The Levelwise Version Space Algorithm and its Application to Molecular Fragment Finding

Luc De Raedt and **Stefan Kramer**
Albert-Ludwigs-University Freiburg
Institute for Computer Science
Georges-Köhler-Allee Geb. 079
D-79110 Freiburg in Breisgau, Germany
{deraedt, skramer}@informatik.uni-freiburg.de

Abstract

A tight integration of Mitchell's version space algorithm with Agrawal *et al.*'s Apriori algorithm is presented. The algorithm can be used to generate patterns that satisfy a variety of constraints on data. Constraints that can be imposed on patterns include the generality relation among patterns and imposing a minimum or a maximum frequency on data sets of interest.

The theoretical framework is applied to an important application in chemo-informatics, i.e. that of finding fragments of interest within a given set of compounds. Fragments are linearly connected substructures of compounds. An implementation as well as preliminary experiments within the application are presented.

1 Introduction

Mannila and Toivonen [Mannila and Toivonen, 1997] formulate the general pattern discovery task as follows. Given a database r, a language $\mathcal{L}$ for expressing patterns, and a constraint q, find the theory of r with respect to $\mathcal{L}$ and q, i.e. $Th(\mathcal{L}, r, q) = \{\phi \in \mathcal{L} \mid q(r, \phi)\ is\ true\}$. Viewed in this way $Th(\mathcal{L}, r, q)$ contains all sentences within the pattern language considered that make the constraint q true. This formulation of pattern discovery is generic in that it makes abstraction of several specific tasks including the discovery of association rules, frequent patterns, inclusion dependencies, functional dependencies, frequent episodes, ... Also, efficient algorithms for solving these tasks are known (cf. [Mannila and Toivonen, 1997]).

So far the the type of constraint that has been considered is rather simple and typically relies on the frequency of patterns. In the past decade the data mining community spent a lot of effort to efficiently compute patterns having a minimum frequency, such as e.g. Apriori [Agrawal *et al.*, 1993]. In this paper, we will extend this popular data mining model by allowing the user to specify a variety of different constraints on the patterns of interest. The constraints that will be considered involve generality constraints on patterns, e.g. to specify that the patterns of interest are (resp. are not) more general than a specific pattern, as well as frequency constraints. Frequency constraints that are considered either impose a maximum or a minimum frequency on a data set of interest. These constraints can then be combined e.g. in order to discover all patterns that are more general than pattern x, have a minimum frequency m_1 on the dataset p, and a maximum frequency m_2 on data set n. The result is a flexible and declarative query language to specify the patterns of interest. From this point of view, the work also fits in the inductive database framework considered by researchers such as [Imielinski and Mannila, 1996; Han *et al.*, 1999; De Raedt, 2000].

The key problem in discovering theories that involve a conjunction of primitive constraints $c_1 \wedge ... \wedge c_n$ is to efficiently combine the solvers for the primitive constraints c_i. It is at this point where Mitchell's version space approach [Mitchell, 1982] is extremely useful. Indeed, each of the primitive constraints results in a version space. Consider e.g. the minimum frequency constraint. As shown by [Mannila and Toivonen, 1997], the minimum frequency constraint results in a space of solutions with the most general pattern $\top$ as the only element of the G-set and the BD^+ boundary as the S-set. Because this property holds for all primitive constraints, the space of solutions of the conjunction of constraints can also be specified as a version space. Moreover, it can – in principle – be computed using Hirsh's version space intersection method [Hirsh, 1994]. However, rather than applying Hirsh's framework, we will employ a tighter integration of version spaces with Apriori.

To demonstrate the relevance of level-wise version spaces, we present an implementation as well as experiments in the domain of *molecular fragment finding*. Molecular fragments are sequences of linearly connected atoms. They are useful and important for the induction of so-called Structure-Activity Relationships (SARs), which are statistical models that relate chemical structure to biological activity. The use of automatically derived fragments in SARs originates from the CASE/MultiCASE systems developed by [Rosenkranz *et al.*, 1999]. With more than 150 published references, the CASE/MultiCASE systems are the most extensively used SAR and predictive toxicology systems. Previous approaches in these areas are based on the "decomposition" of individual compounds: these methods generate *all* fragments occurring in a given single compound. In this regard, our contribution is a language that enables the formulation of complex queries

regarding fragments – users can specify precisely which fragments they are interested in. We also implemented an efficient solver to answer queries in this language. Thus, from the algorithmic point of view, it is no longer necessary to process the results of queries post-hoc.

