Inductive Constraint Logic and the Mutagenesis Problem

Wim Van Laer Hendrik Blockeel
Luc De Raedt
Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
Email:{WimV,Hendrik,LucDR}@cs.kuleuven.ac.be

Abstract

A novel approach to learning first order logic formulae from positive and negative examples is incorporated in a system named ICL (Inductive Constraint Logic). In ICL, examples are viewed as interpretations which are true or false for the target theory, whereas in present inductive logic programming systems, examples are true and false ground facts (or clauses). Furthermore, ICL uses a clausal representation, which corresponds to a conjunctive normal form where each conjunct forms a constraint on positive examples, whereas classical learning techniques have concentrated on concept representations in disjunctive normal form.

We present some experiments with this new system on the mutagenesis problem. These experiments illustrate some of the differences with other systems, and indicate that our approach should work at least as well as the more classical approaches.

1 Introduction

Present work on inductive concept-learning is usually classified as belonging to either the attribute value learning paradigm (see [14, 10, 2], etc.) or the inductive logic programming paradigm [13, 12]. The differences between the two paradigms are due to the representation formalism employed. Although the inductive logic programming paradigm is generally believed to be more expressive, there is no straightforward and generally accepted way to represent an attribute value learning problem as an inductive logic programming task. A possible explanation for this comes from the fact that in attribute value learning examples are true and false interpretations (i.e. models and non-models, or positive and negative examples) of a target theory, whereas in inductive logic programming, examples are true and false facts (or clauses).

Recently, [3] have defined a notion of concept-learning in first order logic, in which examples are true and false interpretations of a target theory, and the target theory is a set of clauses. Each clause in the target theory can be seen as a constraint. These ideas are incorporated in a system named ICL, *inductive constraint logic*. Here, the aim of learning is to induce a set of clauses (or constraints) such that every positive example p is covered by the theory (i.e *all* the clauses are true for this example) and no negative example n is covered (*at least one* of the clauses is false for this example) . This corresponds to learning a conjunctive normal form instead of a disjunctive normal form. Furthermore, as examples are interpretations, most of the other attribute value learning techniques, such as noise handling heuristics, can be nicely upgraded towards the ICL framework.

Some experiments on the mutagenesis problem [16] show that the current implementation of ICL already works quite well (i.e. as well as Progol, and better than FOIL), but also indicate a number of possible improvements to ICL. We illustrate the relationship between ICL's CNF solutions and the classical DNF solutions using the results from these experiments.

The paper is organized as follows: in Section 2, we introduce the inductive constraint logic framework; in Section 3, we introduce the mutagenesis problem; in Section 4, we report experiments on the mutagenesis problem; and finally we conclude in Section 5.

2 Inductive constraint logic

2.1 Definitions and framework

We assume familiarity with first order logic and model theory (see [9, 4] for an introduction).

A first order alphabet is a set of predicate symbols, constant symbols and functor symbols. A clause is a formula of the form $A_1; ...; A_m \leftarrow B_1, ..., B_n$ where the A_i and B_i are logical atoms. An atom $p(t_1, ..., t_n)$ is a predicate symbol p followed by a bracketed n-tuple of terms t_i. A term t is a variable V or a function symbol f immediately followed by a bracketed n-tuple of terms t_i. Constants are function symbols of arity 0. *Functor-free* clauses are clauses that contain only variables as terms.

The above clause can be read as A_1 or ... or A_m if B_1 and ... and B_n. All variables in clauses are universally quantified, although this is not explicitly written. Extending the usual convention for *definite clauses* (where $m = 1$), we call $A_1, ..., A_m$ the *head* of the clause and $B_1, ..., B_n$ the *body* of the clause. A *fact* is a definite clause with an empty body, ($m = 1,\ n = 0$). Throughout the paper, we shall assume that all clauses are *range restricted*, which means that all variables occurring in the head of a clause also occur in its body.

