Inductive Databases for Bio- and Chemo-Informatics

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 Inductive databases are a new form of databases in which one does not only store data but also patterns or regularities. A user can generate, manipulate and query for patterns of interest using an inductive query language. Data mining then becomes an interactive querying process.

The inductive database framework is especially important for bio- and chemo-informatics because of the existence of a large number of databases and the need to mine them in these research fields. Two examples of inductive databases are presented: MOLFEA and PROFEA. MOLFEA is the Molecular Feature Miner that mines for linear fragments of interest in the 2D-structure of chemical compounds. PROFEA mines for fragments of interest in the secondary structure of proteins. MOLFEA and PROFEA are examples of inductive databases that generate patterns in the form of strings. The underlying principles of this class of inductive databases are presented.

1 Introduction

Over the past decade the fields of bio- and chemo-informatics have witnessed an explosive growth. Many results have been produced and the most important ones include the generation of a vast number of databases. Indeed, the human genome [10], the Protein Data Bank [2], the SCOP [22] and DTP's HIV database (`http://dtp.nci.nih.gov`) are just a few results of the data avalanche that bio- and chemo-informatics have caused. It is well-known that the generation of the data is only the first step in the scientific process that ultimately must lead to a full understanding of the underlying phenomena. The second step – in which computer science plays a central role – is the analysis and mining of the data in order to obtain patterns and regularities of interest. These then have to be filtered and interpreted in scientific terms in a third step in order to arrive at novel scientific knowledge.

Inductive databases [13, 11, 6, 3] provide a novel framework for combining data mining with databases. The key idea is that an inductive database does not only store data but also patterns that describe regularities in the data. In addition, there is an inductive query language which allows the user to declaratively specify the patterns of interest; the inductive database management system is then responsible for generating the patterns that satisfy the constraints imposed by queries. The inductive database framework is extremely attractive for bio- and chemo-informatics because it provides a tool to support scientists in each of the three steps

sketched above. Indeed, our favorite view of an inductive database user is that of a scientist that is interactively querying an inductive database (using a possibly graphical interface), that inspects the resulting patterns in the data, obtains new ideas, and reformulates the original query until the scientifically interesting patterns appear. Note that at present many web services in bio- and chemo-informatics can be queried *either* for data *or* for patterns. However, no tight integration of data and patterns in a common framework and implementation is available.

In this paper, we will introduce inductive databases in which patterns are generalized strings and in which inductive queries are conjunctive constraints. The primitive constraints that will be considered involve generality and frequency. Generality constraints can be used to specify that the patterns of interest are (resp. are not) more general than a specific pattern. For string patterns, the generality relation corresponds to the substring or subsequence relation. The frequency of a pattern in a data set denotes the number of occurrences of the pattern in the data set. Frequency constraints either impose a maximum or a minimum frequency on a data set of interest. Generality and frequency constraints can then be combined in a conjunctive manner. Possibly queries may ask for all patterns that are more general than pattern x, have a minimum frequency m_1 on the data set A (e.g., a set of biologically active molecules), and a maximum frequency m_2 on data set I (e.g., a set of inactive molecules). The result is a flexible and declarative query language to specify the patterns of interest. Furthermore, the inductive queries can be efficiently answered using e.g. the level-wise version space algorithm that we have introduced elsewhere [7, 18].

This general framework for inductive databases over strings is then specialized to obtain two domain-specific inductive databases. The first is the Molecular Feature Miner MOLFEA [18] which finds linear fragments or patterns of interest in chemical compounds. The MOLFEA system has been successfully applied to a number challenging toxicological database (cf. [24, 25, 8]). It is able to discover meaningful fragments in large databases such as the DTP's HIV database (containing over 40,000 molecules, cf. `http://dtp.nci.nih.gov`). The second one is the Protein Feature Miner PROFEA. It discovers common linear fragments in the secondary structure of proteins [9] as specified in the Protein Data Bank PDB [2].

The paper is organized as follows. Section 2 introduces the general framework for inductive databases over strings. Section 3 then presents the Molecular Feature Miner MOLFEA, and Section 4, the Protein Feature Miner PROFEA. Finally, in section 5, we conclude and discuss related work.

2 An inductive database framework for strings

An inductive database consists of two components: a set of data sets $\mathcal{D}$ and a set of pattern sets $\mathcal{P}$. This reflects the fact that any single database may contain information about different data sets. E.g. when doing SAR, one may want to distinguish the active molecules from the inactives; also, there are different classes of proteins, cf. SCOP [22]. In a similar manner, there can be different sets of patterns. Certain sets of patterns may be input by the user, other ones may have been computed on the basis of the formulated inductive queries. In the remainder of this paper, we will make abstraction of the specific operations on (data and pattern) sets that are possible. We assume that a traditional data base language (such as Codd's relational algebra) is available to manipulate these sets.

2.1 Patterns, strings and matching

The patterns that we will consider in this paper are strings. More formally, a pattern is a string belonging to a language $\mathcal{L}$ defined over an alphabet Σ of symbols. Depending on the particular language, the alphabet may contain the special don't care symbol $'*'$. Furthermore, as usual in formal languages, not all patterns in Σ^* will belong to $\mathcal{L}$. Examples of syntactic restrictions on patterns (and hence on $\mathcal{L}$) will be discussed below in the sections on MOLFEA and PROFEA. E.g. if $\Sigma = \{a, b, c, ..., z\}$ then the strings $aabbbcc$ and $aa*bb$ could be members of $\mathcal{L}$.