Molecular fragment finding has also been studied within the context of inductive logic programming and knowledge discovery in databases. For instance, Warmr [Dehaspe and Toivonen, 1999] or the approach by Inokuchi *et al.* [Inokuchi *et al.*, 2000] have been used in this context. Warmr is a system discovering frequently succeeding Datalog queries, and thus is not restricted to fragments. The approach by Inokuchi *et al.* deals with arbitrary frequent subgraphs, and thus is not restricted to linear fragments. Both approaches differ in that their pattern domain is more expressive, but finding frequent patterns is likely to be more expensive and complex than for linear fragments. Another approach to fragment finding is bottom-up propositionalization [Kramer and Frank, 2000]. All of these approaches handle only minimum frequency thresholds.

In contrast to all these approaches, the presented work allows the specification of all sorts of constraints, for instance regarding generality or frequency. Also, for the first time, one can pose constraints on the maximum frequency of fragments, and not only on the minimum frequency.

Finally, we would like to stress that the integration with version spaces results in a very compact representation of the resulting solutions. Previous methods would typically output all patterns within the version space. This is interesting for understandability reasons and also for further learning with these fragments as features.

The paper is organised as follows : in Section 2, we introduce the molecular fragment finding task and the primitives for querying such fragments, in Section 3, we present the level-wise version space algorithm, in Section 4, we discuss some experiments in molecular fragment finding, and in Section 5, we conclude and touch upon related work.

2 Framework

2.1 Molecular fragment finding

The task to which we will apply our integrated version space - Apriori framework is that of finding all molecular fragments that satisfy a conjunction of constraints $c_1 \wedge ... \wedge c_n$.

A *molecular fragment* is defined as a sequence of linearly connected atoms. For instance, '$o-s-c$' is a fragment meaning: "an oxygen atom with a single bond to a sulfur atom with a single bond to a carbon atom". In such expressions 'c', 'n', 'cl', etc. denote elements, and '$-$' denotes a single bond, '=' a double bond, '#' a triple bond, and '$\sim$' an aromatic bond. As common in the literature, we only consider "heavy" (i.e., non-hydrogen) atoms in this paper.

We assume that the system is given a database of example compounds and that each of the example compounds in the database is described using a 2-D representation. The information given there consists of the elements of the atoms of a molecule and the bond orders (single, double, triple, aromatic). An example compound in such a representation is shown in Fig. 1.

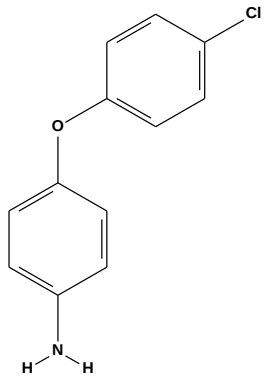

Figure 1: Example compound in a 2-D representation. '$cl - c \sim c \sim c \sim c - o$' is an example fragment occurring in the molecule.

A molecular fragment f *covers* an example compound e if and only if f considered as a graph is a subgraph of example e. For instance, fragment '$cl - c \sim c \sim c \sim c - o$' covers the example compound in Fig. 1.

There are a number of interesting properties of the language of molecular fragments $\mathcal{M}$:

- fragments in $\mathcal{M}$ are partially ordered by the *is more general than* relation; when fragment g is more general than fragment s we will write $g \leq s$;
- within this partial order, two syntactically different fragments are equivalent only when they are a reversal of one another; e.g. '$c - o - s$' and '$s - o - c$' denote the same substructure;
- $g \leq s$ if and only if g is a subsequence of s or g is a subsequence of the reversal of s; e.g. '$c-o$' $\leq$ '$c-o-s$'.
- there is a unique maximally general fragment (the empty fragment), which we denote by $\top$
- there is no maximally specific fragment; however, for convenience we add an artificial one to $\mathcal{M}$, which we denote by $\bot$.

Note that the representation of molecular fragments is relatively restricted compared to some other representations employed in data mining, such as first-order queries [Dehaspe and Toivonen, 1999] or subgraphs [Inokuchi *et al.*, 2000]. Although fragments are a relatively restricted representation of chemical structure, it is easy for trained chemists to recognize the functional group(s) that a given fragment occurs in. Thus, the interpretation of a fragment reveals more than meets the eye.

