Constraint-Based Pattern Set Mining

Luc De Raedt
Katholieke Universiteit Leuven
Departement Computerwetenschappen
Celestijnenlaan 200a - bus 2402
3001 Heverlee
Luc.DeRaedt@cs.kuleuven.be

Albrecht Zimmermann
Katholieke Universiteit Leuven
Departement Computerwetenschappen
Celestijnenlaan 200a - bus 2402
3001 Heverlee
Albrecht.Zimmermann@cs.kuleuven.be

Abstract

Local pattern mining algorithms generate sets of patterns, which are typically not directly useful and have to be further processed before actual application or interpretation. Rather than investigating each pattern individually at the local level, we propose to mine for global models directly. A global model is essentially a pattern set that is interpreted as a disjunction of these patterns. It becomes possible to specify constraints at the level of the pattern sets of interest. This idea leads to the development of a constraint-based mining and inductive querying approach for global pattern mining. We introduce various natural types of constraints, discuss their properties, and show how they can be used for pattern set mining. A key contribution is that we show how well-known properties from local pattern mining, such as monotonicity and anti-monotonicity, can be adapted for use in pattern set mining. This, in turn, then allows us to adapt existing algorithms for item-set mining to pattern set mining. Two algorithms are presented, one level-wise algorithm that mines for all pattern sets that satisfy a conjunction of a monotonic and an anti-monotonic constraint, and an algorithm that adds the capability of asking top_k queries, We also report on a case study regarding classification rule selection using this new technique.

1 Introduction

The traditional *local pattern mining* task that is tackled in data mining is that of finding a theory $Th(\mathcal{L}, \mathcal{D}, q) = \{\phi \in \mathcal{L} \mid q(\phi, \mathcal{D}) \text{ is true}\}$, cf. [10], where $\mathcal{D}$ is a database, $\mathcal{L}$ a language of patterns, and $q(\phi, \mathcal{D})$ a selection predicate that states the constraints under which the pattern ϕ is a solution w.r.t. the database $\mathcal{D}$.

Numerous constraints and primitives have been devised for working with a wide variety of pattern domains. Indeed, constraint-based mining has been applied across a wide range of domains, ranging from item-sets, episodes and strings, to graphs and relational patterns. At the same time, researchers have looked at many classes of constraints, including for instance monotonic, anti-monotonic, succinct, convertible constraints [12, 2, 1], and employed a rich variety of primitives such as frequency, confidence, significance, lift, closedness, freeness, generality, etc. Furthermore, these primitives have been combined to form complex inductive queries, e.g. [12, 2, 3, 13, 7]. For instance, [3, 14] employ arbitrary boolean combinations of monotonic and anti-monotonic primitives to specify the inductive queries q in the $Th(\mathcal{L}, \mathcal{D}, q)$. One such query Q_0 could for instance ask for

$$\begin{aligned} sup(\phi, Actives) > 0.05 \quad &\wedge \\ sup(\phi, Inactives) < 0.01 \quad &\wedge \\ C - H - N \preceq \phi & \end{aligned}$$

This query searches in a database $\mathcal{D}$ of active and inactive molecules for patterns that contain a $C - H - N$ group (more formally: that are more specific than the pattern $C - H - N$), are frequent on the $Actives$ and infrequent on the $Inactives$. Other inductive queries have asked for those patterns for which the significance according to a χ^2 test exceeds a threshold (or belongs to the k best scoring patterns in $\mathcal{L}$), cf. [11]:

$$\chi^2(\phi, \mathcal{D}) > threshold\,(\arg_k \max_{\phi} \chi^2(\phi, \mathcal{D})).$$

Characteristic for this formulation of data mining is that one searches for *local patterns* in that each pattern is tested independently of the other patterns, and that the result is, typically, a (large, unstructured) set of patterns $Th(\mathcal{L}, \mathcal{D}, q)$. For the user of the data mining system, this situation is unsatisfactory, because a lot of time and effort needs to be put into post-processing $Th(\mathcal{L}, \mathcal{D}, q)$ in order to obtain a small set of patterns that is a true nugget of knowledge, i.e., can be readily interpreted or applied to solve problems of interest such as prediction or clustering. At the same time, contemporary data mining techniques offer too little help for supporting this post-processing process.

EXAMPLE 1. *To give an example of such a mining task, consider associative classification as in the case of* CBA *[9].*

All class association rules (association rules that predict one of the class labels) that satisfy minimum support and confidence (accuracy) thresholds are mined. The resulting solution set can easily encompass thousands of rules and to collect a subset of them for use in a classifier needs a post-processing step. CBA *solves the problem by using a heuristic database covering approach.*

The approach taken in this paper is to complement the first phase of data mining, where local patterns are being queried, with a second phase, in which *pattern sets* are being queried. Pattern sets are simply sets of patterns. For individual patterns, we will employ lower case characters such as $\phi, \psi, \ldots$ and for patterns sets, we will employ upper case characters $\Phi, \Psi, \ldots$ More formally, we will assume the following process:

1. $\mathbf{L} := Th(\mathcal{L}, \mathcal{D}, q)$
2. $M := \mathbf{Th}(\mathbf{L}, \mathcal{D}, \mathbf{p})$

in which

$$\mathbf{Th}(\mathbf{L}, \mathcal{D}, \mathbf{p}) = \{\Phi \subseteq \mathbf{L} | \mathbf{p}(\Phi, \mathcal{D}) is\ true\}$$

The resulting set of pattern sets M is obtained by formulating a constraint p that has to hold for pattern sets, i.e. subsets of $\mathbf{L}$, of interest w.r.t. the database $\mathcal{D}$. Here, $\mathbf{L}$ is the result of the local pattern mining query, and M the result of the post-processed query.

This formulation is quite natural for many data mining tasks. Indeed, whether the goal of the mining is to obtain a classifier in the form of a set of rules, a characterization of a set of clusters, or simply a set of patterns that are not too similar to one another, this can – as we will argue – be obtained by formulating appropriate queries in the second step.

One such query (formalized as query Q_3 below) queries for those sets of patterns of size at least two, in which the confidence in predicting the positive class is at least 95%, and the number of examples covered by the intersection of any pair of patterns is at most 10. This type of query could be used in a subgroup discovery context, since the goal there might be to characterize a subset of the data while avoiding too much redundancy.