A *Herbrand interpretation* over a first order alphabet is a set of ground facts constructed with the predicate, constant and functor symbols in the alphabet. A Herbrand interpretation I is a model for a clause c if and only if for all grounding substitutions θ of c : $body(c)\theta \subset I \rightarrow head(c)\theta \cap I \neq \emptyset$. We also say that c is true in I, that c makes the interpretation I true, or even that I is a true interpretation for c. If a Herbrand interpretation I is not a model for a clause c, we say that c is false in interpretation I. A Herbrand interpretation I is a model for a clausal theory T if and only if it is a model for all clauses in T. Roughly speaking, the truth value of a clause c in an interpretation I can be determined by running the query $?-body(c), not\ head(c)$ on a database containing I using a theorem prover (such as Prolog). If the query succeeds, the clause is false in I. If it finitely fails, the clause is true.

By now we can intuitively describe the inductive constraint logic setting. One starts from a set of positive and negative examples (i.e. true and false interpretations of a target theory) and aims at finding a clausal theory for which all positive examples are models and none of the negative examples is a model.

A definite clause background theory can be used as follows. Rather than starting from complete interpretations of the target theory, each example E is completed with the background theory B, i.e. by taking the minimal Herbrand model $M(B \cup E)$. Note that the completion need not be computed explicitly, but can be left to the theorem-prover[1]. The minimal Herbrand model of a definite clause theory is the set of all ground facts that are logically entailed by that theory.

Formally, the setting thus becomes:

- **Given**
  - P is a set of interpretations such that for all $p \in P$, $M(B \cup p)$ is a true interpretation of the unknown target theory (positive examples);
  - N is a set of interpretations such that for all $n \in N$, $M(B \cup n)$ is a false interpretation of the unknown target theory (negative examples);
  - a definite clause background theory B;
  - a language L_H which defines the set of a priori acceptable clauses
- **Find**: a clausal theory $H \subset L_H$ such that
  - for all $p \in P$, $M(B \cup p)$ is a true interpretation of H (Completeness);
  - for all $n \in N$, $M(B \cup n)$ is a false interpretation of H (Consistency).

2.2 Examples

In the following, each example will be completed implicitly with the background knowledge. So whenever we say an example e is true (false) for a clause (or theory), we mean that $M(B \cup e)$ is true (false) for that clause (theory).

Our setting is illustrated in Examples 1 and 2 (taken from [3]).

Example 1 *The well-known autolander problem (from the Irvine database) is described by a table (see figure 1) in attribute value representation (only a part is shown).*

This attribute value learning problem can directly be specified in terms of the framework. E.g., the interpretation $I_1 =$ {stability(stab), error(mm), sign(pp), wind(tail), magnitude(strong), visibility(yes)} (last line) corresponds to a positive example, and $I_2 =$ {stability(stab), error(mm), sign(pp), wind(head), magnitude(strong), visibility(yes)} (one but last line in the table) is a negative example. Given this problem (without background knowledge), the ICL technique derives the following clausal theory for the positive class.

[1] Using Prolog, one can assert background theory and interpretation into the knowledge base, and run the query $? - body(c)$, *not* $head(c)$, in order to test whether a clause c makes an interpretation true or not.

stability	**error**	**sign**	**wind**	**magnitude**	**visibility**	CLASS
xstab	*XL*	*nn*	*head*	*Low*	*no*	$\oplus$
...	...	...	...	...	*no*	$\oplus$
xstab	*XL*	*nn*	*head*	*Low*	*yes*	$\ominus$
xstab	...	...	...	...	...	$\ominus$
stab	*LX*	*nn*	*head*	*Low*	*yes*	$\ominus$
stab	*LX*	...	...	...	*yes*	$\ominus$
stab	*XL*	*nn*	*head*	*Low*	*yes*	$\ominus$
stab	*XL*	...	...	...	*yes*	$\ominus$
stab	*MM*	*nn*	*tail*	*Low*	*yes*	$\ominus$
stab	*MM*	*nn*	*tail*	...	*yes*	$\ominus$
xstab	*XL*	*nn*	*head*	*OutOfRange*	*yes*	$\ominus$
...	...	...	...	*OutOfRange*	*yes*	$\ominus$
stab	*SS*	*nn*	*head*	*Low*	*yes*	$\oplus$
stab	*SS*	...	...	*Low*	*yes*	$\oplus$
stab	*SS*	*nn*	*head*	*Medium*	*yes*	$\oplus$
stab	*SS*	...	...	*Medium*	*yes*	$\oplus$
stab	*SS*	*nn*	*head*	*Strong*	*yes*	$\oplus$
stab	*SS*	...	...	*Strong*	*yes*	$\oplus$
stab	*MM*	*pp*	*head*	*Low*	*yes*	$\oplus$
stab	*MM*	*pp*	*head*	*Medium*	*yes*	$\oplus$
stab	*MM*	*pp*	*tail*	*Low*	*yes*	$\oplus$
stab	*MM*	*pp*	*tail*	*Medium*	*yes*	$\oplus$
stab	*MM*	*pp*	*head*	*Strong*	*yes*	$\ominus$
stab	*MM*	*pp*	*tail*	*Strong*	*yes*	$\oplus$

