the fraction of substitutions of the key variables with which the body of the clause is true while there is no way to make the body true and the head false, or, the fraction of examples in which the clause non-trivially holds (cf. *global coverage* in CLAUDIEN). The confidence of a clause $H \leftarrow B$ can then be defined as the frequency of the whole clause $\mathit{frc}\,(H \leftarrow B, \mathbf{r}, \mathit{key})$ divided by the frequency of the body $\mathit{frq}\,(?\text{- } B, \mathbf{r}, \mathit{key})$ (cf. *global accuracy* in CLAUDIEN). By definition of *frq* and *frc* the following properties hold:

$$\mathit{frq}\,((?\text{- } \neg B), \mathbf{r}, \mathit{key}) + \mathit{frq}\,((?\text{- } B, \neg H), \mathbf{r}, \mathit{key}) + \mathit{frc}\,(H \leftarrow B, \mathbf{r}, \mathit{key}) = 1$$
$$\mathit{confidence}(B \Rightarrow H) + \mathit{confidence}(\neg H \leftarrow B) = 1$$

These properties allow us to translate query extensions $B \Rightarrow H$ to clauses $\neg H \leftarrow B$ and back. For a more detailed account of the relation between clauses and query extensions, we refer to (Dehaspe, 1998).

The key differences between WARMR and clausal discovery algorithms have to do with the logical expressions in the respective search spaces: queries vs. clauses. The efficient levelwise method for traversing the search space is not directly applicable to clausal discovery. For levelwise search, a quality criterion is required that is monotone with respect to the specialization relation, cf. (Mannila and Toivonen, 1997). In a space of queries, frequency is such a monotone quality criterion. In a space of clauses, neither confidence nor frequency qualifies directly, and pruning is often based on the frequency of clause bodies, which is again monotone w.r.t. generality. Moreover, clausal discovery engines often have an any-time character and typically incorporate heuristics to direct the search immediately to regions where highly confident and frequent rules can be expected. In that sense clausal discovery engines are complementary to WARMR, which performs a more exhaustive breadth first search for frequent queries, and only in a post-processing step can discover query extensions that meet both the frequency and the confidence standards.

7. Conclusions

We have presented a general DATALOG formulation of the frequent pattern discovery problem: given a set $\mathcal{L}$ of DATALOG queries, find out which queries succeed frequently in a given database. We outlined WRMODE, a declarative formalism for specifying the language bias, i.e., the search space $\mathcal{L}$ of admissible or potentially interesting DATALOG queries. We also gave an algorithm, WARMR, for solving such tasks.

We have demonstrated the use of WARMR and WRMODE in practical tasks in the domains of telecommunication alarm analysis and chemical toxicology. We have given examples of how to use the presented methods in useful novel settings.

WARMR, which is available for academic purposes upon request, is a flexible tool that can be used by both users and developers as an explorative data mining tool: pattern types can be modified in a flexible way, and thus a number of settings can be easily experimented with without changes in the implementation.

Possible directions for future research on frequent query discovery include at least the following. First, an efficient general method could be developed for query reorganisation to minimize backtracking during query evaluation (cf. Section 3.2.2) and facilitate pruning during query generation (cf. Section 3.2.3). Second, in the spirit of Tables 2 and Table 3, a user-friendly generic system could be developed that automatically selects the most efficient algorithm available. This could be done on the basis of an analysis of the user inputs, i.e. the database and the language bias. Fourth, Table 2 uncovers a number of "gaps" that could be filled with some useful specialized algorithms. Fifth, many optimizations and techniques for mining and postprocessing frequent patterns and association rules have been proposed. Some of these, such as the sampling techniques described in (Toivonen, 1996), could probably be plugged into WARMR.

Acknowledgments

Luc Dehaspe is supported by ESPRIT Long Term Research Project No 20237, ILP^2. Hannu Toivonen is supported by the Academy of Finland. This paper was conceived while Hannu

Toivonen was visiting the Department of Computer Science, Katholieke Universiteit Leuven.