2.2 Constraints on fragments

The task addressed in this paper is that of finding the set of all fragments $f \in \mathcal{M}$ which satisfy a conjunction of primitive constraints $c_1 \wedge ... \wedge c_n$. The primitive constraints c_i imposed on the unknown target fragments f are :

- $f \leq p$, $p \leq f$, $\neg(f \leq p)$ and $\neg(p \leq f)$: where f is the unknown target fragment and p is a specific pattern; this type of primitive constraint denotes that f should (not)

be more specific (general) than the specified fragment p; e.g. the constraint '$c-o$' $\leq f$ specifies that f should be more specific than '$c-o$', i.e. that f should contain '$c-o$' as a subsequence;

- $freq(f, D)$ denotes the frequency of a fragment f on a set of molecules D; the frequency of a fragment f w.r.t. a dataset D is defined as the number of molecules in D that f covers;
- $freq(f, D_1) \leq t$, $freq(f, D_2) \geq t$ where t is a positive integer and D_1 and D_2 are sets of molecules; this constraint denotes that the frequency of f on the dataset D_i should be larger than (resp. smaller than) or equal to t; e.g. the constraint $freq(f, Pos) \geq 100$ denotes that the target fragments f should have a minimum frequency of 100 on the set of molecules Pos.

These primitive constraints can now conjunctively be combined in order to declaratively specify the target fragments of interest. Note that the conjunction may specify constraints w.r.t. any number of datasets, e.g. imposing a minimum frequency on a set of active molecules, and a maximum one on a set of inactive ones. E.g. the following constraint: $('c-o' \leq f) \wedge \neg(f \leq 'c-o-s-c-o-s') \wedge freq(f, Act) \geq 100 \wedge freq(f, InAct) \leq 5)$ queries for all fragments that include the sequence '$c-o$', are not a subsequence of '$c-o-s-c-o-s$', have a frequency on Act that is larger than 100 and and a frequency on $InAct$ that is smaller than 5.

3 Solving constraints

In this section, we will discuss how to find the set of all solutions $sol(c_1 \wedge ... \wedge c_n)$ in $\mathcal{M}$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$.

3.1 The search space

Due to the fact that the primitive constraints c_i are independent of one another, it follows that

$$sol(c_1 \wedge ... \wedge c_n) = sol(c_1) \cap ... \cap sol(c_n)$$

So, we can find the overall solutions by taking the intersection of the primitive ones.

Secondly, each of the primitive constraints c is monotonic or anti-monotonic w.r.t. generality (cf. [Mannila and Toivonen, 1997]). A constraint c is *anti-monotonic* (resp. monotonic) w.r.t. generality whenever

$$\forall s, g \in M : (g \leq s) \wedge (s \in sol(c)) \rightarrow (g \in sol(c))$$

(resp. $(g \in sol(c)) \rightarrow (s \in sol(c))$). The basic anti-monotonic constraints in our framework are: $(f \leq p), freq(f, D) \geq m$, the basic monotonic ones are $(p \leq f), freq(f, D) \leq m$. Furthermore the negation of a monotic constraint is anti-monotonic and vice versa.

Monotonic and anti-monotonic constraints are important because their solution space is bounded by a border. This fact is well-known in both the data mining literature (cf. [Mannila and Toivonen, 1997]), where the borders are often denoted by BD^+, as well as the machine learning literature (cf. [Mitchell, 1982]), where the symbols S and G are typically used.

To define borders, we need the notions of minimal and maximal elements of a set w.r.t. generality. Let F be a set of fragments, then define

$$max(F) = \{f \in F \mid \neg\exists q \in F : f \leq q\}$$
$$min(F) = \{f \in F \mid \neg\exists q \in F : q \leq f\}$$

We can now define the borders $S(c)$ and $G(c)$[1] of a primitive constraint c as

$$S(c) = min(sol(c)) \; and \; G(c) = max(sol(c))$$

Anti-monotonic constraints c will have $G(c) = \{\top\}$ and for proper constraints $S(c) \neq \{\bot\}$; proper monotonic constraints have $S(c) = \{\bot\}$ and $G(c) \neq \{\top\}$. Furthermore, as in Mitchell's version space framework we have that

$$sol(c) = \{f \in \mathcal{M} \mid \exists s \in S(c), \exists g \in G(c) : g \leq f \leq s\}$$

This last property implies that $S(c)$ (resp. $G(c)$) are proper borders for anti-monotone (resp. monotone) constraints.

