Lazy Propositionalisation for Relational Learning

Érick Alphonse and **Céline Rouveirol**[1]

Abstract.
A number of Inductive Logic Programming (ILP) systems have addressed the problem of learning First Order Logic (FOL) discriminant definitions by first reformulating the FOL learning problem into an attribute-value one and then applying efficient learning techniques dedicated to this simpler formalism. The complexity of such propositionalisation methods is now in the size of the reformulated problem which is exponential when tackling non-determinate relational problems. We propose a method that selectively propositionalises the FOL training set by interleaving attribute-value reformulation and algebraic resolution. It avoids, as much as possible, the generation of reformulated examples which are not relevant wrt the discrimination task, and still ensures that explicit correct and complete definitions are learned. We present an AQ-like algorithm exploiting this lazy propositionalisation method and then provide a first empirical evaluation on a standard benchmark dataset for ILP, the Mutagenesis problem.

1 INTRODUCTION

It is a well known fact that concept learning in restrictions of First Order Logic (FOL) has to cope with complexity issues that stem from both the size of the search space and the complexity of the subsumption test in such complex languages.

Both aspects are traditionally handled in Inductive Logic Programming (ILP) directly in a FOL framework by adopting a generate and test approach carefully controlled through sophisticated *language* and *search* bias (see for instance, ij-determination [11], see [12] for an extended study of bias in ILP).

A number of ILP systems (among others, LINUS [8], STILL [17], REPART [21], SP [7]) have adopted an alternative to this approach and have addressed the problem by first reformulating the initial FOL learning problem into an attribute-value or boolean problem and then by applying efficient learning techniques dedicated to this simpler formalism. This representation change, named *propositionalisation* in the remainder is as follows : given a *pattern*, termed P(a formula of the initial FOL search space), FOL examples are reformulated into their multiple matchings with P. The initial FOL learning problem is then described in a tabular representation induced by P.

The advantages of such an approach are that, once the representation change has been performed, well-known robust attribute-value learning techniques may be successfully applied, provided that the discriminant features of the FOL learning problem are preserved by propositionalisation. But, as the subsumption test can be exponential even in restricted of FOL languages such as Datalog, the reformulated training set can be of exponential size [17], as well as highly redundant, and cannot be directly addressed as such for complex relational learning problems like [19]. Stochastic sampling of the matching space [17] or strong bias [8, 20, 21] have been previously applied to build a reformulated learning problem of tractable size that still captures the discriminant features of the initial FOL learning problem.

We present here an alternative method demonstrating that propositionalisation may be directly used in a FOL context. Only a fraction of the positive and negative examples is indeed sufficient to learn a correct and complete FOL definition. An ILP method can efficiently approximate this set, which enables one to have a reformulated learning problem of tractable size. This set of attribute-value instances is then used to efficiently solve the initial ILP problem by applying a data-driven top-down strategy [9]. This strategy allows for focusing the search by dynamically pruning irrelevant branches of the refinement graph.

Our method is based on the notion of *lazy reformulation* of FOL examples into attribute-value vectors that interleaves attribute-value reformulation and algebraic resolution. Information gathered during resolution is used to constrain the propositionalisation process in order to avoid as much as possible the generation of irrelevant reformulated examples. Lazy propositionalisation does not store any negative reformulated instances and only stores a small fraction of positive examples; therefore, it lifts the above mentioned limitations of propositionalisation-based learning techniques. It is presented in this paper independently of the subsumption relation in the original FOL search space and it can be combined with additional bias, which could further improve the overall efficiency of the learning system.

Section 2 introduces the background notions for the paper to be self-contained. Section 3 presents a brief state of the art of ILP propositionalisation based techniques and motivates the need of lazy propositionalisation for learning relational concepts. Section 4 details our lazy propositionalisation based learning algorithm PROPAL. Finally, section 5 sketches results of first experiments with PROPAL, and section 6 concludes on further perspectives for this work.