Figure 1: The Autolander input

visibility(no) $\leftarrow$ stability(xstab)
visibility(no) $\leftarrow$ magnitude(out-of-range)
visibility(no) $\leftarrow$ error(xl)
visibility(no) $\leftarrow$ error(lx)
visibility(no);error(ss) $\leftarrow$ sign(nn)
visibility(no);error(ss);wind(tail) $\leftarrow$ magnitude(strong)

Notice that I_1 is true for all clauses, whereas I_2 is false for the last clause and thus for the clausal theory.

Example 2 *The* Bongard Problems *are general problems developed by the Russian scientist M. Bongard in his book* Pattern recognition *(a nice overview can be found in* Gödel, Escher and Bach, *Douglas R. Hofstadter). Each problem exists of 12 figures, six of class $\oplus$ and six of class $\ominus$. One example problem can be found in figure 2, based on the* Bongard Problem 47. *The goal is to discriminate between the two classes.*

Each figure consists of lines, points, squares, triangles, circles, ... which can be white, black, small, large, horizontal, ... So each of these figures can be described by a set of facts. Take for instance the upper left figure in our example problem. It consists of a small triangle, pointing up, which is in a large circle. This figure can thus be described as : I = {triangle(f1), small(f1), up(f1), circle(f2), large(f2), in(f1,f2)}.

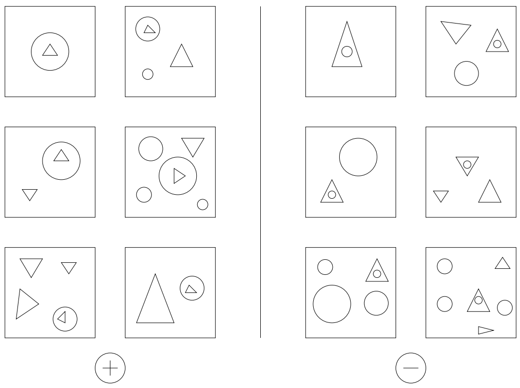

Figure 2: A Bongard Problem

In our setting, each interpretation (example) is a set of facts, describing one figure. The following theory is consistent and complete (found by ICL in +/- 1 sec) :

false $\leftarrow$ circle(Y),in(Y,X)

This constraint says that for each figure of class $\oplus$, there is no circle that is inside another object. This can also be read as : if a figure has a circle inside another object, it is of class $\ominus$.

The view that examples are interpretations and the aim to discriminate between two classes of examples is similar to classical learning from positive and negative examples. This is illustrated by the fact that an attribute value learning task (the Autolander problem in example 1) can be represented in a straightforward way in our framework. Furthermore, for many complex and structural induction problems the view that examples are interpretations is a very natural one, cf. for instance the bongard problems (see example 2), the drug activity problem [5], the mutagenesis problem [16] and Michalski's classical eastbound - westbound train problem [11], which was recently used as the basis for a machine learning competition.