A *simple* pattern does not contain any don't care symbols. A simple pattern $p \in \mathcal{L}$ *matches* a string s over Σ^* if and only if p is a substring of s, i.e. s can be written as the concatenation of three substrings l, p and r such that $s = lpr$, where l and r denote the left and right substring respectively. E.g. the simple pattern $aaabbbccc$ matches the string $aaabbbcccdddd$, where $l = \epsilon$, the empty string, and $r = dddd$. A don't care symbol has to match exactly one symbol. Therefore, the pattern $a * b * c$ matches the string $aabbccd$ where $l = \epsilon, r = cd$, the first don't care symbol matches a and the second one b.

Although one could – in principle – also employ a notion of matching based on subsequences rather than on substrings, we do not allow for this in this paper because of the increased computational complexity. Using this alternative notion of matching, abc would match $adbdcd$ because the former is a subsequence of the latter.

The patterns in $\mathcal{L}$ are partially ordered by the *is more general than* relation. Formally speaking, a pattern g *is more general than* a pattern s, notation $g \preccurlyeq s$, if and only if all strings matching s also match g. Thus $ab \preccurlyeq abc$ and $a * b * c \preccurlyeq adb * cd$. We will sometimes write $g \prec s$ when g is *strictly* more general than s. There are some interesting properties of the *is more general than* relation:

- Two patterns p_1 and p_2 are equivalent w.r.t. generality only if they are identical.
- There is a unique maximally general element in $\mathcal{L}$, the empty string ϵ, it covers all strings.
- We generally also assume the existence of a unique maximally specific element in $\mathcal{L}$, denoted by ω; by definition, ω covers no example; if such an element does not exist in $\mathcal{L}$, as in the case of strings, we will introduce an artificial one.

The resulting partial order is graphically displayed in Figure 1 for a language based on $\Sigma = \{a, b\}$.

2.2 Constraints on patterns

The task addressed in this paper is that of finding the set of all target patterns $T \in \mathcal{L}$ that satisfy a conjunction of primitive constraints $c_1 \wedge ... \wedge c_n$. The primitive constraints c_i that can be imposed on the unknown target pattern T are:

- $T \preccurlyeq p$, $p \preccurlyeq T$, $\neg(T \preccurlyeq p)$ and $\neg(p \preccurlyeq T)$: where T is the unknown target pattern and p is a specific pattern or string; this type of primitive constraint denotes that T should (resp. should not) be more general (resp. more specific) than the specified pattern p; e.g. the constraint $abc \preccurlyeq T$ specifies that the target pattern T should be a superstring of abc; possible targets satisfying the constraint include $abcd$ and $dddabceee$; $\neg(T \preccurlyeq abc)$ specifies that T should not match abc; solutions for this constraint include d and $*abc$.

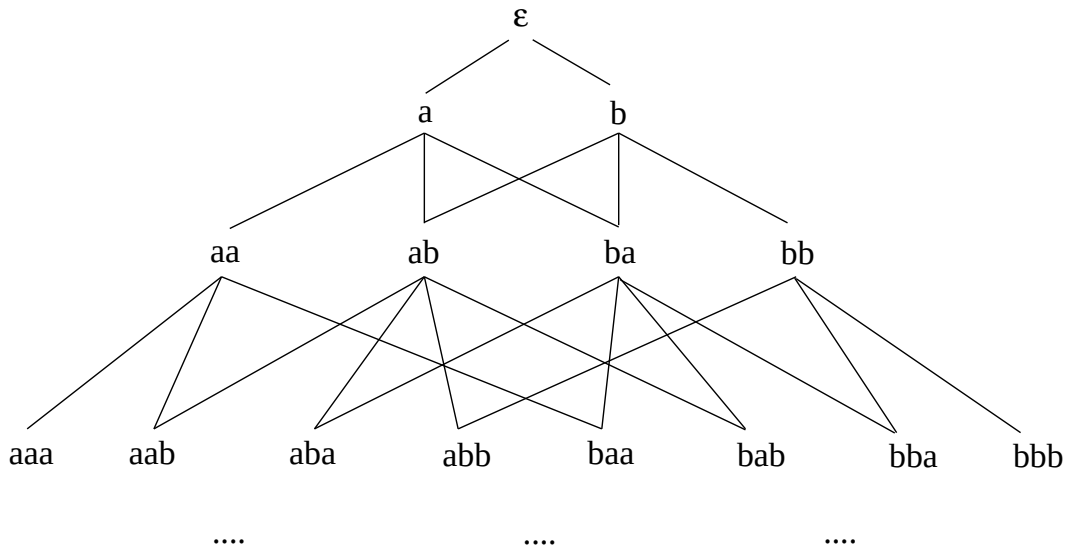

Figure 1: Partial order for a language based on $\Sigma = \{a, b\}$.