2 PROPOSITIONALISATION

Given P, a propositionalisation pattern, either built from the training set [16, 7] or provided by the user [20, 21]), each example E described in FOL is reformulated into the set of matchings of P with E. Each matching defines a set of constraints, represented as boolean or valued attributes, which describes some features of the initial FOL problem that we are going to described in the following.

After [15, 21], a FOL learning problem can be decomposed into two sub-problems, i.e., a *relational* (or structural) one and a *functional* one. A relational learning problem is concerned with discrimination by predicate occurrence(s) and variable links (equality/inequality between variables), while a functional learning prob-

[1] Inference and Learning Group, Laboratoire de Recherche en Informatique, UMR 8623 of CNRS, Bâtiment 490, Université Paris-Sud 91405 - Orsay Cedex (France), email : {alphonse,celine}@lri.fr

lem is concerned by discrimination through the values of variables seen as attributes. This imposes a partition on variables that may occur in clauses of the search space: some are referred to as *object* variables the value of which does not carry any discriminant information, the others are *attribute* variables, which are assumed to be functions of subsets of object variables. As a consequence, propositionalisation shifts the original FOL search space into an attribute-value search space in which boolean attributes represent the relational part of the propositionalisation pattern, and valued attributes its functional part (see [15] for details).

In this paper, we will focus on relational learning, which is typically the non-determinate learning problem, in a learning from entailment context [4]. Consequently, we consider a representation shift that transfers the initial FOL search space to a boolean lattice only. We consider, without loss of generality [13] that we are learning wrt an empty background knowledge in a non recursive Datalog language[2], in which a clause possibly contains multiple occurrences of some variables and/or predicate symbols, and with existential variables in the body of the clause[3]. We do not set any restriction on the depth or level of "indeterminacy" of existential variables, as opposed, for instance, to [10, 8]. Lastly, we consider subsumption relations that are weaker or equivalent to θ-subsumption[4]. In this context, we define our *propositionalisation* process as follows.

2.1 Propositionalisation of the training set

Let T_r be the initial FOL training set, represented as a set of Datalog ground clauses, and let $\preceq_r$ be a partial ordering on the FOL search space $\mathcal{L}_r$. The *pattern of propositionalisation* P is a FOL formula of $\mathcal{L}_r$ made of two parts, $\mathcal{P} \cup \mathcal{C}$. $\mathcal{P} = head \leftarrow l_1, \ldots, l_n$ is a definite clause and $\mathcal{C} = c_1, \ldots, c_m$ is a conjunction of constraints between pairs of variables of $\mathcal{P}$[5].

```
propositionalise(e,P)
% returns a boolean vector associated to $e$ given P.
    $\sigma_k := \emptyset$
    $Yet\_Matched := \emptyset; Still\_To\_Be\_Matched := \mathcal{P}$
    Repeat
        Select literal $l_i$ from $Still\_To\_Be\_Matched$
        $Still\_To\_Be\_Matched := Still\_To\_Be\_Matched - l_i$
        if there exists $\sigma$ / $(Yet\_Matched \cup\ l_i)\sigma_k.\sigma \subseteq e$
            then $\sigma_k := \sigma_k.\sigma$; $b_{l_i} := 1$ ;
                $Yet\_Matched := Yet\_Matched \cup l_i$
            else $b_{l_i} := 0$
    until $Still\_To\_Be\_Matched = \emptyset$
    For each $c_j$ in $\mathcal{C}$ do
        if $c_j.\sigma_k$ is satisfied in $e$
            then $b_{c_j} = 1$ else $b_{c_j} = 0$
    return($b_{l_1}, \ldots, b_{l_n}, b_{c_1}, \ldots, b_{c_m}$).
```

Figure 1. Propositionalisation Algorithm

Each literal l_i (resp. c_j) of P defines a boolean attribute b_{l_i}, (resp. b_{c_j}) that will be used to reformulate each FOL example e (positive or negative) of T_r into a set of boolean vectors as detailed in the algorithm of fig. 1. A boolean vector is thus constructed for each substitution σ_k such that P matches a subset of e. For convenience, we will not distinguish in the remainder of the paper a matching (substitution) and its associated boolean vector.