2.3 CNF and DNF Class Descriptions

Classical ILP systems learn a logic program that enables one to discriminate between those examples that belong to some class, and those that do not. The concept description is then usually of the form

$$class(T) \leftarrow l_{1,1} \wedge \cdots \wedge l_{1,n_1}$$

$$class(T) \leftarrow l_{2,1} \wedge \cdots \wedge l_{2,n_2}$$

$$\vdots$$

$$class(T) \leftarrow l_{k,1} \wedge \cdots \wedge l_{k,n_2}$$

so $class(T)$ is described by the following DNF formula:

$$(l_{1,1} \wedge \ldots \wedge l_{1,n_1}) \vee \cdots \vee (l_{k,1} \wedge \ldots \wedge l_{k,n_1})$$

This shows that the above logic program corresponds to a DNF class description. Notice that in the logic program, the variables that do not appear in the head are existentially quantified; therefore, in this DNF too, the variables are existentially quantified (except those that are part of T; these are free in the formula).

In contrast with the previous approach, ICL learns necessary conditions for a class. Suppose ICL finds the following clausal theory H:

$$l_{1,1} \vee \ldots \vee l_{1,n_1} \leftarrow l_{1,n_1+1} \wedge \cdots \wedge l_{1,m_1}$$

$$\ldots$$

$$l_{k,1} \vee \ldots \vee l_{k,n_k} \leftarrow l_{k,n_k+1} \wedge \cdots \wedge l_{k,m_k}$$

This corresponds to the following conjunctive normal form (CNF):

$$l_{1,1} \vee \ldots \vee l_{1,n_1} \vee \neg l_{1,n_1+1} \vee \cdots \vee \neg l_{1,m_1}$$

$$\wedge$$

$$\ldots$$

$$\wedge$$

$$l_{k,1} \vee \cdots \vee l_{k,n_k} \vee \neg l_{k,n_k+1} \vee \cdots \vee \neg l_{k,m_k}$$

Notice that the variables appearing in this CNF are universally quantified.

There exists a simple relationship between CNF and DNF descriptions. Suppose we learn a class $\oplus$. For each positive example, the above CNF formula must be true. This means that whenever the CNF is false in an example, the example is a negative one. The negation of the CNF, is the following DNF:

$$\neg l_{1,1} \wedge \ldots \wedge \neg l_{1,n_1} \wedge l_{1,n_1+1} \wedge \ldots \wedge l_{1,m_1}$$

$$\vee$$

$$\ldots$$

$$\vee$$

$$\neg l_{k,1} \wedge \ldots \wedge \neg l_{k,n_k} \wedge l_{k,n_k+1} \wedge \ldots \wedge l_{k,m_k}$$

Notice that because of the negation, all variables are now existentially quantified. Whenever this DNF is true, the above CNF is false, so the example is negative. The DNF therefore expresses sufficient conditions for the negative class.

We can conclude that for each CNF description for a class, there exists a corresponding DNF description of the complementary class, which has the same structure, but with $\vee/\wedge$, $\exists/\forall$ and positive/negative literals switched. This relationship between CNF and DNF will be illustrated in the mutagenesis experiments.

2.4 Practical aspects

For a complete overview of the algorithm, we refer to [3]. There is a working implementation in ProLog by BIM. ICL uses a covering approach, and uses beam search as search strategy. To traverse the search space, a classical refinement operator under θ-subsumption (Plotkin 70, Shapiro 83) is used together with CLAUDIEN's mechanism to specify the declarative bias (i.e. clause models). Basically, clause models are used to define the syntax of clauses that can appear in the hypotheses. From these models, one can automatically derive a refinement operator that only generates clauses that are allowed by the syntax. A full discussion of this declarative bias mechanism is outside the scope of this paper, but see [1, 18] for more details.

3 The Mutagenesis Problem

A classification problem that has received some attention lately, is that of classifying nitroaromatic molecules into mutagenic and non-mutagenic ones. As in a lot of chemical problems, the data from which to learn are structured. One cannot predict the class of a molecule using only its global properties; properties of the atoms and bonds that occur in it, as well as their relationships, also play a role. Therefore, the problem is particularly suited for ILP learners. Experiments where the ILP-system Progol was compared with statistical learners, neural networks and attribute-value learners [16, 17, 6, 8, 7] have confirmed that Progol indeed performs better than non-ILP systems in several respects (higher accuracy, less complex theories, more easily interpretable rules).