The authors are grateful to Luc De Raedt and Heikki Mannila for comments on the paper and for many fundamental ideas and discussions, to Hendrik Blockeel, Bart Demoen, and Wim Van Laer for their share in the implementation of WARMR, and to Hendrik Blockeel, Wim Van Laer, and Hélène Legras for proofreading. Data for the telecommunication alarm sequence analysis experiments is provided by Juha Leino from Nokia Telecommunications. The PTE data were made available by Ashwin Srinivasan and Ross King. Ross King also provided the evaluation of patterns in the PTE experiment.

Profound comments from anonymous reviewers and Sašo Džerodski are gratefully acknowledged.

References

Adé, H., De Raedt, L. and Bruynooghe, M. 1995. Declarative Bias for Specific-to-General ILP Systems. Machine Learning 20:119 – 154.

Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95), pp. 3 – 14.

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

Agrawal, R. Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A. I. 1996. Fast discovery of association rules. Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA, pp. 307 – 328.

Ashby, J. and Tennant, R. W. 1991. Definitive relationships among chemical structure, carcinogenicity and mutagenicity for 301 chemicals tested by the U.S. NTP. Mutation Research, 257:229–306.

Bettini, C., Wang, X. S. and Jajodia, S. 1996. Testing complex temporal relationships involving multiple granularities and its application to data mining. Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96), pp. 68 – 78.

Blockeel, H. and De Raedt, L. 1996. Relational knowledge discovery in databases. Proceedings of the 6th International Workshop on Inductive Logic Programming. Lecture Notes in Artificial Intelligence, Springer-Verlag, 1314, pp. 199–212.

Blockeel, H. and De Raedt, L. 1998. Top-down induction of first order logical decision trees. Artificial Intelligence, 101:285–297.

Blockeel, H., De Raedt, L., Jacobs, N. and Demoen, B. 1998a. Scaling up ILP by learning from interpretations. This volume.

Blockeel, H., De Raedt, L. and Ramon, J. 1998b. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, 55–63. Morgan Kaufmann.

Bristol, D., Wachsman, J. and Greenwell, A. 1996. The NIEHS predictive-toxicology evaluation project. *Environmental Health Perspectives* Supplement 3:1001–1010.

Clark, P. and Niblett, T. 1989. The CN2 algorithm. *Machine Learning* 3(4):261–284.

De Raedt, L. and Dehaspe, L. 1997. Clausal discovery. *Machine Learning* 26:99–146.

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

De Raedt, L. and Van Laer, W. 1995. Inductive constraint logic. In Jantke, K. P.; Shinohara, T.; and Zeugmann, T., eds., *Proceedings of the 6th International Workshop on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*, 80–94. Springer-Verlag.

De Raedt, L., Blockeel, H., Dehaspe, L. and Van Laer, W. 1998. Three companions for first order data mining. In Džeroski, S., and Lavrač, N., eds., *Inductive Logic Programming for Knowledge Discovery in Databases*, Lecture Notes in Artificial Intelligence. Springer-Verlag. To appear.

De Raedt, L. 1996. Induction in logic. In Michalski, R., and J., W., eds., *Proceedings of the 3rd International Workshop on Multistrategy Learning*, 29–38.

Dehaspe, L. and De Raedt, L. 1996. DLAB: A declarative language bias formalism. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS96)*, volume 1079 of *Lecture Notes in Artificial Intelligence*, 613–622. Springer-Verlag.

Dehaspe, L. and De Raedt, L. 1997. Mining association rules in multiple relations. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *Lecture Notes in Artificial Intelligence*, 125–132. Springer-Verlag.

Dehaspe, L., Toivonen, H. and King, R. 1998. Finding frequent substructures in chemical compounds. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, 30 – 36. AAAI Press.

Dehaspe, L. 1998. *Frequent pattern discovery in first-order logic*. Ph.D. Dissertation, K.U.Leuven.