The FOL search space is shifted to a boolean lattice of bottom P. The search space of the reformulated problem is therefore that of concepts more general (or equal) than P under set inclusion.

A FOL example is theoretically to be reformulated, under θ-subsumption and given a pattern P, as the set of $n_j^{m_j}$ boolean vectors, where n_j and m_j are the number of occurrences of predicate symbol p_j in the FOL example e and in P. This combinatoric explosion comes from the fact that, in our target concept language $\mathcal{L}_r$, existential variables non deterministically map on constants of the examples of the training set.

Example 1 *Consider a FOL search space ordered by θ-subsumption, denoted $\preceq_\theta$ in the following. Let E, and E' be two positive examples and CE be a negative example of the target concept, the tabular representation of which is described in figure 2:*

$E' : c(a) \leftarrow p(a,b), q(b), q(a), r(c)$
$E : c(a) \leftarrow p(a,b), p(b,c), q(c), q(a)$
$CE : c(a) \leftarrow p(a,b), p(b,c), q(b), q(a)$

Let us assume that E' has been selected as the seed example, that will be used for constructing the pattern P: $\mathcal{P}$ is chosen here as the maximal variabilization under $\preceq_\theta$ of the seed example E' , i.e., $\mathcal{P} : c(U) \leftarrow p(V,W), q(X), q(Y), r(Z)$. $\mathcal{C}$ is set to the conjunction of equality constraints between pairs of variables of $\mathcal{P}$ which are satisfied in E', i.e., $\mathcal{C} : U = V, U = Y, V = Y, W = X$.

By construction of algorithm in fig. 1, E' is projected to the most specific element of the boolean lattice, σ_P. Let us now consider more closely how example E is reformulated. The propositionalisation process first builds the substitution $\sigma_{E,1} = \{U/a, V/a, W/b, X/a, Y/c\}$. Notice that literal $r(Z)$ of $\mathcal{P}$ has not been matched to any literal of E. Only constraint $U = V$ of P is satisfied for this substitution. When backtracking for another possible matching for literals $q(X)$ and $q(Y)$ onto literals of E, one gets another substitution $\sigma_{E,2} = \{U/a, V/a, W/b, X/c, Y/a\}$, with constraints $U = V, U = Y$ and $V = Y$ of P. Another backtrack on the matching of literal $p(U,V)$ in E yields $\sigma_{E,i} = \{U/a, V/b, W/c, X/c, Y/a\}$, with constraints $U = Y, W = X$ of P. Five other substitutions (and therefore five other boolean vectors) are obtained when backtracking on all possible partial matchings of variables of $\mathcal{P}$ on constants of E.

2.2 New learning task

As pointed out in [20], our learning task is no longer to induce a FOL concept consistent with all positive and negative boolean vectors but is now a *multi-instance problem*[5]:

Definition 2.1 *The reformulated learning task consists in finding a concept that covers for each FOL positive example at least one of its associated boolean vectors (completeness) and none of the boolean vectors associated to any FOL negative example (correctness).*

[20] gives a theorem, which states when the FOL original problem and the reformulated problem, as described above, are equivalent. We reformulate it with our notations.

[2] i.e., a non recursive Horn clause language without function symbols other than constants.
[3] i.e., variables that occur in the body of the clause and not in its head.
[4] Let us recall that in the considered concept language, θ-subsumption is equivalent to logical implication.
[5] Strictly,when learning in a Datalog language, the language of constraints only accepts equalities between variables of $\mathcal{P}$.

P	$c(U)$	$p(V,W)$	$q(X)$	$q(Y)$	$r(Z)$	$U=V$	$U=Y$	$V=Y$	$W=X$
σ_P	1	1	1	1	1	1	1	1	1
$\sigma_{E,1}$	1	1	1	1	0	1	0	0	0
$\sigma_{E,2}$	1	1	1	1	0	1	1	1	0
...									
$\sigma_{E,i}$	1	1	1	1	0	0	0	0	1
...									
$\sigma_{CE,1}$	1	1	1	1	0	1	0	0	1
$\sigma_{CE,2}$	1	1	1	1	0	1	1	1	1
...									
$\sigma_{CE,j}$	1	1	1	1	0	0	0	1	0
...									