The key contributions of this paper are: (1) the introduction of a novel framework for constraint-based mining of pattern sets; (2) the introduction of a number of interesting constraints for mining pattern sets; (3) the insight that the properties of constraints known from local pattern mining can be adapted in a natural way to those of constraints at the level of pattern sets; (4) the use of these properties to adapt existing algorithms from local pattern mining to pattern set mining, in particular, a variant of the level-wise algorithm of [10] and the branch-and-bound approach by [11]; (5) the introduction of an algorithm that combines the level-wise algorithm with that of the branch-and-bound one to compute the answer to complex queries.

To the best of the authors' knowledge this approach is new, with possibly the exception of the work by Shima *et. al* [15], which reports on a system for answering one specific type of pattern set mining query.

This paper is organized as follows: Section 2 introduces the formal framework of pattern set mining; Section 3 discusses properties of constraints for local pattern mining as well as pattern set mining; Section 4 introduces several primitives for pattern set mining, which are then integrated with aggregation primitives in Section 5; Section 6 contains some example queries; Section 7 analyzes the properties of the pattern set mining language that was introduced; Section 8 introduces the algorithm; Section 9 presents some experiments; and finally, Section 10 concludes and discusses related work.

2 Formal Framework

To keep our framework for supporting *pattern set queries* as general as possible, we will be imposing as few constraints on the local pattern mining task $Th(\mathcal{L}, \mathcal{D}, p)$ as possible. We will only assume that

- $\mathcal{D}$ is a database consisting of transactions t_i, these transactions are organized into possibly overlapping subsets $D_1, \cdots, D_n \subset \mathcal{D}$; the different sets D_i correspond to different classes, like active and inactive molecules;
- for each pattern $\phi \in \mathcal{L}$ and each transaction t, we can decide whether ϕ covers or matches the transaction t, i.e., whether $match(\phi, t)$ is true;
- it follows that for each data set D_i, we can compute $cov(D_i, \phi) = \{d \in D_i | match(\phi, d)\}$, the set of transactions in D_i that ϕ covers; it will often be more efficient to work with the transaction identifiers instead of the proper transactions; to this aim we use the notation $tid(D_i, \phi)$.
- the support of a pattern ϕ in a data set D_i, $sup(D_i, \phi) = |cov(D_i, \phi)|$
- $\mathcal{L}$ is a set of patterns that is partially ordered according to generality $\preceq$; basically for two patterns ϕ, ψ, we say that $\phi \preceq \psi$ if and only if for all possible transactions t: $match(\psi, t) \rightarrow match(\phi, t)$, i.e., whenever t is covered by ψ, it is also covered by ϕ; and
- p is an inductive query that can be answered.

For instance, when working with a set of items I, we have that $\mathcal{L} = 2^I$, each data set $D_i \subseteq 2^I$, a transaction t is covered by a pattern ϕ if $\phi \subseteq t$, ϕ is more general than ψ ($\phi \preceq \psi$) if and only if $\phi \subseteq \psi$. These definitions

easily extend towards other pattern domains, such as strings, episodes, graphs and relational queries.

The operations at the level of individual patterns ϕ can now easily be lifted towards the level of pattern sets Φ. While doing so, it is often useful to interpret a pattern set $\Phi = \{\phi_1, \cdots, \phi_n\}$ as the disjunction $\phi_1 \vee \cdots \vee \phi_n$ of the patterns ϕ_i it contains. This is reasonable as more pattern domains that are being employed (like item-sets) can be considered conjunctions. From this perspective, a pattern set Φ is a kind of formula in disjunctive normal form.

EXAMPLE 2. *To return to our earlier example, this is indeed consistent with the usage of rule lists used in classifiers such as the ones produced by* CBA. *Every single classification rule is treated as a conjunction of attribute-value combinations. But only a single rule is used for classification of a new instance, usually the highest-ranked rule matching it. If no rule covers it, a majority rule is used for classification.*

- A pattern set Φ matches a transaction t if and only if there exists a pattern $\phi \in \Phi$ that matches t; i.e., $match(\Phi, t)$ is true if and only if $\exists \phi \in \Phi : match(\phi, t)$ is true.
- By adopting this notion of matching, the definitions of cov, tid and sup directly carry over to the level of pattern sets.

The notion of generality deserves some more attention. We could directly apply the definition of generality and write that pattern set $\Phi \preceq \Psi$ if and only if for all possible transactions t it holds that $cov(\Psi, t) \rightarrow cov(\Phi, t)$. The problem with this is, however, that deciding $\Phi \preceq \Psi$ may be non-trivial. For instance, in case of Φ and Ψ being logical formulae (e.g., in disjunctive normal form as sketched above), this would amount to deciding whether Ψ logically entails Φ, i.e., whether $\Psi \models \Phi$. At the same time, it will be much harder to define refinement operators that compute minimal generalizations or specializations that belong to **L**. Therefore, we will adopt an easier definition of generality at the pattern set level. More formally,

DEFINITION 2.1. **Generality** *Pattern set Φ is more general than pattern set Ψ, notation $\Phi \preceq \Psi$, if and only if for all patterns $\psi_i \in \Psi$ there exists a pattern $\phi_j \in \Phi$ such that $\phi_j \preceq \psi_i$.*

It is easy to see that this definition is sound, but – depending on the pattern domain – not necessarily complete, i.e., whenever $\Phi \preceq \Psi$ this implies that for all transactions t, $match(\Psi, t) \rightarrow match(\Phi, t)$, but the reverse direction does not necessarily hold.

When manipulating pattern sets, it will sometimes be convenient to further simplify this notion of generality and use the simple subset relation.

DEFINITION 2.2. **Subset Generality** *Using the subset relation, Φ is (subset) more general than Ψ, notation $\Phi \preceq_s \Psi$ if and only if $\Psi \subseteq \Phi$.*

It is instructive to observe that $\Phi \preceq_s \Psi$ implies that $\Phi \preceq \Psi$, i.e., that Φ is more general than Ψ, though the opposite does not always hold. Consider for instance, in the item-set domain: for pattern sets $\{\{a\}\} \preceq \{\{a, b\}\}$ holds but $\{\{a\}\} \preceq_s \{\{a, b\}\}$ does not. There is an interesting property that can be exploited to characterize the relationship between $\preceq$ and $\preceq_s$, based on the notion of a reduced pattern set.