So, we have that the set of solutions $sol(c_i)$ to each primitive constraint is a simple version space completely characterized by $S(c_i)$ and $G(c_i)$. Therefore, the set of solutions $sol(c_1 \wedge ... \wedge c_n)$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$ will also be completely characterized by the corresponding $S(c_1 \wedge ... \wedge c_n)$ and $G(c_1 \wedge ... \wedge c_n)$. At this point there are two issues:

1. computing the borders $S(c_i)$ and $G(c_i)$ for each primitive constraint c_i
2. computing the $S(c_1 \wedge ... \wedge c_n)$ and $G(c_1 \wedge ... \wedge c_n)$ given the individual borders $S(c_i)$ and $G(c_i)$

The first issue could be addressed using (variants of) the common level-wise algorithm for data mining (cf. [Mannila and Toivonen, 1997]). The second one could – in principle – be solved using Hirsh's version space merging algorithm. By now, it is probably clear that we need to integrate the level-wise algorithm with that of version spaces. However, rather than taking a loose coupling of the two approaches (as sketched above) we will provide a tighter integration of the two approaches. As we will show in the experimental section, this will lead to computational advantages.

The integrated algorithm computes the overall border sets incrementally. It initializes the borders S and G with the minimal and maximal elements, and then repeatedly updates for each primitive constraint. To update the borders with regard to a primitive constraint involving generality, it employs Mellish 's description identification algorithm. Mellish's algorithm extends Mitchell's version space algorithm in that it not only allows to process constraints fo the type $f \leq p$ and $\neg(f \leq p)$[2] but also handles the dual constraints $p \leq f$ and $\neg(p \leq f)$. Secondly, to update the version space w.r.t. frequency constraints, the integrated algorithm uses a variant of the level-wise algorithm that starts from the given borders rather than from $\top$ or $\bot$.

[1] At this point, we will follow Mitchell's terminology, because he works with two dual borders (a set of maximally general solutions G and a set of maximally specific ones S). In data mining, one typically only works with the S-set.

[2] The positive and negative examples in concept-learning.

3.2 Mellish's description identification algorithm

In order to formulate Mellish's description identification algorithm, we need to introduce some operations on fragments f_1 and f_2.

- $glb(f_1, f_2) = max\{f \mid f_1 \leq f \; and \; f_2 \leq f\}$
  the smallest "merged" fragments
- $lub(f_1, f_2) = min\{f \mid f \leq f_1 \; and \; f \leq f_2\}$
  the largest common subfragments
- $mgs(f_1, f_2) = max\{f \mid f_1 \leq f \; and \; not(f \leq f_2)\}$
  the smallest fragments more specific than f_1 but not more general than f_2
- $msg(f_1, f_2) = min\{f \mid f \leq f_1 \; and \; not(f_2 \leq f)\}$
  the largest fragments more general than f_1 but not more specific than f_2

Notice that these operators may generate more than one fragment; e.g. $lub('o = c = s', 'o = s') = \{'o', 's'\}$. These operations can now be used to instantiate Mellish's algorithm:

$S := \{\bot\}$; $G := \{\top\}$;
for all primitive constraint c **do**
  case c of $p \leq f$:
    $S := \{s \in S \mid p \leq s\}$
    $G :=$ max $\{glb \mid glb \in glb(p, g) \; and \; g \in G \; and$
      $\exists s \in S : glb \leq s\}$
  case c of $f \leq p$:
    $G := \{g \in G \mid g \leq p\}$
    $S :=$ min $\{lub \mid lub \in lub(p, s) \; and \; s \in S \; and$
      $\exists g \in G : g \leq lub\}$
  case i of $\neg(f \leq p)$:
    $S := \{s \in S \mid not(s \leq p)\}$
    $G :=$ max $\{m \mid \exists g \in G : m \in mgs(g, p)$
      $and \; \exists s \in S : m \leq s\}$
  case i of $\neg(p \leq f)$:
    $G := \{g \in G \mid not(p \leq g)\}$
    $S :=$ min $\{m \mid \exists s \in S : m \in msg(s, p)$
      $and \; \exists g \in G : g \leq m\}$

3.3 Variants of the level-wise algorithm

The algorithms outlined below employ refinement operators.