Figure 2. The tabular representation of a FOL problem

Theorem 1 *Given a FOL search space $\mathcal{L}_r$, with subsumption ordering $\preceq_r$ and a pattern P (a formula of $\mathcal{L}_r$), searching for a generalization of P under $\preceq_r$ that is a subset of P (up a to variable renaming) is equivalent to searching the reformulated space with P as propositionalisation pattern.*

2.3 Partial ordering on the instance space

A large fraction of the boolean vectors generated by propositionalisation do not directly take part in the process of building a correct and complete discriminant solution (we call them *redundant* in the remainder of the paper) and need not be generated (see for example in fig. 2 $\sigma_{E,1}$ and $\sigma_{CE,1}$ wrt $\sigma_{E,2}$ and $\sigma_{CE,2}$ respectively).

As far as negative examples are concerned, and after [18, 16], there is a partial ordering (*nearest-miss*) of the negative instance space, and it has been shown that only maximally specific negative examples are sufficient for solving the discriminant learning problem. For positive examples, after definition 2.1, a FOL example is covered in the boolean search space if at least one of its corresponding boolean vectors is covered, hence, by transitivity of the subsumption ordering, if one of its more specific associated boolean vectors (dually termed *nearest-hit*) is covered.

The size of the set of non redundant elements of the boolean instance space is upper-bounded by the maximal number of incomparable elements belonging to a boolean lattice, i.e., $\binom{n}{\lfloor n/2 \rfloor}$ with $n = |P|$. As an illustration, if we consider the Mutagenesis dataset, with θ-subsumption as partial ordering, the matching set size is in 40^{40}, to be compared with the number of non-redundant vectors, which is in $\mathcal{O}(2^{40})$. This theoretical upper-bound reflects the inherent complexity of the relational learning process in this kind of learning method.

3 RELATED WORKS

Some ILP systems have addressed the problem of the exponential space complexity of the reformulated problem.

The LINUS system, that first introduced the idea of propositionalisation, and its extended version DINUS [8] handle restricted FOL target concept languages in which the subsumption test is polynomial in the size of clauses. By construction in such languages, each FOL example is reformulated into a single attribute-value vector.

As far as we know, two learning systems have addressed the problem of the intractable space complexity inherent to propositionalisation in the context of non determinate FOL problems: STILL [17] and REPART [21]. For the former, the propositionalisation process is done through a stochastic selection of η example matchings (η is a system parameter) with the pattern, which allows for bounding the size of the reformulated problem, yielding a polynomial generalization process. To offset the imperfection of such generalizations as ”single” classifiers, STILL learns a committee of them (one per example), that classify unseen examples in a nearest neighbor-like way.

For the latter, restriction of the reformulated problem is performed through the choice of a relevant propositionalisation pattern. The user/expert must provide a pattern as a (strong) bias which allows to drastically decrease the matching space. The validity of the method relies on the assumption that the selected pattern preserves the discrimination information sought for.

We present in the next section a lazy propositionalisation method which does not generate the whole reformulated training set (therefore yielding a redundant training set of exponential size), but generates the reformulated problem ”on the fly” during learning. This enables us to generate a reformulated problem of reasonable size, first because it only partially stores positive reformulated instances, and because it empirically avoids the generation of redundant instances.

4 LAZY PROPOSITIONALISATION

Our method is inspired from test incorporation methods [2] and from more recent developments in constraint satisfaction methods, in the sense that information gathered during resolution (i.e., current most general correct definitions) is actively used to avoid generating irrelevant boolean vectors wrt the current discrimination task, such as for instance, boolean vectors that are not covered by any hypothesis maintained by the algorithm. It therefore dynamically exploits:

- information gathered during resolution to only generate boolean examples that are useful for (in)validating the current specialization step
- the partial ordering on the instance space in order to generate useful examples, that is, that are “close to” most specific ones.