DEFINITION 2.3. **Reduced Pattern Sets** *A pattern set Φ is reduced if and only if $\forall \phi \in \Phi : \neg[(\Phi - \{\phi\}) \preceq \Phi]$. We use the notation $reduced(\Phi)$ to signify that Φ has this property.*

Reduced pattern sets contain no redundant patterns, which can be deleted without affecting the level of generality. Given a non-reduced pattern set Φ, it is easy to obtain a pattern set Ψ that is equivalent (w.r.t. $\preceq$) by repeatedly deleting patterns ϕ from Φ for which $(\Phi - \{\phi\}) \preceq \Phi$. The reduced set will be unique provided that the pattern language does not contain syntactic variants.

At this point, the reader commonly working with item-sets may notice that the direction of generality is reversed here. Indeed, when working at the local pattern level of item-sets, $\phi \subseteq \psi$ implies that $\phi \preceq \psi$. To see why this is the case, it is – again – convenient to recall the view of pattern sets as formulae in disjunctive normal form. No matter whether one is working with individual patterns or pattern sets, in this view, $G \preceq S$ if and only if $S \models G$. Subset generality will be convenient to generate pattern sets, since when using the subset relation, generalization corresponds to adding a pattern, and specialization to deleting one.

3 Properties of Constraints

Before discussing various primitives for pattern set mining, let us define – by analogy with the local pattern mining setting – some computationally interesting properties of such primitives.

DEFINITION 3.1. **Monotonicity** *A constraint p is said to be* monotone *w.r.t. a generality relation $\leq$ if and only if for all hypotheses A and B, $A \leq B$ and $p(A) \rightarrow p(B)$. It is said to be* anti-monotone *if and only if for all hypotheses A and B, $A \leq B$ and $p(B) \rightarrow p(A)$.*

This definition is applicable to local pattern mining using $\preceq$ as well as to pattern set mining using the relations $\preceq$ or $\preceq_s$. When the monotonicity property holds at the pattern level set for $\preceq_s$ but not for $\preceq$, we say that the constraint is *restricted* monotone. Restricted anti-monotone is defined similarly.

To give an intuition of the meaning of this property, monotone formalizes the notion that a constraint that is satisfied will stay satisfied if the pattern becomes more specific. Similarly, anti-monotone makes the same statement about a pattern that becomes more general.

As an example of a constraint that is anti-monotone, consider the classical minimal support threshold $sup(D, \phi) > threshold$ in the local item-set mining problem.

It is well-known that one can logically combine monotone and anti-monotone predicates. If the a's are anti-monotone and the m's monotone, then

- $\neg a$ is monotone and $\neg m$ is anti-monotone;
- $a_1 \wedge a_2$ and $a_1 \vee a_2$ are anti-monotone; and
- $m_1 \wedge m_2$ and $m_1 \vee m_2$ are monotone.

A relaxation of the monotone and anti-monotone properties that is still useful for local pattern mining assumes that a particular lexicographic order on the patterns exists. One then often talks about *convertible* constraints. To define convertible constraints, we will – for simplicity – assume that the hypotheses we work with are sets. This could be item-sets or pattern-sets. In addition, we assume that the generality relation corresponds to the subset relation.

DEFINITION 3.2. **Convertible Monotonicity** *A constraint p is said to be* convertible anti-monotone *(resp. convertible* monotone*) w.r.t. a pattern language $\mathcal{L}$, if and only if there exists an order $<$ on $\mathcal{L}$, such that for all prefix ordered pattern-sets $\{\phi_1, \cdots, \phi_n\}$ (for which $\phi_1 < \cdots < \phi_n$), $p(\{\phi_1, \cdots, \phi_{n-1}\}) \rightarrow p(\{\phi_1, \cdots, \phi_n\})$ (resp. $p(\{\phi_1, \cdots, \phi_n\}) \rightarrow p(\{\phi_1, \cdots, \phi_{n-1}\})$).*

An example convertible anti-monotone constraint is $avg(price(\phi)) > 100$, which states that the average price of the items in ϕ should be larger than 100. By ordering the items in I according to descending price, the constraint becomes convertible anti-monotone.

A third type of constraint that is often used is that of succinctness [12]. Again, we define succinct constraints at the level of item- and pattern-sets only. We also employ a simpler definition (proposed by Goethals, personal communication) that is equivalent to the original one in [12].

DEFINITION 3.3. **Succinctness** *A constraint predicate p defined on sets is* succinct *if and only if for all sets S: $p(S)$ can be expressed as $\forall e \in S : r(e) = true$ for a predicate r.*

So, for succinct constraints, one can test whether a hypothesis satisfies the constraint by testing that *all* its members satisfy the constraint. A standard example is requiring that the maximum price of the individual items is 1000 in item-set mining. This could be expressed as $max(price(\phi)) < 1000$. This definition is applicable to local pattern mining using $\preceq$ as well as to pattern set mining using the relations $\preceq$ or $\preceq_s$.

Within the literature on local pattern mining, branch-and-bound algorithms have also been successfully employed to answer top_k queries, which look for the k local patterns that score best or to find all patterns for which the score exceeds a certain threshold w.r.t. convex scoring function, such as χ^2 [11]. This is realized using *boundable* constraints.

DEFINITION 3.4. **Boundable** *A function $f : \mathcal{L} \mapsto \mathbb{R}$ is said to be* upper-boundable (resp. lower-boundable) *w.r.t. to a pattern language $\mathcal{L}$ if given $f(\phi) = k$, $k' \in \mathbb{R}$ can be derived s.t. $f(\phi') \leq k'$ (resp. $f(\phi') \geq k'$) for all $\phi' \prec \phi$ (resp. $\phi \prec \phi'$).*

Note that we do *not* call a function boundable if k' corresponds to a global maximum or minimum for all p'. If a function is upper-boundable for $\phi' \prec \phi$ (resp. $\phi \prec \phi'$), it is said to be *generalization* (resp. *specialization*) upper-boundable, and the dual property holds for lower-boundable. *Accuracy* for instance is not specialization upper boundable, since the bound is always 1.

This property has been used in the literature for mining of correlated conjunctive patterns, using measures such as information gain and χ^2 [11]. The techniques employed there base on the fact that the non-increase of support of such patterns w.r.t. specialization allows calculation of upper bounds on future values of the measure. For instance, anti-monotone constraints such as minimum support are also specialization upper-boundable since for all specializations ψ of a pattern ϕ, $sup(\psi, D) \leq sup(\phi, D)$.

4 Primitives for Pattern Set Mining

In this section, we introduce some useful primitives for pattern set mining and discuss their properties. The list is by no means exhaustive; it merely serves to illustrate the framework and its intuitive appeal.