The data for this problem, for each molecule, consist of:

- the atoms that occur in the molecule; each atom is represented by a fact *atom(atom, element, type, charge)* where *atom* is a unique identifier for the atom, *element* is the chemical element the atom is an instance of, *type* is a more specific description of the atom (distinguishing e.g. aromatic carbon atoms from other carbon atoms), and *charge* is the partial charge of the atom.

- the bonds between atoms: denoted as facts *bond(atom1, atom2, bondtype)* where *bondtype* is one of six bond types.

- descriptions of higher-level structures (aromatic rings etc.) that appear in the molecule; these are represented by several predicates, e.g. *benzene([atom1, atom2, atom3, atom4, atom5, atom6])* denotes that the listed atoms form a benzene ring.

- $\log P$: a measure of the hydrophobicity of the molecule; represented by one fact *logp(logp)*.

- ϵ_{LUMO}: the energy of the lowest unoccupied molecular orbital in the molecule; represented by one fact *lumo*(ϵ_{LUMO}).

The last three are properties of the molecule that are expected to have an important influence on its mutagenicity. The first two are, of course, a more objective description of the molecule.

	FOIL	Progol	ICL
accuracy	0.83 (0.03)	0.83 (0.03)	0.84 (0.03)
size	54	11.1	28

Table 1: A comparison of FOIL, Progol and ICL

4 Running ICL on the Mutagenesis Problem

4.1 Comparison with Progol and FOIL

We have performed several experiments with ICL. First of all, we have run it on the same data on which Progol and FOIL have been run (see [15]), to have a means of comparing ICL with these systems. In [15], four different background theories are used, and the results of FOIL and Progol compared. In the experiment we discuss here, we use background $B3$ of the above paper, which consists of knowledge about atoms, bonds, $\log P$ and ϵ_{LUMO}. As ICL learns CNF instead of DNF, we have made it learn the negative class. That way, the rules found by ICL should be comparable to those found by Progol and FOIL.

Table 1 shows the accuracies and sizes of the theories found by Progol, FOIL and ICL.

These are results of a tenfold cross-validation where every molecule has been used nine times in the training set and once in the test set. The accuracy is then computed as $p = C/N$ with C the number of correctly predicted examples and N the total number; the standard deviation (given between parentheses) is $\sqrt{p(1-p)/N}$. The theory size is the average size (i.e. the number of literals) of the 10 theories obtained.

ICL performs 1% better than FOIL and Progol, but this is not significant, so we can say that the performance of ICL is as good as the other systems on this problem. While its theories are more complex than Progol's, they are less complex than FOIL's. Also, ICL does not show the strong bias towards numeric equations that is apparent in FOIL's theories (with this background knowledge, FOIL uses only equations on $\log P$ and ϵ_{LUMO} values in its theory, and no structural conditions; see [15]). The theories found by ICL are therefore closer to Progol's than to FOIL's.

An example theory for the negative class, found by ICL is

false $\leftarrow$ atom(A1,Elem1,27,Charge1),lumo(Lumo),Lumo < -2
false $\leftarrow$ atom(A1,Elem1,29,Charge1),lumo(Lumo),Lumo < -2
false $\leftarrow$ atom(A1,Elem1,195,Charge1)
false $\leftarrow$ atom(A1,Elem1,28,Charge1)
false $\leftarrow$ lumo(Lumo), Lumo < -2, atom(A1,Elem1,1,Charge1)
false $\leftarrow$ atom(A1,Elem1,10,Charge1),Charge1 < 0, lumo(Lumo), Lumo < -1
false $\leftarrow$ atom(A1,Elem1,10,Charge1), atom(A2,Elem2,29,Charge2)
false $\leftarrow$ atom(A1,Elem1,16,Charge1),atom(A2,Elem2,10,Charge2)
false $\leftarrow$ atom(A1,Elem1,27,Charge1), Charge1 < 0