- A refinement operator $\rho_s(f) = max\{f' \in \mathcal{M} \mid f < f'\}$, i.e. extending a fragment by one atom.
- A generalization operator $\rho_g(P) = min\{f' \in \mathcal{M} \mid f' < P\}$, i.e. removing one atom from one side of a fragment.

To deal with the frequency constraints c, we may employ the following generalization of the level-wise algorithm. This is the *downwards* version.

Let c be a constraint of type $freq(f, D) \geq m$
$L_0 := G$; $i := 0$
while $L_i \neq \emptyset$ **do**
  $F_i := \{p \mid p \in L_i$ *and p satisfies constraint c*$\}$
  $I_i := L_i - F_i$ the set of infrequent fragments considered
  $L_{i+1} := \{p \mid \exists q \in L_i : p \in \rho_s(q)$
    $and \; \exists s \in S : p \leq s \; and \; \rho_g(p) \cap (\cup_j I_j) = \emptyset \}$
  $i := i + 1$
endwhile
$G := F_0$
$S := min(\cup_j F_j)$

To explain the algorithm, let us first consider the case where $S = \{\bot\}$ and $G = \{\top\}$. In this case, the above algorithm will behave roughly as the level-wise algorithm. The L_i will then contain only fragments of size i and the algorithm will keep track of the set of frequent fragments F_i as well as the infrequent ones. The algorithm will then repeatedly compute a set of candidate refinements L_{i+1}, delete those fragments that cannot be frequent by looking at the frequency of its generalizations, and evaluate the resulting possibly frequent fragments on the database. This process continues until L_i becomes empty.

The basic modifications to the level-wise algorithm that we made are concerned with the fact that we need not consider any fragment that is not in the already computed version space (i.e. any element not between an element of the G and the S set). Secondly, we have to compute the updated S set, which should contain all frequent fragments whose refinements are all infrequent.

Finding the updated G and S sets can also be realized in the dual manner. In this case, one will initialize L_0 with the elements of S and proceed otherwise completely dual.

The resulting *upwards* algorithm is shown below:

Let c be a constraint of type $freq(f, D) \geq m$
$L_0 := S$; $i := 0$
while $L_i \neq \emptyset$ **do**
  $F_i := \{p \mid p \in L_i$ *and p satisfies constraint c*$\}$
  $I_i := L_i - F_i$ the set of infrequent fragments considered
  $L_{i+1} := \{p \mid \exists q \in L_i : p \in \rho_g(q) \; and$
    $\exists g \in G : g \leq p \; and \; \rho_s(p) \cap (\cup_j F_j) = \emptyset \}$
  $i := i + 1$
endwhile
$G := \{g \in G \mid g \; satisfies \; c\}$
$S := min(\cup_j F_j)$

Whether the top down or bottom up version works more efficiently is likely to depend on the application and query under consideration. At this point it remains an open question as to when which strategy works more efficiently.

Finally, it is also possible to modify the above algorithms (exploiting the dualities) in order to handle *monotonic* frequency constraint of the form $freq(f, D) \leq m$. In this case, one can use the following algorithm (or its dual):

Let c be a constraint of type $freq(f, D) \leq m$
$L_0 := G$; $i := 0$
while $L_i \neq \emptyset$ **do**
  $I_i := \{p \mid p \in L_i$ *and p satisfies constraint c*$\}$
  $F_i := L_i - I_i$ the set of frequent fragments considered
  $L_{i+1} := \{p \mid \exists q \in L_i : p \in \rho_s(q)$
    $and \; \exists s \in S : p \leq s \; and \; \rho_g(p) \cap (\cup_j I_j) = \emptyset \}$
  $i := i + 1$
endwhile
$G := max(\cup_j I_j)$
$S := \{s \in S \mid s \; satisfies \; c\}$

3.4 Optimisations

Various optimisations to the algorithms are possible.

First, though we have adopted the standard level-wise algorithm to search for the borders when handling frequency constraints, it would also be possible to adopt some more recent and more efficient algorithms, such as those presented by [Bayardo, 1998; Gunopulos *et al.*, 1997]. These directly focus on the most specific (the longest) patterns, i.e. the S-set.

Secondly, Apriori-style algorithms can be made more efficient, if elements of one level in level-wise search are combined to give the candidates for the subsequent one. This can also be done for fragments. For instance, if '$o-s-c$' and '$s-c-o$' are known to be frequent at level 3, '$o-s-c-o$' is a candidate for a frequent fragment at level 4. However, several variants (with respect to order) have to be considered; e.g. '$o-s-c$' and '$s-o-c$' can be combined into '$c-s-o-c$' as well as into '$c-o-s-c$'.