Dietterich, T. G., Lathrop, R. H. and Lozano-Pérez, T. 1997. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence* 89(1-2):31–71.

Djoko, S., Cook, D. J. and Holder, L. B. 1995. Analyzing the benefits of domain knowledge in substructure discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, 75 – 80.

Dousson, C., Gaborit, P. and Ghallab, M. 1993. Situation recognition: Representation and algorithms. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, 166 – 172.

Džeroski, S., De Raedt, L. and Blockeel, H. 1998. Relational reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann.

Džeroski, S. 1996. Inductive logic programming and knowledge discovery in databases. In Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*. MIT Press. 118–152.

Goodman, R. M. and Latin, H. 1991. Automated knowledge acquisition from network management databases. In Krishnan, I., and Zimmer, W., eds., *Integrated Network Management, II*. Amsterdam, The Netherlands: Elsevier Science Publishers B.V (North-Holland). 541 – 549.

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)*, 420 – 431.

Hätönen, K.; Klemettinen, M.; Mannila, H.; Ronkainen, P.; and Toivonen, H. 1996. Knowledge discovery from telecommunication network alarm databases. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, 115 – 122. New Orleans, Louisiana: IEEE Computer Society Press.

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)*, 150 – 155. Montreal, Canada: AAAI Press.

Kietz, J. and Lübbe, M. 1994. An efficient subsumption algorithm for inductive logic programming. In *Proceedings of the 11th International Conference on Machine Learning*. Morgan Kaufmann.

Kietz, J.-U. and Wrobel, S. 1992. Controlling the complexity of learning in logic through syntactic and task-oriented models. In Muggleton, S., ed., *Inductive logic programming*. Academic Press. 335–359.

King, R. and Srinivasan, A. 1996. Prediction of rodent carcinogenicity bioassays from molecular structure using inductive logic programming. *Environmental Health Perspectives* 104(5):1031–1040.

King, R., Muggleton, S., Srinivasan, A. and Sternberg, M. 1996. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences* 93:438–442.

Klemettinen, M., Mannila, H. and Toivonen, H. 1998. Rule discovery in telecommunication alarm data. *Journal of Network and Systems Management*.

Klösgen, W. 1996. Explora: A multipattern and multistrategy discovery assistant. In Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*. MIT Press.

Kramer, S., Pfahringer, B. and Helma, C. 1997. Mining for causes of cancer: machine learning experiments at various levels of detail. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, 223 – 226.

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

Lindner, G. and Morik, K. 1995. Coupling a relational learning algorithm with a database system. In Kodratoff, Y.; Nakhaeizadeh, G.; and Taylor, G., eds., *Proceedings of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*.

Lu, H., Setiono, R., and Liu, H. 1995. Neurorule: A connectionist approach to data mining. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, 478 – 489.

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

Mannila, H. and Toivonen, H. 1997. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* 1(3):241 – 258.

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.

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

Morris, R. A., Khatib, L. and Ligozat, G. 1995. Generating scenarios from specifications of repeating events. In *Second International Workshop on Temporal Representation and Reasoning (TIME-95)*.

Muggleton, S. and De Raedt, L. 1994. Inductive logic programming : Theory and methods. *Journal of Logic Programming* 19,20:629–679.

Muggleton, S. 1995. Inverse entailment and Progol. *New Generation Computing* 13.

Muggleton, S. 1996. Learning from positive data. In Muggleton, S., ed., *Proceedings of the 6th International Workshop on Inductive Logic Programming*, 225–244. Stockholm University, Royal Institute of Technology.

Nédellec, C., Adé, H., Bergadano, F. and Tausend, B. 1996. Declarative bias in ILP. In De Raedt, L., ed., *Advances in Inductive Logic Programming*, volume 32 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 82–103.

Oates, T. and Cohen, P. R. 1996. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, 346 – 354. San Francisco, CA: Morgan Kaufmann.

Padmanabhan, B. and Tuzhilin, A. 1996. Pattern discovery in temporal databases: A temporal logic approach. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, 351–354.