- $freq(T, E)$ denotes the frequency of a pattern T on a set of examples E; the frequency of a pattern T on a data set E is defined as the number of examples in E that T matches; e.g. if $E = \{abcd, bcde, cdef\}$ then $freq(bcd, E) = 2$ and $freq(ab, E) = 1$.
- $freq(T, E_1) \leq c_1$, $freq(T, E_2) \geq c_2$ where the c_i are positive integers and E_1 and E_2 are sets of examples; this constraint denotes that the frequency of T on the data set E_i should be less than (resp. greater than) or equal to c_i; e.g. the constraint $freq(T, Act) \geq 100$ denotes that the target patterns T should have a minimum frequency of 100 on the set of active molecules Act.

These primitive constraints can now conjunctively be combined in order to declaratively specify the target patterns of interest. Note that the conjunction may specify constraints w.r.t. any number of data sets, e.g. imposing a minimum frequency on a set of active molecules, and a maximum one on a set of inactives.

As an illustration consider the following data sets:

- $E_1 = \{aabbcc, abbc, bb\}$
- $E_2 = \{abc, bc, cc\}$
- The constraint $(freq(T, E_1) \geq 2) \wedge (freq(T, E_2) \leq 0) \wedge (a \preccurlyeq T)$ would have as solutions abb and $abbc$ if one does not allow for don't cares. abb and $abbc$ match the two first elements of E_1, none of the elements in E_2 and are more specific than a. If don't cares are allowed, other solutions would be possible, such as $a * b$.

3 Solving conjunctive constraints

In this section, we discuss how the solutions to a conjunctive constraint can be computed. This section is slightly more technical and the reader not interested in algorithmic issues might skip this section without loss of continuity.

We will first characterize the set of all solutions $sol(c_1 \wedge ... \wedge c_n)$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$. More specifically, we will show that $sol(c_1 \wedge ... \wedge)$ is completely characterized by its two border sets S and G. S will contain the maximally specific patterns, and G the maximally general ones. We will then also discuss how these border sets can be computed, although we refer for full technical details and algorithms to [7, 18].

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. [19]). A constraint c is *anti-monotonic* (resp. monotonic) w.r.t. generality whenever

$$\forall\, patterns\ s, g : (g \preccurlyeq s) \wedge (s \in sol(c)) \rightarrow (g \in sol(c))$$

(resp. $(g \in sol(c)) \rightarrow (s \in sol(c))$). Anti-monotonic (resp. monotonic) constraints state that whenever a pattern s satisfies the constraint, all its generalizations (resp. specializations) will also satisfy the constraint. The basic anti-monotonic constraints in our framework are: $(T \preccurlyeq p), freq(T, D) \geq m$, the basic monotonic ones are $(p \preccurlyeq T), freq(T, D) \leq m$. Furthermore the negation of a monotonic 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. [19]), where the borders are often denoted by BD^+, as well as the machine learning literature (cf. [21, 20]), 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 P be a set of patterns, then define

$max(P) = \{p \in P \mid \neg\exists q \in P : p \preccurlyeq q\}$, i.e. $max(P)$ contains the maximally specific elements in P. E.g. if $P = \{aa, aab\}$, then $max(P) = \{aab\}$.

$min(P) = \{p \in P \mid \neg\exists q \in P : q \preccurlyeq p\}$, i.e. $min(P)$ contains the maximally general or minimally specific elements in P. E.g. if $P = \{aa, aab\}$, then $min(P) = \{aa\}$.

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

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

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

$$sol(c) = \{p \in \mathcal{L} \mid \exists s \in S(c), \exists g \in G(c) : g \preccurlyeq p \preccurlyeq s\}$$

This last property implies that $S(c)$ (resp. $G(c)$) are proper borders for anti-monotonic (resp. monotonic) 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 $G(c_1 \wedge ... \wedge c_n)$.

Before continuing let us illustrate these important definitions. Indeed, reconsider our earlier example and the constraint $c = (freq(T, E_1) \geq 2) \wedge (freq(T, E_2) \leq 0) \wedge (a \preccurlyeq T)$. First notice that $G((freq(T, E_1) \geq 2)) = \{\epsilon\}$ and $S((freq(T, E_1) \geq 2)) = \{abbc\}$. Secondly, $G((freq(T, E_1) \geq 2) \wedge (freq(T, E_2) \leq 0)) = \{bb\}$ and $S(freq(T, E_1) \geq 2) \wedge (freq(T, E_2) \leq 0)) = \{abbc\}$. Finally, $G(c) = \{abb\}$ and $S(c) = \{abbc\}$ yielding $sol(c) = \{abb, abbc\}$. Notice that in general the border sets may contain more than one element. E.g. $S((freq(T, \{aab, aba, baa\}) \geq 2) = \{aa, ab, ba\}$.

3.2 The algorithms

Elsewhere [7, 18], we have presented the level wise version space algorithm that computes the borders $S(c)$ and $G(c)$ w.r.t. a conjunctive constraint c. The level wise version space algorithm integrates two well-known algorithms. First, there is the level wise algorithm in data mining due to [19] that generalizes the earlier Apriori algorithm by [1]. This algorithm efficiently solves constraints of the type $freq(T, D) \geq x$. It starts its search at the maximally general element, i.e. ϵ, and then searches layer after layer until all elements in the present layer become infrequent. While searching layer n, the algorithm considers only patterns of length n that are relevant. The relevant patterns are those that are potentially frequent. These are generated by joining two frequent patterns of layer $n - 1$. E.g. if abc and bcd are frequent at layer 3, then $abcd$ may be frequent at level 4. If on the other hand, either abc or bcd would be infrequent, then $abcd$ cannot be frequent either. Part of the efficiency of the level wise algorithm comes from the fact that the number of scans of the database is minimized; it is equal to the number of layers searched.