In the next section, we present the lazy propositionalisation method used in our system PROPAL. We then discuss the efficiency of this method in terms of how well the set of non redundant vectors is approximated, as this factor obviously has a direct influence on the quality of the algebraic resolution.

4.1 PROPAL's learning algorithm

PROPAL is a relational learning algorithm that uses an AQ-like resolution strategy [9], the inner loop of which is summarized in figure 3. As far as we know, it is the first ILP learning system that learns clausal concepts using a data-driven specialization operator.

```
$P \leftarrow \mathcal{P} \cup \mathcal{C}$ from a seed positive example (see section 2.1)
$G \leftarrow \{\top\}$ (the most general element of the lattice)
**For each** negative relational example $ce$ **do**
    **Repeat**
        Select $g \in G$ incorrect wrt $ce$ (* $\exists \sigma / g.\sigma \prec_r ce$ *)
      **(1)** Compute a boolean vector $b$ from $ce$
          (* $P \prec_b b \preceq_b g$, $b$ as specific as possible *)
      **(2)** Specialize $G$ to discriminate $b$ (* *data-driven strategy* *)
      **(3)** Evaluate each element of $G$ wrt pos. example coverage
          Update $G$ (* *beam search strategy* *)
    **Until** all elements of $G$ reject $ce$
**endfor**
**return** the best element of $G$
```

Figure 3. PROPAL, an AQ-like algorithm for relational learning

Lazy propositionalisation takes place at steps 1 and 3 of the algorithm. For lack of space, we will not describe lazy reformulation of positive examples (step 3). Given an incorrect hypothesis g and a negative relational example ce such that $g\sigma \subseteq ce$, we use the properties of the reformulated problem (see section 2.2) to lazily propositionalise ce. There is a one-to-one mapping between subsets of P (viewed as a FOL formula) and the boolean lattice of bottommost element P (viewed as a boolean vector). We thus know g is incorrect, because there exists a negative vector, more specific or equal to g, that can be extracted from ce. While computing this negative vector, PROPAL can efficiently specialize the set of hypotheses maintained in G (seen as boolean vectors) in order to reject ce [9] (step 2).

We use the fact that $P \prec^{6} b \preceq g$. The boolean vector b is extracted as follows : after computing σ as a matching substitution of the current g with a negative FOL instance ce, PROPAL completes σ by deterministically matching literals of $P - g$ with ce. For this purpose, PROPAL implements a polynomial heuristic search in the matching space based on an efficient graph related subsumption algorithm [14]. We therefore cannot ensure that the extracted boolean vector is a most specific one, which would require an exponential complexity in the P size[7], but is only "close to" a most specific one. Notice that, once b has been rejected, it does not need to be stored.

Example 2 *As an illustration of how* PROPAL *works, let us solve the ILP problem described in section 2.1. Let us assume that the learning algorithm evaluates the incorrect hypothesis* $g : c(U) \leftarrow p(V, W)$, *provided that* $g\sigma \subseteq CE$ *with* $\sigma = \{U/a, V/a, W/b\}$. *g is mapped to the vector* $\{11000000\}$. *Thus, we know g is incorrect because there exists a negative vector, more specific than g, which can be extracted from* CE. *Applying the lazy propositionalisation process, we deterministically complete* σ *with* $\sigma' = \{X/b, Y/b\}$, *yielding* $\sigma_{CE,1} = \{111101001\}$ *(see figure 2). In order to produce refinements of g which no longer cover* $\sigma_{CE,1}$, PROPAL *adds to g one of the following literals or constraints:* $r(Z), U = Y, V = Y$. *Therefore, g is refined into the set of hypotheses:* $\{c(U) \leftarrow p(V, W), r(Z); c(U) \leftarrow p(V, W), q(Y), U = Y; c(U) \leftarrow p(V, W), q(Y), V = Y\}$.

[6] In case $P = b$, there is no discriminant solution.