The first type of constraints are direct adaptations of the well-known constraints for local pattern mining. In particular, some standard constraints include

- $sup(D, \Phi) > threshold$, is anti-monotone, and $sup(D, \Phi) < threshold$, is monotone;
- $size(\Phi) > threshold$, where $size(\Phi) = |\phi|$, is anti-monotone, and $size(\Phi) < threshold$, is monotone.
- $\Phi \preceq S$, where S is a particular pattern set, is anti-monotone, and $G \preceq \Phi$, where G is a particular pattern set, is monotone.

Before specifying some further primitive constraints, let us define some useful measures that can be derived from the support.

DEFINITION 4.1. **Redundancy** *Given two patterns ϕ, ϕ', their* overlap *in a data set D is defined as $ovlp(\phi, \phi', D) = cov(D, \phi) \cap cov(D, \phi')$. Their* redundancy *in D is defined as $red(\phi, \phi', D) = |ovlp(\phi, \phi', D)|$. Their* relative *redundancy is $relred(\phi, \phi', D) = \frac{red(\phi, \phi', D)}{|D|}$.*

The redundancy is a measure of the degree to which two patterns overlap. As for frequency, it can be defined either absolutely or defined relatively to the size of the data set D.

EXAMPLE 3. *This becomes important in cases in which e.g. a characterization of a dataset is wanted, consisting of patterns showing little redundancy which thus highlight characteristic properties of certain subsets. Similarly, when using an unordered rule set, little redundancy means few instances which might be classified differently by different rules, lessening the need for conflict solution heuristics.*

Similarly to overlap, symmetric difference is a useful primitive relating two patterns (kindly suggested by Nijssen, personal communication):

DEFINITION 4.2. **Distinctiveness** *Given two patterns ϕ, ϕ', the set $\mathrm{diff}(\phi, \phi', D) = \{cov(\phi, D) \cup cov(\phi', D)\} \setminus ovlp(\phi, \phi', D)$ is called their* symmetric difference. *Their* distinctiveness *is $dist(\phi, \phi', D) = |\mathrm{diff}(\phi, \phi', D)|$.*

Dually to redundancy, distinctiveness quantifies how much of the covered data is *exclusively* covered by one of the two patterns.

Again, one could define the relative distinctness $reldist$, as a variant.

Especially for settings such as subgroup discovery the existing data is split into subsets corresponding to interesting groups. For pattern sets mined in such a setting, predicates that are defined w.r.t. the different subsets have to be used for effective mining. This will very likely lead to a situation in which boolean combinations of predicates have to be used for mining and pruning, probably with differing properties. In this context, measures such as the representativeness are useful.

DEFINITION 4.3. **Representativeness** *Given a pattern set or pattern S and database $\mathcal{D} = \{D_1, ..., D_n\}$, the* representativeness *of S w.r.t. a given D_k is defined as $rep(S, D_k, \mathcal{D}) = \frac{sup(S, D_k)}{sup(S, \mathcal{D})}$.*

The representativeness of a pattern indicates how characteristic the examples covered by the pattern set are for the subset D_k. In case membership of the data set D_k would be represented as an item or attribute, the representativeness would correspond to the confidence of the association rule with the pattern as condition part and the class as the conclusion part.

5 Aggregation

Size, redundancy, distinctiveness and representativeness are primitives of the constraint language. Typically, these primitives will not be employed in an isolated fashion, but rather they will be combined with aggregates that range over the whole set of patterns in the set or, in the case of pairwise primitives, over the set of pairs of patterns in the set. We shall be using the typical aggregates such as avg, min, sum and max. For instance, the constraint

$$max(sup(\Phi, D)) < threshold$$

denotes that the maximum support of any pattern in Φ should be less than $threshold$. So, it is actually an abbreviation for

$$\max_{\phi \in \Phi}(sup(\phi, D)) < threshold$$

Similarly, the constraint

$$sum(red(\Phi, D)) < threshold$$

denotes that the sum, taken over all pairs of patterns in Φ, of the redundancies should be less than $threshold$. More formally, this amounts to

$$\sum_{i<j, \phi_i, \phi_j \in \Phi} red(\phi_i, \phi_j, D) < threshold$$

In a similar manner, $all(red(\Phi, D)) \leq threshold$ and $exists(red(\Phi, D)) \leq threshold$ will be interpreted.

6 Top-k Mining

A slightly different mining task is represented by top-k mining. The goal here is not mining of *all* patterns satisfying some constraints. Instead, the user specifies an integer k and a quality measure f and formulates a query:

$$\arg_k \max_{\Phi} f(\Phi, \mathcal{D})$$

This query denotes that the arguments, i.e. pattern sets, of the k highest scores assigned by f should be returned as solution. In addition, a minimum (or maximum) threshold can be set w.r.t. the evaluation function.

7 Example Queries

In this section, we list some example queries that are meant to illustrate the expressiveness and the use of constraint-based pattern set mining. Recall also that all the queries sketched assume that a previous query at the *local pattern mining* level has already been formulated. It may be convenient to keep the query Q_0 in mind that was formulated in the introduction The pattern set variable Φ then ranges over all subsets of $Th(\mathcal{L}, Mol, Q_0)$, where the database Mol of

molecules is composed of two disjoint sets: Act, the actives, and $InAct$, the inactives.

First, in a summarization or clustering context, the user might pose Q_1:

$$sup(\Phi, Act) \geq k_1 \wedge max(red(\Phi, All)) \leq 1$$

It asks for those sets of patterns that together cover at least k_1 active molecules and in which the conjunction of each pair of patterns covers at most 1 example. The minimum support ensures that the pattern collection is really representative of the dataset while the maximum redundancy, as mentioned above, means that individual capture characteristics particular to certain subsets.

Second, when focusing on classification or accuracy, one might pose Q_2:

$$all(rep(\Phi, Act, Mol)) \geq 0.95 \wedge$$
$$max(red(\Phi, Mol)) \leq k_1 \wedge size(\Phi) \geq 2$$

Q_2 generates sets of patterns of size at least two, in which the representativeness, (i.e., the confidence in predicting an active) is at least 95%, and the number of examples covered by the intersection of any pair of patterns is at most k_1. This type of query is related to the process of subgroup discovery.