The level wise algorithm has been adapted for use with borders in the level wise version space algorithm [7, 18]. A number of modifications have been necessary. First, the primitive constraints are handled one after the other. Secondly, rather than starting from the single maximally general element ϵ, it may be necessary to start from the elements that are in the G-set already computed from the previous primitive constraints; the elements in G may be more specific than ϵ. Thirdly, while searching for frequent patterns starting from G, one only needs to search those elements that are more general than an element in the already computed S-set. Finally, the level wise version space algorithm does not only cope with minimal frequency constraints (as the level wise algorithm) but also with maximal frequency constraints. The component for dealing with such maximal frequency constraints can be obtained by exploiting the dualities in the problem. Rather than starting from G, specializing and testing for frequency, one must start from S, generalize and test for infrequency.

The second component of the level wise version space algorithm is the candidate elimination algorithm by [21] and its extension, i.e. the description identification algorithm by [20]. This algorithm is a classic one in machine learning and it takes care of the generality constraints. More specifically, in the original setting, Mitchell's candidate algorithm computed the S and G-set w.r.t. a conjunctive constraint in which each of the primitive constraints were of the form $T \preccurlyeq p$ and $\neg(T \preccurlyeq n)$. Constraints of this form correspond to positive and negative examples in concept-learning. Mellish' description identification algorithm then also copes with the dual constraints, i.e. $p \preccurlyeq T$ and $\neg(n \preccurlyeq T)$. Mellish's algorithm is applied $-$ as is $-$ in the level wise version space algorithm.

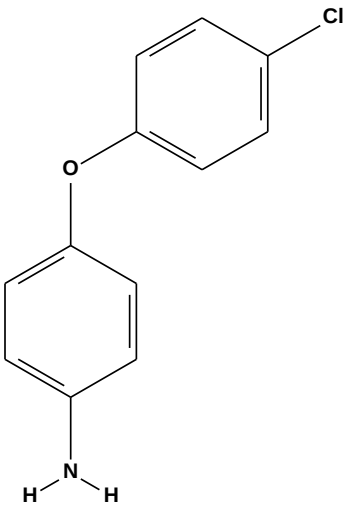

Figure 2: Example compound in a 2-D representation. `'Cl-C:C:C:C-O'` is an example fragment occurring in the molecule.

More details on the resulting algorithm as well as experiments can be found in [7, 18].

4 The Molecular Feature Miner MOLFEA

MOLFEA is an instance of the above sketched inductive database framework. The task addressed by MOLFEA is to find linear patterns or fragments of interest in sets of compounds. An example compound and fragment can be found in Figure 2.

4.1 Molecular Fragments

Basically, a 2-D representation of molecules is employed, and fragments are linear sequences of atoms and bonds. 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". So, fragments are strings over an alphabet consisting of elements and bond types (or bond orders). Don't cares are not allowed in MOLFEA. Furthermore, two syntactically different fragments are equivalent (w.r.t. generality) only when they are a reversal of one another; e.g. `'C-O-S'` and `'S-O-C'` denote the same fragment.

In contrast to the general framework, the *examples* are no longer strings. This necessitates the use of a different coverage relation. A molecular fragment F *matches* an example compound e if and only if F considered as a graph is a subgraph of example e. For instance, fragment `'Cl-C:C:C:C-O'` covers the example compound in Figure 2.

4.2 Implementation

MOLFEA is based on the chemical description languages SMARTS [15] and SMILES [26]. Fragments (see above) are formulated in SMARTS [15]. In other words, the language of fragments is a simple subset of SMARTS. Note that SMARTS would not only allow for fully specified linear sequences of elements, but for arbitrary substructures and abstractions (e.g.,

`'~'` is used to denote *any* type of bond, and `'*'` denotes *any* type of atom). Extensions of MOLFEA including side-chains and abstractions are planned.

Whereas fragments are formulated in SMARTS, chemical compounds are represented by the chemical line notation SMILES. Each chemical compound can be represented as a string in SMILES. The SMILES representation of the compound in Figure 2 is

```
'Nc1ccc(Oc2ccc(Cl)cc2)cc1',
```

or

```
'N-c1:c:c:c(-O-c2:c:c:c(-Cl):c:c2):c:c1',
```

when the bonds are explicitly written. This notation can be understood as follows:

- A SMILES representation is essentially a sequence of atoms and bonds, such as `'O-S-C'`. The elements of aromatic atoms are written in lower-case, otherwise the elements start with a capital.
- The hydrogen atoms and the single and aromatic bonds need – under certain conditions – not be written. They can be computed from the other information.
- Cyclic structures are represented by breaking one bond in each ring. The atoms adjacent to the broken bond obtain the same number. E.g. `'cccccc'` denotes a sequence of aromatic carbons and `'c1ccccc1'` denotes a *ring* of 6 aromatic carbons, a benzene ring. In transcribing Figure 2, the lower ring is identified by number 1, and the upper one by number 2. The first two carbons subsequent to the nitrogen on the right-hand side of the lower benzene ring can be broken and labeled with '1'. Analogously, the upper ring can be encoded.
- Side-structures are written between brackets. E.g. in the example compound, the upper substructure starting from the oxygen atom is written as `'Oc2ccc(Cl)cc2'`. So, we have an oxygen followed by a benzene ring. Furthermore, there is a chlorine bonded to the fourth carbon of the benzene ring.