[7] In fact, extracting a nearest-miss is equivalent to solve the max-clique problem, which is much harder than the subsumption problem.

4.2 Gain of lazy propositionalisation

On the one hand, thanks to the top-down search performed directly in the FOL search space, the number of vectors generated by lazy propositionalisation is in the worst case equal to the size of the matching space searched. This space is induced by literals of g as opposed to P, i.e., by relevant literals wrt the current discriminant task. The expected benefit is theoretically dramatic and becomes larger as the learning description involves multiple predicate occurrences or relations which are not discriminant.

On the other hand, we can ensure that at each step of the search, the generated boolean vectors are necessarily incomparable or more specific than previously generated vectors.

For negative examples, as a result of the specialization strategy (step 2 of the algorithm), new elements of G are necessarily incomparable with b. As a consequence, further extracted boolean vectors are incomparable to b, or more specific than b in case b is not a nearest-miss.
For positive examples, during the (top-down) search, a given boolean vector can take part in several coverage tests wrt positive examples (step 3 of the algorithm). For example, if a lazy propositionalisation of a positive example gives a boolean example equal to the pattern, each element of G during the whole learning process will cover it and the propositionalisation process need not be be invoked for this example anymore. Therefore, PROPAL stores generated boolean positive examples for further coverage tests as long as they are covered by an element of G. Consequently, if a boolean vector is computed during the positive example coverage test for a given positive FOL example, it is necessarily incomparable than any previously generated boolean vector for this FOL example, or more specific in case it is not a nearest-hit.

The quality of approximation of the set of non-redundant vectors has a direct impact on the relevance of the generated refinements, and therefore on the efficiency of heuristic search. As an illustration, if we consider again example 2, the heuristic search in the matching space has reformulated CE into $\sigma_{CE,1}$, which is not the best move. Indeed, would the heuristic search have generated $\sigma_{CE,2}$ (i.e., a near-miss in Winston's terminology), the learning process would have generated a single refinement of g: $c(U) \leftarrow p(V, W), r(Z)$.

5 EXPERIMENTATIONS

We have evaluated PROPAL by performing experiments on the Mutagenesis dataset, a well-known ILP problem used as a benchmark test. In this dataset, each example consists of a structural description of a molecule as a definite clause. The molecules have to be classified into mutagenic and non-mutagenic ones. As our paper focuses on relational learning only, the representation language used has been defined from background knowledge B_1 (see [19] for a deeper explanation, as well as for details of the experimental protocol). In a few words, positive and negative examples of the target concept are molecules described in terms of atoms (between 25 and 40 atoms) and bonds between some of these atoms.

The search space of the algorithm is a Datalog clause space ordered under OI [6], a partial ordering weaker than θ-subsumption, where each matching substitution has to be injective. It has been proved [1] that, given such a partial ordering, the set of solutions of the initial FOL problem is preserved through propositionalisation.

The beam size, the only parameter of the system, was set to 5, its default value. Figure 4 shows PROPAL's performance on the Mutagenesis problem compared with that of FOIL (version 6.2), PROGOL (the Srinivasan's P-PROGOL) and TILDE as reported in [3], averaged over tenfold cross-validation. Time for FOIL and PROGOL were measured on a HP-720, TILDE on a Sun SPARC-20 and PROPAL on Pentium II-350. Because of the different hardware, the learning time should be considered indicative.

	Accuracy (%)	Time (s.)
PROGOL	76	117039
FOIL	61	4950
TILDE	75	41
PROPAL	83	517

Figure 4. Learning results of PROGOL, FOIL, TILDE, PROPAL for the Mutagenesis problem

We can see that the accuracy of the learned theory by PROPAL outperforms the others systems in a reasonable learning time, considering that no user restriction has been set a priori on the search space.

This result demonstrates that lazy propositionalisation allows for learning clausal concepts in a highly non-determinate domain. Considering the good results obtained with a relatively crude search heuristic, this also demonstrates the claim that a data-driven strategy compensates for, or at least has to be used upstream of the complex language and search bias used by PROGOL and TILDE.