Thirdly, we keep track of the fragments in canonical form. As indicated earlier, each fragment is equivalent to its reversal. In the implementation, we use the canonical form of a fragment which is defined as the maximum (w.r.t. a lexicographic ordering) of the fragment and its reversal. The implementation of the operators takes care of this.

Fourthly, one problem with the implementation of the framework for fragments stems from the fact that the bottom $\perp$ is not a "valid" fragment that can be manipulated. In particular, $\rho_g(\perp)$ is not defined. Thus, we cannot search upwards from the set S if $S = \{\perp\}$. Instead, we have to search downwards from G. If however, S is not equal to $\{\perp\}$, then we can process maximum frequency constraints "upwards" starting with S. For the same reason ($\rho_g(\perp)$ is undefined), we cannot start a query with a constraint $\neg(f \leq p)$, where p is a concrete fragment.

3.5 Optimisation primitives

Two primitives that seem especially useful are *minimize* and *maximize*. Indeed, one could imagine being interested in those fragments that satisfy a number of constraints and in addition have maximum frequency on a certain dataset or minimally general. It is easy to extend the framework with primitives $minimize(c, crit)$ and $maximize(c, crit)$ that finds those fragments in $\mathcal{M}$ that satisfy a conjunctive constraint c and that are minimal or maximal with regard to the specified criterion. In this paper we consider only criteria that are monotonic or anti-monotonic (such as frequency and generality).

In order to find the elements with regard to these optimisation primitives, one first computes the S and G sets with regard to c and then selects those elements within S or G (depending on the $crit$) that are minimal or maximal with regard to the criterion.

4 Experiments

In order to validate our approach, we applied it to the Predictive Toxicology Evaluation challenge dataset of Srinivasan et al. [Srinivasan *et al.*, 1999]. This data set consists of over 300 compounds (and takes more than 1 Mbyte of memory in its Prolog encoding) and has been used as a standard benchmark in predictive toxicology and artificial intelligence. In this application, the goal is to discover molecular fragments that are (relatively) frequent in carcinogenic compounds and infrequent in non-carcinogenic compounds. Such activating, toxic fragments are called *structural alerts* in the toxicological literature [Ashby and Patton, 1993]. One interesting question in this context is whether it is possible to rediscover known alerts. In the following, we summarize our experience with the new approach with example queries and systematic experiments.

4.1 Some interesting queries

One open research question in toxicological research is the role of chlorinated compounds in carcinogenicity. In the new framework, an example query concerning chlorinated fragments that are frequent in active compounds and infrequent in inactive ones looks as follows:

$$('cl' \leq f) \wedge (freq(f, Act) \geq 25) \wedge (freq(f, InAct) \leq 5)$$

A related query concerns the existence of activating, non-halogenated fragments:

$$\neg('f' \leq f) \wedge \neg('cl' \leq f) \wedge \neg('br' \leq f) \wedge \neg('i' \leq f) \wedge (freq(f, Act) \geq 25) \wedge (freq(f, InAct) \leq 5)$$

4.2 Quantitative results

To gather more quantitative evidence, we performed systematic experiments in the above domain. From the application side, it is quite clear that structural alerts are relatively rare for carcinogenicity, so that it can safely be assumed that alerts have a frequency of less than 25 in the positive, active compounds. Also, we are not interested in fragments with *any* frequencies in the positive resp. negative examples. Rather, we are seeking fragments that are, statistically significant, over-represented in the active compounds and under-represented in the inactives. Setting the minimum frequency in the actives to 6, 10, 16 and 20, respectively, we apply the χ^2-Test to a 2×2 contingency table with the class as one variable and the occurrence of the fragment as the other one to determine the maximum allowable frequency in the inactive compounds. In this way, we obtain maximum frequency thresholds of 0, 2, 5 and 7, respectively. For instance, we require the minimum to be 6 and the maximum to be 0 for the first experiment.

Methodwise, we performed a comparison of three approaches to the search for these fragments. Each of these approaches consists of two stages: the first stage handling the minimum frequency query (using the first algorithm in Section 3.3), and the second stage handling the maximum frequency query. The three approaches differ in the second stage, dealing with the maximum frequency query.