This means that a molecule is not mutagenic if *all* these clauses are true, i.e. if none of the bodies succeed. In other words, a molecule is mutagenic as soon as one of the bodies succeeds, so this theory corresponds to the Progol-style theory

pos ← atom(A1,Elem1,27,Charge1),lumo(Lumo), Lumo < -2
pos ← atom(A1,Elem1,29,Charge1),lumo(Lumo), Lumo < -2
pos ← atom(A1,Elem1,195,Charge1)
pos ← atom(A1,Elem1,28,Charge1)
pos ← lumo(Lumo), Lumo < -2, atom(A1,Elem1,1,Charge1),
pos ← atom(A1,Elem1,10,Charge1), Charge1 < 0, lumo(Lumo), Lumo < -1
pos ← atom(A1,Elem1,10,Charge1), atom(A2,Elem2,29,Charge2)
pos ← atom(A1,Elem1,16,Charge1),atom(A2,Elem2,10,Charge2)
pos ← atom(A1,Elem1,27,Charge1), Charge1 < 0

A theory such as this one might be found by Progol, except for the fact that in Progol a reference to the molecule itself is needed (through an extra variable), while in ICL this is avoided by choosing one interpretation per molecule. This allows for a more natural representation.

4.2 Comparing CNF and DNF

Sometimes a CNF concept description is shorter than its DNF counterpart, and sometimes the opposite is true. This is problem-dependent, and because of the duality between CNF and DNF it is also related to learning the negative class instead of the positive.

We have run ICL on the same background as above, but now learning a CNF for the positive class. The same cross-validation method has been used as in the previously mentioned experiment. In the following table we compare the results:

	CNF	DNF
accuracy	0.84 (0.03)	0.83 (0.03)
size	33	42

This table shows that, while the accuracy of the CNF theory is (almost) as high as that of the DNF theory, it is (on the average) more complex (the difference in size is significant at the 5% level).

An example CNF theory is

false ← atom(A1,Elem1,34,Charge1), Charge1 < −0.4
false ← atom(A1,Elem1,50,Charge1),logp(LP),LP < 3
false ← logp(LP),lt(LP,2),atom(A1,o,Type1,Charge1),Charge1 < −0.4
false ← atom(A1,Elem1,93,Charge1),logp(LP),LP < 4
false ← atom(A1,Elem1,35,Charge1)
false ← logp(LP),LP < 2,atom(A1,f,Type1,Charge1)
false ← atom(A1,Elem1,1,Charge1),logp(LP), LP < 3, atom(A2,Elem2,29,Charge2)

5 Discussion

In this section we discuss the strengths and the weaknesses of ICL, in particular on the mutagenesis problem.

First, notice that ICL is meant to learn binary concepts from examples. As such it addresses the same task as propositional learners (but in a more expressive framework) and it does not learn logic programs from examples as Progol, FOIL and many other ILP systems do, cf. [13]. In these systems, examples are usually true and false ground facts of a target predicate, and the result is a logic program.

Another difference with the majority of ILP systems, is that there is a clear view of how to use examples in heuristics. This is because in ICL one interpretation counts as one example. In many ILP systems (such as FOIL) it is unclear what counts as an example. At some points, FOIL counts facts, and at other points it counts substitutions. We believe this is one of the reasons that heuristics for attribute value learning are much better understood than for ILP. Our framework resolves this confusion and allows to carry over many results on using heuristics in attribute value learning to inductive constraint logic.

A problem with the current implementation of ICL is that its heuristics do not work very well with respect to the addition of literals that do not improve the quality of a clause by themselves, but introduce new variables that can lead to an important improvement in a second step. In these experiments, this is reflected by the fact that bond types are never specified in the theory: before a variable can be required to be a specific bond type, it first has to be introduced by some literal which may not lead to an improvement in itself. This problem does not occur with attribute-value learners, because these simply do not use variables.