Third, our chemical expert suggested query Q_3, which models a form of chemical interestingness:

$$\Phi \preceq \{p_1\} \wedge sup(\Phi, Act) \geq k_1$$
$$\wedge size(\Phi) \leq 20 \wedge sup(\Phi, InAct) \leq k_2$$

Here, the expert is looking for pattern sets that are frequent in the actives, infrequent in the inactives, contain pattern p_1 and have size at most 20.

Fourth, one may want to find sets of patterns that cover similar sets of examples in Act by employing query Q_4.

$$min(red(\Phi, Act)) \geq 5 \wedge size(\Phi) \geq 2$$

This requires that the intersection of each pair of patterns covers at least 5 examples and that the size of the pattern set is larger than 2. This could be used to identify families of patterns that characterize the same instances.

Finally, an associative classification rule learner such as *CBA* aims at selecting a subset of the mined rules with high accuracy as the final classifier. The rule set should have high accuracy on a validation set and little redundancy among the rules. Thus Q_5 has the form:

$$\max(red(\Phi, D) \leq k \wedge \arg_1 \max_{\Phi} acc(\Phi, D) \geq r$$

The second term denotes that we are querying for the single most accurate pattern set among those with accuracy at least r.

8 Properties of Pattern Set Constraints

Constraint based data mining system rely heavily on the properties of the employed constraints in order to 'push' the constraints in the data mining system and to develop efficient and effective algorithms. Therefore, we study in this section the properties of the pattern set constraint primitives introduced earlier.

THEOREM 8.1. *Each constraint mentioned in Table 1 possesses the properties listed.*

We omit the proof due to the space restrictions.

Using this list of properties, we can now investigate the properties of the conjunctive queries, given earlier as examples:

- Q_1 is written as the conjunction of an anti-monotone and an monotone constraint, i.e., in the form 'anti-monotone $\wedge$ monotone';
- Q_2 is in the form 'succinct $\wedge$ monotone $\wedge$ anti-monotone';
- Q_3 is in the form (anti-monotone $\wedge$ anti-monotone) $\wedge$ (monotone $\wedge$ monotone)', and hence, 'anti-monotone $\wedge$ monotone';
- Q_4 is in the form 'restricted monotone $\wedge$ anti-monotone; and
- Q_5 is monotone $\wedge$ 'generalization upper boundable'.

Observe that it is – of course – also possible to provide further queries and analyze their properties. For instance, the specific type of query which was answered by [15] corresponds to

$$|\bigcup_{i<j,\phi_i,\phi_j \in \Phi} ovlp(\{\phi_i, \phi_j\}, D)| \leq k.$$

It is a monotone constraint.

9 Algorithms

Because many of the pattern set constraints exhibit similar properties as those of traditional local pattern mining, it is possible to adapt existing algorithms from local pattern mining to mine pattern sets. Constrained pattern set mining poses two challenges. On the one hand, and most importantly, the user should be able to specify in a flexible manner various types of constraints without getting overwhelmed by the number of answers. On the other hand, the algorithms answering the inductive queries should be computationally feasible.

Since many constraints listed in Table 1 show anti-monotone or monotone properties, a logical choice for an efficient pattern set mining algorithm is an adaptation of

Table 1: Set constraints and their properties.

Constraint	Property
$G \preceq \Phi$	monotone, primitive
$\Phi \preceq S$	anti-monotone, primitive
$size(\Phi) \geq k$	anti-monotone, primitive
$size(\Phi) \leq k$	monotone, primitive
$sup(\Phi, \mathcal{D}) \leq k$	monotone, primitive
$sup(\Phi, \mathcal{D}) \geq k$	anti-monotone, primitive
$rep(\Phi, D_1, D) \leq k$	generalization boundable$^{\#}$, primitive
$rep(\Phi, D_1, D) \geq k$	generalization boundable$^{\#}$, primitive
$all(rep(\Phi, D_1, D)) \leq k$	succinct, aggregate
$all(rep(\Phi, D_1, D)) \geq k$	succinct, aggregate
$max(sup(\Phi, D)) \leq k$	succinct, aggregate
$min(sup(\Phi, D)) \geq k$	succinct, aggregate
$max(red(\Phi, D)) \leq k$	monotone, aggregate
$max(red(\Phi, D)) \geq k$	anti-monotone, aggregate
$min(red(\Phi, D)) \leq k$	restricted anti-monotone*, aggregate
$min(red(\Phi, D)) \geq k$	restricted monotone*, aggregate
$avg(sup(\Phi, D))\{\geq, \leq\}k$	convertible, aggregate
$max(rep(\Phi, D_1, D)) \leq k$	succinct, aggregate
$min(rep(\Phi, D_1, D)) \geq k$	succinct, aggregate
$sum(red(\Phi, D)) \leq k$	monotone, aggregate
$sum(red(\Phi, D)) \geq k$	anti-monotone, aggregate
$\chi^2(sup(\Phi, D_+), sup(\Phi, D_-)) \geq k$	specialization upper-boundable, primitive
$min(red(\Phi, D)) \leq k$	restricted anti-monotone, aggregate
$min(red(\Phi, D)) \geq k$	restricted monotone, aggregate
$max(dist(\Phi, D))\{\geq, \leq\}k$	(restricted monotone)$^{+}$, aggregate
$min(dist(\Phi, D))\{\geq, \leq\}k$	(restricted monotone)$^{+}$, aggregate

the well-known level-wise algorithm by [10] extended to cope with monotone constraints. In the local pattern mining world, minimum frequency is arguably the most important and influential constraint. Adding, e.g., additional items to an itemset usually rapidly reduces its support, making this constraint so powerful. Appropriate thresholds will make sure that the number of itemsets mined stays relatively small and the itemsets themselves are of low cardinality and thus easy to interpret. When mining constrained pattern sets, the same properties should show up in sets satisfying constraints, e.g. few patterns should be involved and not too many pattern sets returned. Thus, minimum frequency's role is transferred to *maximum redundancy* since the disjunctive nature of pattern sets implies that even strict support thresholds are satisfied by many high-cardinality pattern sets.

The first algorithm assumes a constraint on pattern sets that takes the form, shown as Algorithm 1, $anti(\Phi) \wedge mono(\Phi)$, where $anti$ denotes an anti-monotone predicate and $mono$ a monotone one. Notice that these predicates themselves could be logical combinations of several constraints, as discussed in Section 3, such as the nones used in, for instance, [7].