The final SMILES code for the overall compound in Figure 2 can then be regarded as a nitrogen (hydrogens are implicitly added), followed by a benzene ring which is connected by an oxygen to a chlorinated benzene ring: `'Nc1ccc(Oc2ccc(Cl)cc2)cc1'`.

There are three key advantages of using SMILES for data mining applications in computational chemistry. Firstly, SMILES is a language designed, understood and "spoken" by (computational) chemists. Secondly, SMILES code is extremely compact for storage and manipulation as compared to other representations. Indeed, on the website of Daylight Chemical Information Systems one finds the statement that "a typical SMILES will take 50% to 70% less space than an equivalent connection table." A database of 23,137 structures is said to require only an average of 1.6 bytes per atom. Using Ziv-Lempel compression, this can further be reduced to 0.42 bytes per atom. Thirdly, efficient and optimized tools exist for testing whether a fragment matches a compound. Using SMILES and SMARTS one can rely on the existing state-of-the-art computational chemistry technology.

At the time of writing, MOLFEA is implemented in two versions: One employing the SMILES and SMARTS toolkits from Daylight Chemical Information Systems, and one employing the OpenEye library (OELib, cf. `http://www.eyesopen.com/oelib.html`) as the underlying computational chemistry tool. For matching a fragment against an example compound, an internal data structure provided by these libraries is constructed and used.

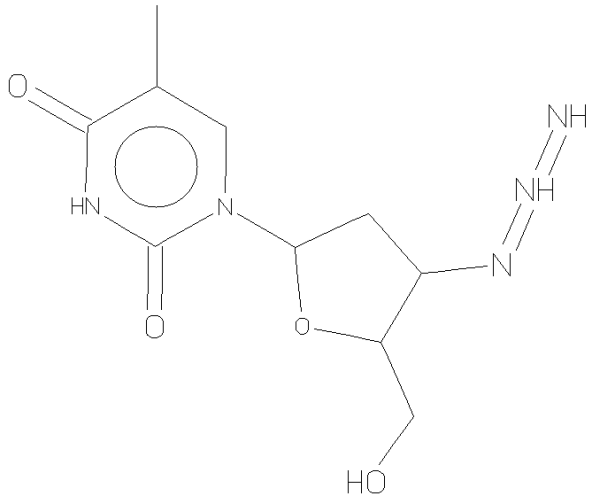

Figure 3: Chemical Structure of Azidothymidine

4.3 Experiments

In this section, we briefly summarize our experiments with a dataset by the DTP AIDS Antiviral Screen program (`http://dtp.nci.nih.gov`) and with structure-activity relationships (SARs) using MOLFEA-generated features. SARs are statistical models that relate chemical structure to biological activity.

DTP AIDS Antiviral Screen database The DTP AIDS Antiviral Screen program has checked tens of thousands of compounds for evidence of anti-HIV activity. The available database (October 1999 Release) contains the screening results for 43,382 compounds. We are not aware of previous data mining attempts in this domain.

The screen utilizes a soluble formazan assay to measure protection of human CEM cells from HIV-1 infection [27]. Compounds were classified as either confirmed active (CA, providing protection), confirmed moderately active (CM, not reproducibly providing protection), or confirmed inactive (CI). In our experiments, class CA consisted of 417 compounds, class CM of 1069 compounds, and class CI of 40282 compounds.

The aim of this experiment was to find fragments that are, statistically significant, over-represented in one class and under-represented in another one. The first task was to compare CA with CI, and the second one to compare CA with CM.

The following queries were posed to the system: (1) $(freq(f, CA) \geq 13) \wedge (freq(f, CI) \leq 516)$ and (2) $(freq(f, CA) \geq 13) \wedge (freq(f, CM) \leq 8)$. The thresholds in the queries were determined as follows: The minimum frequency threshold in these queries corresponds to 3 % of the active molecules. In order to determine the maximum allowable frequency in the non-active molecules, we used the χ^2-Test applied to a 2×2 contingency table with the class as one variable and the occurrence of the fragment as the other one. In this way, we obtained a maximum frequency of 516 in inactive compounds for the first task, and a maximum frequency of 8 in the moderately actives for the second. Given these frequencies, the occurrence of a fragment in the active class is not due to chance at a significance level of 0.999.

For the first task, the total computation time was 19212.31 CPU seconds (measured in CPU seconds on a Linux PC with a Pentium III 600 MHz processor), where the first part (the minimum frequency query) took only 1544.09 CPU seconds, and the second part (the maximum frequency query) took 17668.22 CPU seconds. The boundary set G contained 222 elements, and S contained 314 elements. This contrasts with a total of 1623 patterns in the solution space bounded by G and S, which demonstrates the utility of version spaces in this kind of application.