Plotkin, G. 1970. A note on inductive generalization. In *Machine Intelligence*, volume 5. Edinburgh University Press. 153–163.

Quinlan, J. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Sasisekharan, R., Seshadri, V. and Weiss, S. M. 1996. Data mining and forecasting in large-scale telecommunication networks. *IEEE Expert, Intelligent Systems & Their Applications* 11(1):37 – 43.

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)*, 432 – 444.

Shen, W., Ong, K., Mitbander, B. and Zaniolo, C. 1996. Metaqueries for data mining. In Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*. MIT Press. 375–398.

Srikant, R. and Agrawal, R. 1995. Mining generalized association rules. In Dayal, U., Gray, P. M. D. and Nishio, S., eds., *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, 407 – 419. Zürich, Switzerland: Morgan Kaufmann.

Srikant, R. and Agrawal, R. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology—5th International Conference on Extending Database Technology (EDBT'96)*, 3 – 17.

Srikant, R., Vu, Q. and Agrawal, R. 1997. Mining association rules with item constraints. In Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., eds., *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, 67 – 73. AAAI Press.

Srinisavan, A., King, R. D., Muggleton, S. H. and Sternberg, M. J. E. 1997. The predictive toxicology evaluation challenge. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*. Morgan Kaufmann.

Srinivasan, A., King, R., Muggleton, S., and Sternberg, M. 1997. Carcinogenesis predictions using ILP. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, 273–287. Springer-Verlag.

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

Ullman, J. D. 1988. *Principles of Database and Knowledge-Base Systems*, volume I. Rockville, MD: Computer Science Press.

Wang, K. and Liu, H. 1997. Schema discovery for semistructured data. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, 271 – 274.

Wang, J. T.-L., Chirn, G.-W., Marr, T. G., Shapiro, B., Shasha, D. and Zhang, K. 1994. Combinatorial pattern discovery for scientific data: Some preliminary results. In Snodgrass, R., and Winslett, M., eds., *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'94)*, 115 – 125. Minneapolis, MI: ACM.

Wang, X., Wang, J. T. L., Shasha, D., Shapiro, B., Dikshitulu, S., Rigoutsos, I. and Zhang, K. 1997. Automated discovery of active motifs in three dimensional molecules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, 89 – 95.

Weber, I. 1997. Discovery of first-order regularities in a relational database using offline candidate determination. Proceedings of the 7th International Workshop on Inductive Logic Programming. Lecture Notes in Artificial Intelligence, Springer-Verlag, 1297, pp. 288–295.

Weber, I. 1998. A declarative language bias for levelwise search of first-order regularities. Proc. Fachgruppentreffen Maschinelles Lernen (FGML-98). Techn. Univ. Berlin, Technischer Bericht 98/11. http://www.informatik.uni-stuttgart.de/ifi/is/Pers onen/Irene/fgml98.ps.gz.

Wrobel, S. 1997. An algorithm for multi-relational discovery of subgroups. Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD '97). Springer-Verlag, pp. 78 – 87.

Luc Dehaspe received a Masters degree in Philology and a Masters degree in Computer Science from the University of Leuven. He obtained his Ph.D. in Computer Science from that same university in December 1998, with a thesis on frequent pattern discovery in a first-order logic framework. He is currently a member of the data mining and inductive logic programming research group at the Department of Computer Science. His research mainly focuses on the development and application of data mining tools that use first-order logic as the language to represent data and patterns.

Hannu Toivonen leads a data analysis and mining team at Rolf Nevanlinna Institute, University of Helsinki. He also holds an assistant professorship at the Department of Computer Science, where he earned his Ph.D. in 1996 with a thesis on the discovery of frequent patterns. Prior to joining the university Hannu Toivonnen was a research engineer at Nokia Research Center and at Nokia Telecommunications. His current research interests include, in addition to data mining, the use of Markov chain Monte Carlo methods for data analysis.