A remedy for ICL's "short-sightedness" in this respect is to extend the refinement operator with some sort of lookahead: when a literal is added that introduces a new variable, it may be worthwhile to immediately add other literals that use this variable and to consider these clauses as well when looking for refinements. A change in the heuristics themselves seems less interesting, as one of the main reasons for our approach is exactly the fact that we can borrow good heuristics from attribute-value learners.

The fact that ICL uses a language bias is an advantage in the sense that much faster learning is possible; in these experiments, ICL seems to be about ten times faster than Progol. Of course, this is very dependent on the exact form of the bias. A sensible language bias can be found very easily in this case, but in other cases it is often much harder. We think some research concerning weaker bias specifications (closer to those used in Progol, where only types and modes are entered) would be useful. However, the current bias specification method should not be abandoned, as it provides a very flexible way of entering non-trivial information concerning the search space to the system, which can speed up the search process a lot.

6 Conclusions

We have used a novel approach to learn classification rules in the predicate calculus. Whereas the large majority of results of induction in first order logic views examples as ground facts or clauses, ICL view examples as interpretations. This corresponds to learning a conjunctive normal form instead of a disjunctive one. This technique seems promising for learning classification rules because it inherits many well-known techniques from attribute value learning as well as the much more expressive framework of inductive logic programming.

Practical results with ICL on the mutagenesis problem show that our approach does indeed perform well, but also point out possible improvements. The most important of these is the extension of the refinement operator in such a way that it can add more than one literal when that is appropriate. The addition of some number-handling system, and allowing a weaker language bias specification, are also interesting topics for future work.

Acknowledgements

The authors would like to thank Gunther Sablon for his very helpful comments on a draft version of this paper. Wim Van Laer and Luc De Raedt are supported by the Belgian National Fund for Scientific Research. Hendrik Blockeel is supported by the IWT (Flemish Institute for the Advancement of Scientific Technological Research in Industry). This research is also part of the ESPRIT project no. 20237 on Inductive Logic Programming II.

References

[1] H. Adé, L. De Raedt, and M. Bruynooghe. Declarative Bias for Specific-to-General ILP Systems. *Machine Learning*, 20(1/2):119 – 154, 1995.

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

[3] L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1995.

[4] M. Genesereth and N. Nilsson. *Logical foundations of artificial intelligence.* Morgan Kaufmann, 1987.

[5] R.D. King, S. Muggleton, R.A. Lewis, and M.J.E. Sternberg. Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23), 1992.

[6] R.D. King, M.J.E. Sternberg, A. Srinivasan, and S.H . Muggleton. Knowledge discovery in a database of mutagenic chemicals. In *MLnet familiarisation Workshop: Statistics, Machine Learning and Kn owledge Discovery*, 1995.

[7] R.D. King, M.J.E. Sternberg, A. Srinivasan, and S.H. Muggleton. Relating chemical activity to structure: an examination of ilp successes. *New Generation Computing*, 1995.

[8] R.D. King, M.J.E. Sternberg, A. Srinivasan, and S.H. Muggleton. Representing molecular structure in structure activity relationships: The use of atoms and their bond connectivities to predict mutagenicity using inductive logic programming. *J. Am. Chem. Soc.*, 1995.

[9] J.W. Lloyd. *Foundations of logic programming.* Springer-Verlag, 2nd edition, 1987.

[10] R.S. Michalski. A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume 1. Morgan Kaufmann, 1983.

[11] R.S. Michalski and R.E. Stepp. Learning from observation: conceptual clustering. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume 1. Tioga Publishing Company, 1983.

[12] S. Muggleton, editor. *Inductive Logic Programming.* Academic Press, 1992.

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

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

[15] A. Srinivasan, S.H. Muggleton, and R.D. King. Comparing the use of background knowledge by inductiv e logic programming systems. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming.* IOS Press, 1995.

[16] A. Srinivasan, S.H. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: Ilp experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 217–232. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.

[17] A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg, and R.D. King. Theories for mutagenicity: a study in first-order and feature-based induction. *Artificial Intelligence*, 1995. To appear.

[18] W. Van Laer, L. Dehaspe, and L. De Raedt. Applications of a logical discovery engine. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, pages 263–274, 1994.