For the second task, the total computation time was 2054.07 seconds, where the first part took 1532.50 and the second part took 521.57 seconds. The boundary set G contained 110 elements, and S contained 127 elements. Here, the total number of patterns between G and S was 684.

MOLFEA searched for fragments of up to length 25 in the minimum frequency part of the queries. The longest solution fragment was then of length 24 for the first task, and 22 for the second. So, it has been shown that MOLFEA can search for very long patterns in a structural database of over 40,000 compounds.

From the boundary sets G and S, we picked fragments based on their class distribution (statistical significance and accuracy). A majority of these fragments, e.g.

```
'N=N=N-C-C-C-n:c:c:c=O'
```

and

```
'N=N=N-C-C-C-n:c:n:c=O'
```

indicate compounds that are derivatives of Azidothymidine (AZT, Retrovir, Zidovudine, 3'-Azido-3'-deoxythymidine, CAS 30516-87-1, see Figure 3), a potent inhibitor of HIV-1 replication, which is widely used in the treatment of HIV infection. Other fragments indicate another class of reverse transciptase inhibitors, mainly thiocarboxanilide derivatives, which are, according to our knowledge, drugs that are still in an experimental phase. The automated rediscovery of the most important classes of anti-HIV drugs indicates the utility of the presented approach.

Use of MOLFEA **features for SAR** In a series of other experiments [17], we have employed MOLFEA-generated features in structure-activity relationship prediction. SAR prediction in this context works in three steps. In a first step, MOLFEA queries are used to construct a set of patterns. In the second step, one selects the most interesting fragments using certain statistical criteria possibly in combination with syntactic ones (one might have a preference for maximally specific and/or maximally general fragments). In a third step, one employs the resulting fragments as binary features (a fragment either occurs in a molecule or it does not) to describe the molecules and then feeds the resulting data sets into a traditional data mining system (such as WEKA [28]). The result is then a predictive model for the obtained data sets. This method can be combined with virtually any data mining technique; we have induced decision trees, classification rules, as well as Support Vector Machines (SVMs).

In the experiments sketched in [17], we have investigated the effects of class-sensitive vs. class-blind fragment construction in the domains of biodegradability, mutagenicity and

carcinogenicity prediction [8, 24, 25]. Class-sensitive feature construction is performed using combined minimum and maximum frequency queries as described above. Class-blind feature construction is performed by a simple minimum frequency query.

The predictive accuracies turned out to be at least competitive with the best published results in the literature so far. Support Vector Machines were able to take advantage of a large number of features (fragments) constructed in a class-blind manner, whereas classical inductive Machine Learning approaches (decision trees and rules) seemed to benefit from class-sensitive feature construction. Summing up, these experiments clearly demonstrated the utility of MOLFEA-generated features in the induction of SARs.

4.4 Related work

Molecular fragments are, among other purposes, useful and important for the the induction of Structure-Activity Relationships. The use of automatically derived structural fragments in SARs originates from the CASE/MultiCASE systems developed by [23]. 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 a 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 [5] and the approach by Inokuchi *et al.* [14] 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.

5 The Protein Feature Miner PROFEA

In this section, we present PROFEA, another instance of the above framework for inductive databases. PROFEA is an inductive database for the analysis of secondary structures of proteins.

5.1 Data: Secondary Structures from the Protein Data Bank (PDB)

The data used in PROFEA originate from the *Protein Data Bank* (PDB, `http://www.rcsb.org/pdb`), which contains experimentally determined structures of a large number of macromolecules (mostly proteins). For proteins, the PDB comprises information about their primary structure (i.e., the amino acid sequence), their secondary structure (i.e, their structure in terms of higher level elements such as helices and sheets), and their tertiary structure (i.e, the location of each atom in the molecule in terms of three-dimensional coordinates). In PROFEA, we are working at the level of secondary structures. For a brief guide to the

biological terminology, we refer the reader to [12]. The secondary structure of a protein in the PDB is described in terms of their helices, sheets and turns. In the following, we will refer to helices, sheets and turns as secondary structure elements.

An *example* for our inductive database PROFEA is a sequence of secondary structure elements. We adopted the representation from [4]. Thus, we abstracted from the turns, so that secondary structures are represented as sequences of helices and strands (belonging to some sheets).

Helices are represented by expressions `'h_`x`_`y`'`, where $x \in \{1, \ldots, 10\}$ denotes the helix type according to the PDB, and $y \in \mathbb{N}$ equals the number of amino acids of the helix. The helix types are defined as follows: 1 encodes a right-handed alpha helix, 2 a right-handed omega helix, 3 a right-handed pi helix, 4 a right-handed gamma helix, 5 a right-handed 3-10 helix, 6 a left-handed alpha helix, 7 a left-handed omega helix, 8 a left-handed gamma helix, 9 a 2-7 ribbon/helix, and 10 encodes polyproline. For instance, `h_1_14` stands for a right-handed alpha helix of length 14.

Strands are represented similarly: `'s_`x`_`y`'` stands for a strand of orientation $x \in \{-1, 0, +1\}$, and $y \in \mathbb{N}$ is the length of the structure. An orientation of 0 means that we have the first strand of a sheet, orientation 1 denotes a parallel strand, and -1 an anti-parallel strand. This encoding has been adopted from the PDB as well.