At the algorithmic level, it is necessary to employ a refinement operator at the level of pattern-sets. Essentially, applying a refinement operator ρ to a pattern-set Φ returns a set of its direct specializations (or generalizations) w.r.t. a particular generality relation. The generality relation $\preceq_s$ is easy to use in this respect, because refinement is very similar, though dual, to item-set mining. More formally:

DEFINITION 9.1. *The generalization refinement operator* $\rho_g(\Phi) = \{\Phi \cup \{\phi\} | \phi \in \mathbf{L}\}$.

So, ρ_g simply adds patterns to the existing pattern set Φ in all possible ways. Whereas in item-set mining this would yield minimal specializations, it yields minimal generalizations for pattern sets.

A slightly more complex operator is needed to deal with the other generality relation amongst pattern sets, $\preceq$. Essentially, one needs to consider also the replacement of a pattern $\phi \in \Phi$ by its minimal generalizations in $\mathbf{L}$.

The algorithm enumerates pattern sets from specific to general, to create low-cardinality sets. At each level the monotone constraint is tested since pattern sets that don't satisfy it can never be generalized into pattern sets that do. All sets satisfying the monotone constraint are then extended using ρ_g. Furthermore, the sets satisfying the anti-monotone constraint as well are included in the solution set. These operations are repeated at each level until no candidates are left that need to be tested against $mono$.

As usual, various optimizations are possible. A potential improvement could for instance be gained from the fact that several constraints in table 1 are pairwise. This means that either the subset check has to be performed only for the "2-generalizations" or even better, that this check can be made an additional aspect in selecting the sets to be specialized.

Algorithm 1 The Levelwise algorithm.

Input:
$\mathbf{L}$, a set of patterns;
$anti(\Phi) \wedge mono(\Phi)$;
a database $\mathcal{D}$;
Output:
$\mathbf{Th}(\mathbf{L}, \mathcal{D}, anti(\Phi) \wedge mono(\Phi)) = \bigcup_i T_i$;

$L_0 := \{\emptyset\}; i := 0;$
while $L_i \neq \emptyset$ **do**
$M_i := \{\Phi \in L_i | mono(\Phi)\}$
$T_i := \{\Phi \in M_i | anti(\Phi)\}$
$L_{i+1} := \{\Psi | \Psi \in \rho_g(\Phi)$ for some $\Phi \in M_i\}$
$i := i + 1;$

In addition, many variants of this type of algorithm can be obtained, by referring to other search-strategies, e.g., depth-first [4] instead of breadth-first, or better evaluation strategies in case the constraints $mono$ and $anti$ are complex by themselves. Alternatively, one might be interested in finding maximal or minimal pattern sets that satisfy the constraints, which corresponds to the (positive) borders of version spaces, which could easily be devised, cf. [7], free or closed pattern sets, cf [2].

One of the problems with the levelwise algorithm for item-set mining is that typically too many solutions are found and also that it is hard to set the thresholds. Therefore, we expect that this problem for this type of algorithm carries over to the levelwise algorithm for mining pattern sets. Within local pattern mining, the typical solution is to search for the top-k best local patterns according to a heuristic criterion such as χ^2 or *information gain* or *weighted relative accuracy*. The algorithm of choice for tackling such a mining task would usually be a branch-and-bound algorithm, such as the one proposed by [11]. The convexity property means that upper bounds on the value these measures assign to specializations of already seen patterns can be calculated. Since convexity also allows for generalization upper bounds, convex measures used in local pattern mining can also used for pattern set mining. In addition, a measure such as *accuracy*, which is not specialization boundable, can be bounded for the pattern set case (under the assumption that no re-classification of instances by other patterns occurs).

Unfortunately, the typical branch-and-bound approach suffers from two problems. On the one hand, upper bounds are often too generous and do not leverage the same pruning power as (anti-)monotone constraints. On the other hand, while the best-first strategy would ideally raise the threshold quickly and also needs limited memory, in practice raising the threshold often happens in the early stages of the min-

ing run. The branch-and-bound approach thus sacrifices the pruning capabilities that can be gained by using the information about all subsets of the current set which is collected at the level before the current, should one adopt the levelwise approach.

Therefore, rather than presenting the typical branch-and-bound algorithm for mining for top k patterns, we introduce an extension of the levelwise algorithm that computes the answers to queries of the form

$$\arg_k \max_{\Psi}\{f(\Psi)|anti(\Psi) \wedge mono(\Psi)\}$$

So, we are interested in computing the top k pattern sets scored according to the function f of those pattern sets that satisfy $anti(\Psi) \wedge mono(\Psi)$. The resulting branch-and-bound algorithm using a generalization upper-boundable constraint is shown as Algorithm 2. By not specifying a top-k constraint, the regular level-wise algorithm described above is derived.

Algorithm 2 The Levelwise algorithm using a generalization upper-boundable constraint.

Input:
  L, a set of patterns;
  $anti(\Phi) \wedge mono(\Phi)$;
  $f(\Phi) \mapsto R$, selection measure;
τ minimum threshold;
k size of solution;
  a database $\mathcal{D}$;
Output:
  Th$= \arg_k \max_\Phi\{f(\Phi)|anti(\Phi) \wedge mono(\Phi)\}$;

$L_0 := \{\emptyset\}; i := 0;$
$Sol = \emptyset$
while $L_i \neq \emptyset$ **do**
  $M_i := \{\Phi \in L_i | mono(\Phi)\}$
  $T_i := \{\Phi \in M_i | anti(\Phi)\}$
  $Sol \cup \{\Phi \in T_i | f(\Phi) \geq \tau\}$
  while $|Sol| > k$ **do**
    $Sol \setminus \arg\min_{\Phi \in Sol} f(\Phi)$
  if $|Sol| = k$ **then**
    $\tau = \min_{\Phi \in Sol} f(\Phi)$
  $L_{i+1} := \{\Psi | \Psi \in \rho_g(\Phi), \Phi \in M_i, ub(f(\Phi)) > \tau\}$
  $i := i + 1;$
return Sol

There are a few differences to Algorithm 1. The first one lies in the fact that the size of the solution set is potentially limited. Thus, (potential) solutions do not only have to satisfy $mono \wedge anti$ but also be better than the k best pattern sets seen so far. Only then are they included in the solution set Sol (and can be replaced by better-scoring ones in the future). Also, candidates for expansion are not only tested against $mono$ but also on whether their upper bound exceeds the kth-best score seen so far.

What is perhaps more important than the specific algorithm is that essentially any principle that applies to item-set mining can be adapted towards pattern set mining. The main difference is that the direction of generality is inverted, as adding new elements to a set corresponds to generalization when working with pattern sets, instead of specialization.