The first approach is based on version spaces, searching upwards from S (using the dual version of the third algorithm in Section 3.3). In Table 1, the results for this method can be found in the column for up. In contrast, the second one searches downwards, starting from $\top$ until all elements of set G are determined (using the third algorithm in Section 3.3 –

min	max	*up*	*down*	*post*
6	0	353.4	335.0	436.5
10	2	166.7	137.9	163.9
16	5	60.2	61.1	70.6
20	7	49.2	53.8	55.7

Table 1: Runtimes in seconds on a Pentium II.

column *down* in Table 1). The third method performs simple post-processing (column *post* in the table): it filters those fragments that are too frequent in the given dataset. Table 1 summarizes the runtimes of these three methods for the given minimum/maximum frequency parameter settings in seconds CPU time.

The outcome of these experiments is not clear a priori, because additional bookkeeping is done by our Version Space approach. Still, the experiments show that in 7 out of 8 cases, the Version Space approach pays: it outperforms the rather ad-hoc post-processing method in terms of computation time. Another result from the experiments is that – for the given queries – it does not make a big difference whether we search upwards or downwards for maximum frequency queries.

Perhaps the most important outcome of the experiments is the answers to the queries, which are shown below. They indicate that – for the given queries – version spaces constitute indeed a suitable and compact representation for the solution sets to the queries. Also, as outlined above, the computational time needed for answering the queries is reasonable.

```
 6:0: G={c-c-c-c-o-c-c~c}
      S={c-c-c-c-o-c-c~c~c~c~c~c}
10:2: G={c~c-c~c, br,
         c-o-c~c~c~c~c~c-n,c-o-c~c-n}
      S={c~c~c~c~c~c-c~c~c~c~c~c,
         br-c, c-o-c~c~c~c~c~c-n,
         c-o-c~c-n}
16:5: G=S={c-c~c~c~c-n}
20:7: G=S={n-c~c~c~c~c~c-o,
           c-c~c~c~c-n,n-c~c-o}
```

Further results and experiments in the context of feature construction and propositionalization can be found in [Kramer and De Raedt, 2001].

5 Conclusions and Related Work

The presented work contributes to 1) the theory of data mining and machine learning because of its integration of version spaces with the level-wise algorithm, 2) the framework of inductive databases, because the constraints can and should be interpreted as queries in a molecular fragment finding language, and, as discussed above, 3) to molecular fragment finding. We briefly review the key contributions in these domains and relate to relevant work where possible.

With regard to 1) and 2) our work builds on that by [De Raedt, 2000; 1998], who presents an integration of the version space and level-wise algorithms. However, we expand our earlier theoretical work in various respects. Indeed, in contrast to our earlier work, we report on an implementation, experiments that show the validity of the framework and an application in molecular fragment finding. Thus our work provides – for the first time – evidence that the framework is not only of theoretical interest but also effective with regard to applications.

With regard to 1), the presented algorithm provides a generalized theoretical framework for data mining. The resulting framework extends the borders in the levelwise techniques sketched by [Mannila and Toivonen, 1997], who link the level-wise algorithm to the S set of Mitchell's version space approach but do not further exploit the version space model. The experimental evidence indicates that this approach could form a viable extension to the classical level-wise algorithm.

Using two borders (i.e. the version space representation) to characterize the space of solutions to inductive queries is also done by [Dong and Li, 1999]. They use the version space representation to search for emerging patterns. Emerging patterns are defined as itemsets whose supports increase significantly from one dataset to another. Such patterns are also closely related to the significant fragments we discover using the χ^2 test. However, the primitive constraints we support seem to be different from those by [Dong and Li, 1999]. To compute the borders, Dong and Li do not employ the levelwise algorithm. Instead, they rely on more efficient algorithms such as Bayardo's [1998] Max-Miner. In principle it must be possible to adapt these more recent algorithms [Bayardo, 1998; Gunopulos *et al.*, 1997] to our framework too.

For what concerns 2), our work can also be regarded as a domain specific inductive database [Imielinski and Mannila, 1996; Meo *et al.*, 1998]. As sketched by [Han *et al.*, 1999], inductive databases allow the user to specify constraints on the patterns of interest. Recently, many of these constraints have been considered in the data mining literature, cf. e.g. [Ng *et al.*, 1998; Han *et al.*, 2000; 1999]. In this context however, the use of frequency constraints on different data sets seems new.