Now, the examples are sequences of helices and strands, or from a more abstract point of view, strings. Here are the encodings for a few ribonuclease A proteins:

```
[...]
1rnd(A) 'h_1_10 h_1_9 s_0_4 h_1_6 h_5_3 s_0_2 s_-1_2 s_-1_7
        s_-1_7 s_-1_7 s_-1_8'
3rn3(A) 'h_1_10 h_1_10 s_0_6 h_1_10 s_-1_21 s_-1_16'
1bel(A) 'h_1_8 h_1_7 s_0_4 h_1_8 s_0_2 s_-1_2 s_-1_7 s_-1_7
        s_-1_3 s_-1_4'
1rbx(A) 'h_1_8 h_1_7 s_0_4 h_1_8 s_0_2 s_-1_2 s_-1_7 s_-1_7
        s_-1_3 s_-1_4'
[...]
```

An advantage of the chosen representation is that it is easy to read and handle. An obvious disadvantage is that it does not consider important information such as turns or loops. Other secondary structure elements may be biologically slightly less relevant than helices and strands, but still could be of importance in the search for substructures. The chosen representation also does not consider the assignment of strands to sheets. Consecutive strands of the same sheets are not distinguishable from consecutive strands of different sheets. Furthermore, proteins may consist of several chains. The PDB assigns chain IDs to secondary structure elements in order to enable the identification of parts that belong together. All these pieces of information are not considered in the representation adopted from [4]. In the meantime, we have developed a logical representation of secondary structure data that captures this kind of information. At the time of this writing, we have already performed preliminary experiments with logical hidden Markov models (LOHMMs; cf. [16]) and the first-order sequence miner MINESEQLOG applied to these data.

5.2 Secondary Structure Patterns

Patterns in this domain are, like the examples, sequences of secondary structure elements. Unlike in the small-molecule domain of MOLFEA, we do not have to take care of reversals, because there is a defined order in these sequences. One of the properties of the domain is that the symbols within strings have an internal structure (see above).

In contrast to MOLFEA patterns, PROFEA patterns may consist of *don't cares*, denoted by an asterisk `'*'`. Don't cares can apply to a complete secondary structure element, or to specific "parameters" of a secondary structure element, for instance, its length. E.g., a pattern `'h_1_10 * s_0_6'` covers all proteins with a right-handed alpha helix of length 10 followed by the beginning of a sheet (a strand of length 6), with exactly one arbitrary secondary structure element in between. In contrast, the pattern `'h_*_* s_0_*'` covers all proteins with an arbitrary helix followed by the beginning of a sheet (a strand of arbitrary length). It is also possible to have don't cares in the first position: e.g., `'*_*_5'` denotes an arbitrary secondary structure element of length 5.

5.3 Preliminary Experiments

In this section, we describe preliminary experiments with PROFEA applied to three sets of well-studied proteins: hemoglobin, myoglobin and ribonuclease A. In order to enable a comparison with the results of experiments performed with the SUBDUE system, we adopted the same set-up as in [4]. The goal there was to find *compressive* structural patterns in these three proteins, which were then compared with structures from the rest of the PDB. To this purpose, more than 4,000 proteins were randomly sampled from the rest of the PDB.

In our analysis, we posed the same type of combined minimum/maximum frequency query to PROFEA as above to MOLFEA, in order to find secondary structure patterns that are characteristic for hemoglobin, myoglobin, or ribonuclease A, respectively. The minimum frequency threshold was varied in steps of 5%, the maximum frequency threshold was set arbitrarily to 5. In these preliminary experiments, don't cares were not used (the option was turned off for efficiency reasons). In the following, we present the boundary sets G and S for one of the queries: $(freq(f, RibonucleaseA) \geq 26) \wedge (freq(f, RestPDB) \leq 5)$. The minimum frequency of 26 corresponds to 30% of all ribonuclease A proteins. The boundary set G returned by PROFEA consisted of 13 secondary structure patterns:

```
'h_1_10 h_1_10',  'h_1_10 s_0_3'
's_-1_5 s_-1_3',  's_-1_4 s_-1_8'
's_0_3 s_-1_4',   's_-1_8 s_-1_5'
's_-1_8 s_-1_8',  's_-1_10 s_-1_10'
's_-1_10 s_-1_8', 's_0_3 s_0_3'
'h_1_10 s_0_7',   's_0_7 h_1_10'
's_-1_1'
```

Boundary set S contains, along with two other patterns, one long secondary structure pattern consisting of 13 secondary structure elements:

```
'h_1_10 s_0_3 s_0_3 s_-1_4 s_-1_4 s_-1_8 s_-1_1
 s_-1_10 s_-1_10 s_-1_8 s_-1_8 s_-1_5 s_-1_3'
'h_1_10 h_1_10 s_0_7'
```

```
's_0_7 h_1_10 s_0_3'
```

In our experiments, we found many patterns similar to those discovered by the SUBDUE system [4]. Differences can be explained by the fact that SUBDUE attempts to compress graphs as much as possible, whereas PROFEA can be used, among other things, to search for patterns that are frequent in one class and infrequent in another. So, SUBDUE sometimes returns long patterns that do not cover many examples, but achieve a good compression of the graphs. For instance, some patterns in ribonuclease A molecules occur only twice, but consist of 16 or 17 secondary structure elements. In contrast, PROFEA also finds short patterns, if they satisfy the frequency requirements of the given query.