10 Experiments

Throughout the paper we provided evidence that pattern set mining is quite analogous to local pattern mining. Therefore a key goal of the experiments is to verify whether the levelwise algorithm behaves in a similar way regarding pattern sets as it would mining item-sets. We evaluate this in the first experimental setting.

In addition, the use of a top-k-query for classifier construction is evaluated to provide example of an application of the pattern set mining approach.

For the first part of the experimental evaluation we mined local patterns on two datasets. The first data set is the DTP aids antiviral database, using the molecular feature miner MolFea, with a minimum support threshold on the *actives* of 27 and a maximum threshold on the *inactives* of 8. The resulting set of local patterns contains 287 patterns. This set was once used as is for the evaluation of Q_1 and Q_2 and for the evaluation of Q_3 a subset was selected in such a way that each *tid*-list occurred only once, leaving 29 patterns. The second data set is the well-known UCI mushroom dataset. We mined maximal patterns both on the entire dataset and on each subset separately, using Apriori and a minimum support of 25%, finding 101 and 131 patterns, respectively. We then used those sets of local patterns for our experiments.

10.1 Evaluating conjunctions *anti* $\wedge$ *mono* For this setting, we evaluated Q_1, Q_2, and Q_3 with different thresholds on the patterns mined on the DTP aids antiviral database. Results are shown in Tables 2, 3, 4. All tables report the constraint thresholds, number of constrained sets returned, minimum and maximum cardinality of constrained sets, respectively.

In Table 2, k denotes the minimum support threshold on *Act*, in Table 3, m denotes the maximum redundancy allowed. Only the values 0 and 11 are reported since any value below 11 behaves as in setting maximum redundancy to 0 and any value above 11 is obviously not practical. We also added a support threshold to Q_2, to additionally evaluate its effect, its threshold is denoted by k.

Finally, in Table 4, k_1 and k_2 denote the maximum support on the *inactives* and minimum support on the *actives*, respectively.

On the one hand are the resulting pattern sets are of

Table 2: Q_1 evaluated on $\mathbf{L} = sup(\phi, Act) \geq 27$

k	$\lvert\mathbf{Th}\rvert$	$\min \lvert\Phi\rvert$	$\max \lvert\Phi\rvert$
50	8525392	2	5
100	8195886	3	5
150	2649152	5	5
160	42688	5	5
162	13920	5	5

Table 3: Q_2 evaluated on $\mathbf{L} = sup(\phi, Act) \geq 27$

m	k	$\lvert\mathbf{Th}\rvert$	$\min \lvert\Phi\rvert$	$\max \lvert\Phi\rvert$
0	n/a	3690	2	2
0	40	180	2	2
0	50	90	2	2
0	55	32	2	2
0	57	0	n/a	n/a
11	n/a	776602	2	4
11	40	741028	2	4
11	50	134650	2	4
11	55	960	2	3
11	58	928	3	3
11	59	0	n/a	n/a

Table 4: Q_3 evaluated on $\mathbf{L} = sup(\phi, Act) \geq 27$, unique *tid*-lists

k_1	k_2	$\lvert\mathbf{Th}\rvert$	$\min \lvert\Phi\rvert$	$\max \lvert\Phi\rvert$
10	30	7678	2	13
10	50	6040	2	13
10	100	2	5	6
20	30	706975	2	17
20	50	703904	2	17
20	100	242377	4	17
20	130	2417	5	13
20	137	541	6	13
20	138	0	n/a	n/a

Table 5: Mushroom dataset, maximal sets mined on complete dataset
$max(red(\Phi, All)) < 1$, varying support

	$\lvert\mathbf{Th}\rvert$	$\min \lvert\Phi\rvert$	$\max \lvert\Phi\rvert$
$min(sup(\Phi, All) > 0$	923	1	3
$min(sup(\Phi, All) > 3000$	823	1	3
$min(sup(\Phi, E) > 3000$	245	2	3
$min(sup(\Phi, P) > 3000$	171	2	3

Table 6: Mushroom dataset, maximal sets mined on subsets
$max(red(\Phi, All)) < 1$, varying support

	$\lvert\mathbf{Th}\rvert$	$\min \lvert\Phi\rvert$	$\max \lvert\Phi\rvert$
$min(sup(\Phi, All) > 0$	23647	1	6
$min(sup(\Phi, All) > 3000$	21227	2	6
$min(sup(\Phi, E) > 3000$	1063	2	6
$min(sup(\Phi, P) > 3000$	5046	3	6

relatively low cardinality, facilitating the understanding of the results. On the other hand, the amount of pattern sets returned is high, even for restrictive constraint settings.

Additionally, as can be seen, selecting the right thresholds for the constraints is a non-trivial task. Sometimes, changing support by 1 means that no pattern set is returned at all.

These are well-known phenomena from local pattern mining. The fact that, as we expected, they are mirrored in the task of constrained pattern set mining as well gives additional support to our claim that the two tasks are structurally similar.

An interesting result is that using the patterns without ad-hoc removal of non-unique *tid*-lists allows for a larger minimum support on Act than if those patterns are removed. This shows that the task of removing non-unique *tid*-lists is itself essentially a constrained pattern set mining task.

In the case of patterns mined on the mushroom dataset, we constrained the maximum redundancy to 0 and evaluated the effect of setting a minimum support threshold of 3000 on the entire dataset, and each of the subsets. The results for these experiments are shown in Tables 5 and 6.

Again, it can be seen that pattern sets are of small cardinality, making it easy for the user to comprehend the entire set. On the other hand, even setting relatively strong thresholds (considering that each subset consists of roughly 4000 instances) still leads to a large number of pattern sets returned.

The experiment provides evidence that the levelwise algorithm behaves very similarly to the item-set mining task, including the regrettable effect of typically producing far more solutions than humans can peruse. Also, the cardinality of returned pattern sets is relatively small.

On the positive side this means that applying the lessons

Table 7: Comparing *CBA*'s classifier with constrained set mining result

	Size set	Acc_{train}	Acc_{test}
CBA	17.4 ± 5.1	0.8443 ± 0.019	0.7918 ± 0.0459
CSM	4.9 ± 1.66	0.8443 ± 0.019	0.7918 ± 0.0459

learned in local pattern mining regarding condensed representations, different enumeration (and ordering) strategies, and techniques for top-k-mining will be applicable to the pattern set mining task as well.