Finally, let us also note that the presented work on discovering molecular structures and regularities (also using version spaces) is related to the well-known Meta-Dendral system by [Buchanan and Mitchell, 1978].

References

[Agrawal *et al.*, 1993] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1993.

[Ashby and Patton, 1993] J. Ashby and D. Paton. The Influence of Chemical Structure on the Extent and Sites of Carcinogenesis for 522 Rodent Carcinogens and 55 Different Human Carcinogen Exposures. *Mutation Research*, 286:3-74, 1993.

[Bayardo, 1998] R. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1998.

[Buchanan and Mitchell, 1978] B. Buchanan and T. Mitchell. Model-directed learning of production rules. In Waterman, D. A. and Hayes-Roth, F. (Eds.) *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

[Dehaspe and Toivonen, 1999] L. Dehaspe, H. Toivonen. Discovery of Frequent Datalog Patterns, in *Data Mining and Knowledge Discovery Journal*, Vol. 3, 1999.

[De Raedt, 1998] L. De Raedt. An inductive logic programming language for database mining. in *Proceedings of the 4th International Conference on Artificial Intelligence and Symbolic Computation*, Lecture Notes in Artificial Intelligence, Vol. 1476, Springer Verlag, 2000.

[De Raedt, 2000] L. De Raedt. A Logical Database Mining Query Language. in *Proceedings of the 10th Inductive Logic Programming Conference*, Lecture Notes in Artificial Intelligence, Vol. 1866, Springer Verlag, 2000.

[Dong and Li, 1999] G. Dong and J. Li. Efficient mining of emerging patterns : discovering trends and differences. In *Proceedings of KDD*, ACM, 1999.

[Gunopulos *et al.*, 1997] D. Gunopulos, H. Mannila, S. Saluja: Discovering All Most Specific Sentences by Randomized Algorithms. In Foto N. Afrati, Phokion Kolaitis (Eds.): *Database Theory - ICDT '97, 6th International Conference*, Lecture Notes in Computer Science 1186, Springer 1997.

[Han *et al.*, 1999] J. Han, L. V. S. Lakshmanan, and R. T. Ng, Constraint-Based, Multidimensional Data Mining, *Computer*, Vol. 32(8): 46-50, 1999.

[Han *et al.*, 2000] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM SIGMOD Conference on Management of Data* , 2000.

[Hirsh, 1994] H. Hirsh. Generalizing Version Spaces. *Machine Learning*, Vol. 17(1): 5-46 (1994).

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

[Inokuchi *et al.*, 2000] A. Inokuchi, T. Washio, H. Motoda. An Apriori-based algorithm for mining frequent substructures from graph data. in D. Zighed, J. Komorowski, and J. Zyktow (Eds.) *Proceedings of PKDD 2000*, Lecture Notes in Artificial Intelligence, Vol. 1910, Springer-Verlag, 2000.

[Kramer and Frank, 2000] S. Kramer and E. Frank. Bottom-Up Propositionalization. *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 156-162, 2000.

[Kramer and De Raedt, 2001] S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical applications. In *Proceedings of the 18th International Conference on Machine Learning*, Morgan Kaufmann, 2001.

[Mannila and Toivonen, 1997] H. Mannila and H. Toivonen, Levelwise search and borders of theories in knowledge discovery, *Data Mining and Knowledge Discovery*, Vol. 1, 1997.

[Meo *et al.*, 1998] R. Meo, G. Psaila and S. Ceri, An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, Vol. 2, 1998.

[Mellish, 1990] C. Mellish. The description identification algorithm. *Artificial Intelligence*, 1990.

[Mitchell, 1982] T. Mitchell. Generalization as Search, *Artificial Intelligence*, 1980.

[Ng *et al.*, 1998] R. T. Ng, L. V.S. Lkshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of ACM SIGMOD Conference on Management of Data* , 1998.

[Rosenkranz *et al.*, 1999] H.S. Rosenkranz, A.R. Cunningham, Y.P. Zhang, H.G. Clayhamp, O.T. Macina, N.B. Sussmann, S.G. Grant and G. Klopman. Development, Characterization and Application of Predictive-Toxicology Models. *SAR and QSAR in Environmental Research*, 10:277-298, 1999.

[Srinivasan *et al.*, 1999] A. Srinivasan, R.D. King and D.W. Bristol. An Assessment of Submissions Made to the Predictive Toxicology Evaluation Challenge. *Proc. of IJCAI-99*, 270-275, 1999.