6 Conclusions

We have presented a novel database and data mining approach based on the concept of inductive databases. Even though our framework was presented for string-like patterns, it should be clear that one could easily adapt it towards richer data structures such as e.g. graphs, or towards other application domains in bio- and chemo-informatics. We have also demonstrated that the inductive database framework is useful for knowledge discovery in databases in general and in bio- and chemo-informatics in particular. The authors hope that the work on MOLFEA and PROFEA will stimulate other researchers to add inductive query languages to the many existing databases in bio- and chemo-informatics. This in turn should allow scientists to understand their data more easily and to discover new knowledge more effectively.

Acknowledgements

This work is partly supported by the European Union IST FET project cInQ. The authors are grateful to Johannes Fischer for implementing PROFEA and to Christoph Helma for his help with the chemical data and interpretation.

References

[1] 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.

[2] H.M. Berman, J. Westbrook, Z. Feng,G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne: The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000. `http://www.rcsb.org/pdb/`

[3] Jean-Francois Boulicaut, Mika Klemettinen, Heikki Mannila: Querying Inductive Databases: A Case Study on the MINE RULE Operator. In *Proceedings of PKDD-98*, Lecture Notes in Computer Science, Vol. 1510, Springer Verlag, pp. 194-202, 1998.

[4] D.J. Cook, L.B. Holder, S. Su, R. Maglothin, and I. Jonyer: Structural mining of molecular biology data. *IEEE Engineering in Medicine and Biology*, special issue on Advances in Genomics, 20(4):67–74, 2001.

[5] L. Dehaspe and H. Toivonen: Discovery of frequent datalog patterns, in *Data Mining and Knowledge Discovery Journal*, 3(1):7–36, 1999.

[6] 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.

[7] L. De Raedt, S. Kramer, The level wise version space algorithm and its application to molecular fragment finding, in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 2001.

[8] S. Džeroski, H. Blockeel, B. Kompare, S. Kramer, B. Pfahringer, W. Van Laer: Experiments in predicting biodegradability. in *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, 80–91, Springer Verlag, 1999.

[9] J. Fischer. *Objektorientiertes Design einer induktiven Datenbank und eine Anwendung des Levelwise Version Space Algorithmus auf die Sekundärstruktur von Proteinen*. Studienarbeit at the Machine Learning Lab, University of Freiburg, Germany, 2002.

[10] sGDB(TM) Human Genome Database. Toronto (Ontario, Canada): The Hospital for Sick Children, Baltimore (Maryland, USA): Johns Hopkins University, 1990-. `http://www.gdb.org/`

[11] J. Han, L. V. S. Lakshmanan, R.T. Ng: Constraint-based, multidimensional data mining, *Computer*, 32(8):46–50, 1999.

[12] L. Hunter: Molecular biology for computer scientists. *Artificial Intelligence and Molecular Biology*, L. Hunter (ed.), 1–46, AAAI Press, Menlo Park, CA, 1993.

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

[14] 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.

[15] C.A. James, D. Weininger, J. Delany: *Daylight theory manual – Daylight 4.71*, Daylight Chemical Information Systems, 2000. `http://www.daylight.com/`

[16] K. Kersting, T. Raiko, S. Kramer, L. De Raedt: Towards discovering structural signatures of protein folds based on logical hidden Markov models, submitted, 2002.

[17] S. Kramer and L. De Raedt: Feature construction with version spaces for biochemical applications. *Proceedings of the 18th International Conference on Machine Learning*, 258–265, Morgan Kaufmann, 2001.

[18] S. Kramer, L. De Raedt, C. Helma: Molecular feature mining in HIV data, in: *Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, 136–143, 2001.

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

[20] C. Mellish: The description identification algorithm. *Artificial Intelligence*, 52:151–167, 1991.

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

[22] A.G. Murzin, S.E. Brenner, T. Hubbard, C. Chothia: SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995. `http://scop.mrc-lmb.cam.ac.uk/scop/`

[23] H.S. Rosenkranz, A.R. Cunningham, Y.P. Zhang, H.G. Clayhamp, O.T. Macina, N.B. Sussmann, S.G. Grant, G. Klopman: Development, characterization and application of predictive toxicology models. *SAR and QSAR in Environmental Research*, 10:277-298, 1999.

[24] A. Srinivasan, S. Muggleton, R.D. King, M. Sternberg: Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996.

[25] A. Srinivasan, R.D. King, D.W. Bristol: An assessment of submissions made to the predictive toxicology evaluation challenge. *Proc. of IJCAI-99*, 270–275, 1999.

[26] D. Weininger: SMILES 1. Introduction and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28, 31, 1988.

[27] O.S. Weislow, R. Kiser, D.L. Fine, J.P. Bader, R.H. Shoemaker, M.R. Boyd: New soluble formazan assay for HIV-1 cytopathic effects: application to high flux screening of synthetic and natural products for AIDS antiviral activity. *Journal of the National Cancer Institute*, 81:577–586, 1989.

[28] I. Witten and E. Frank: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.