10.2 Classifier construction As mentioned before, *CBA* mines the set of all classification rules that have a minimum support and minimum accuracy on a set of training data. To construct a classifier out of these rules, a post-processing step is necessary to select a useful subset. This is done heuristically, traversing an ordered list of rules, adding ones that correctly classify instances that are not being classified by the rules chosen so far for the subset. The quality of the classifier constructed is measured on the training data. This also means that the order of the rules has quite an impact on the final classifier. Especially if rules are completely equal in support, accuracy and generality, the ordering becomes essentially arbitrary and different orderings will result in different classifiers.

In constrained pattern set mining, using a constraint such as Q_5 allows to ignore the ordering though and thus find one classifier with the highest accuracy on the training set.

To illustrate this, we set up the following small experiment: using *CBA*, we mined class association rules on ten folds of the UCI balance-scale dataset. On average 98.8 rules ($\sigma = 17.38$) were found. When randomly permutating the order of equal rules and using *CBA*'s post-processing step on the resulting rule sets, classifiers of different size and accuracy on the training sets are constructed.

Then we chose one permutation and additionally ran query Q_5 on that permutation to select a pattern set used as final classifier. Average and standard deviation for the size of the rule set selected by the post-processing step, accuracy on the training and the test data are shown in Table 7 for *CBA* and the constrained pattern set mining (*CSM*) approach in the first and second row, respectively. Maximum redundancy was set to 5 and minimum accuracy to 0.5.

While the two approaches construct classifiers of identical accuracy both on the training and test data, constrained pattern set mining allows to construct a far more compact classifier. Visual inspection of the rule sets mined by our approach shows that often rules are selected that are not included in *CBA*'s solution. Specifically, quite often, the highest-ranked rule is ignored by our approach while *CBA*'s post-processing will always include this one. Additionally, on several folds, the constrained pattern set did not include a single rule with confidence 1 without this decreasing the accuracy of the resulting classifier.

While mining for the constrained pattern sets, on average 18.6 levels were traversed ($\sigma = 2.319$), far from the search space that an exhaustive search for the best subset of on average 99 rules would need to consider.

11 Conclusions

We have introduced a novel framework for constraint-based mining at the pattern set level rather than at the level of individual local patterns. We introduced various types of pattern set primitives and constraints and analyzed their properties. It turned out that the key principles and findings of constraint-based local pattern mining can directly be adapted towards pattern set mining. This allowed us to adapt well-known algorithms from local pattern mining, such as a variant of the level wise algorithm that deals with conjunctions of monotonic and anti-monotonic constraints, and the branch-and-bound approach for dealing with top-k criteria due to [11] towards our purposes.

We also illustrated the approach in two pattern mining settings and a small-scale case study demonstrating an application of the technique. The experiments show that the behavior of the mining process and the relation between underlying language and query results closely resemble the phenomena encountered in local pattern mining.

To the best of the authors' knowledge, this is a novel approach to inductive querying [6, 3]. It is, however, related to the work by [15], who proposed an algorithm for solving one specific type of inductive query (as sketched above). There has been work on allowing for dynamic constraints in data mining [8, 5] that also attempts to give the user more control about the amount of patterns returned by adjusting parameters while the mining process is underway. These works focus mainly on the well-known minimum frequency constraint and somewhat on anti-monotone and succinct constraints. A main difference to our work lies in the fact that the mining process returns exactly the solution to these adjusted constraints while we suggest mining patterns at fixed constraints first and then selecting a subset that has further properties that are not that easily pushed into the local pattern mining step.

There are a lot of opportunities for further work. Nevertheless, we believe that our work already illustrates the benefits of the approach, and also, that – in principle – pattern set mining should work as efficient as item-set mining, where an important factor is the number of items (or, in our case, of patterns) to be considered.

There is a lot of room for developing further constraints, studying their properties and developing interesting algorithms for solving pattern set mining problems. Additionally, it might be interesting to transfer the lesson of constrained

pattern set mining back to local pattern mining, namely the positive effect of using an additional selection criterion to return a smaller solution set to the user. Finally, do the similarities between local pattern and global model mining also raise some general questions w.r.t. the *globality* of "global" models.

12 Acknowledgments

The authors would like to thank Andreas Karwath for providing some help with the MolFea experiments, Siegfried Nijssen for interesting discussions and suggesting the notion of distinctness, and Kristian Kersting for suggesting the name Aposteriori :-). This work was partly supported by the EU IST project IQ.

References

[1] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Examiner: Optimized level-wise frequent pattern mining with monotone constraint. In *ICDM*, pages 11–18. IEEE Computer Society, 2003.

[2] Toon Calders, Christophe Rigotti, and Jean-François Boulicaut. A survey on condensed representations for frequent sets. In Jean-François Boulicaut, Luc De Raedt, and Heikki Mannila, editors, *Constraint-Based Mining and Inductive Databases*. Springer, 2005.

[3] L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2), 2002.

[4] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[5] Christian Hidber. Online association rule mining. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD Conference*, pages 145–156. ACM Press, 1999.

[6] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Commun. ACM*, 39(11):58–64, 1996.

[7] Stefan Kramer, Luc De Raedt, and Christoph Helma. Molecular feature mining in hiv data. In *KDD-2001*, 2001.

[8] Laks V. S. Lakshmanan, Carson Kai-Sang Leung, and Raymond T. Ng. Efficient dynamic mining of constrained frequent sets. *ACM Trans. Database Syst.*, 28(4):337–389, 2003.

[9] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *KDD 1998*, pages 80–86, New York City, New York, USA, August 1998. AAAI Press.

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

[11] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *PODS*, 2000.

[12] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *SIGMOD Conference*, pages 13–24, 1998.

[13] Jian Pei, Jiawei Han, and Laks V. S. Lakshmanan. Pushing convertible constraints in frequent itemset mining. *Data Min. Knowl. Discov.*, 8(3):227–252, 2004.

[14] Luc De Raedt, Manfred Jaeger, Sau Dan Lee, and Heikki Mannila. A theory of inductive query answering. In *ICDM*, pages 123–130. IEEE Computer Society, 2002.

[15] Yoshikazu Shima, Kouichi Hirata, and Masateru Harao. Extraction of frequent few-overlapped monotone dnf formulas with depth-first pruning. In Tu Bao Ho, David Cheung, and Huan Liu, editors, *PAKDD*, pages 50–60. Springer, 2005.