6 CONCLUSION

A number of ILP methods have addressed the problem of learning FOL discriminant definitions by first reformulating the FOL learning problem into an attribute-value one and then applying efficient learning techniques dedicated to this simpler formalism.

However, the new learning problem has some inherent properties not suitable for efficiently learning clausal theories from non determinate domains. We have proposed an original propositionalisation method, termed lazy propositionalisation, which interleaves reformulation of FOL examples and resolution step in order to address them.

Such lazy propositionalisation enables us to implement a data-driven learning algorithm, PROPAL, in the spirit of AQ, to tackle the problem of learning relational definitions. This algorithm has been validated on the Mutagenesis problem, a highly non-determinate learning problem used as a benchmark dataset by the ILP community. PROPAL outperforms some state of the art generate-and-test ILP methods when operating on examples described with low level structural information.

This first result is very encouraging and demonstrates the relevance of the approach. Moreover, lazy propositionalisation is adaptable to any subsumption relation in the original FOL search space, and it can be combined with additional bias that can further improve the overall efficiency. For instance, user bias [20, 21] can be incorporated in the pattern definition to further decrease the size of the matching space.

As a perspective, we plan to extend the lazy propositionalisation technique to handle numerical attributes and to improve the heuristic of PROPAL, by integrating heuristics implemented in the latter versions of AQ. These steps will allow us to perform a better empirical evaluation of the method.

ACKNOWLEDGEMENTS

We wish to thank M. Sebag for the many discussions about STILL and propositionalisation methods in general.

REFERENCES

[1] E. Alphonse and C. Rouveirol, 'Object identity for relational learning', Technical report, Deliverable LRIc(1), ESPRIT LTR 20237 (ILP2), (1999).

[2] J. S. Bennett and T. G. Dietterich, 'The test incorporation hypothesis and the weak methods', Technical Report CSTR-86-30-4, Oregon State University, Department of Computer Science, (1986).

[3] H Blockheel and L. De Raedt, 'Top-down induction of first order decision trees', *Artificial Intelligence*, **101**, 285–297, (1998).

[4] L. De Raedt, 'Logical settings for concept learning', *Artificial Intelligence*, **95**, 187–201, (1997).

[5] T. Dietterich, R. Lathrop, and T. Lozano-Perez, 'Solving the multi-instance problem with axis-parallel rectangles', *Artificial Intelligence*, **89**, 31–71, (1996).

[6] F. Esposito, A. Laterza, D. Malerba, and G. Semeraro, 'Refinement of datalog programs', in *Proc. of the MLnet Familiarization Workshop on Data Mining with Inductive Logic Programming (ILP for KDD)*, pp. 73–94, (July 1996).

[7] S. Kramer, B. Pfahringer, and C. Helma, 'Stochastic propositionalization of non-determinate background knowledge', in *Proc. of the 8th International Workshop on Inductive Logic Programming*, ed., D. Page, pp. 80–94. Springer Verlag, (1998).

[8] N. Lavrač and S. Džeroski, *Inductive Logic Programming : techniques and Applications*, Ellis Horwood, 1994.

[9] R. S. Michalski, 'A theory and methodology of inductive learning', in *Machine Learning: An Artificial Intelligence Approach*, eds., R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, volume I, pp. 83–134, Palo Alto, CA, (1983). Tioga.

[10] S. Muggleton, 'Inverse entailment and PROGOL', *New Generation Computing*, **13**, 245–286, (1995).

[11] S. Muggleton and C. Feng, 'Efficient induction in logic programs', in *International Workshop on Inductive Logic Programming*, ed., S. Muggleton, 281–298, Academic Press, (1992).

[12] C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend, 'Declarative bias in ILP', in *Advances in Inductive Logic Programming*, ed., L. De Raedt, 82–103, IOS Press, (1996).

[13] C. Rouveirol, 'Flattening and saturation: Two representation changes for generalization', *Machine Learning*, **14**(2), 219–232, (1994).

[14] T. Scheffer, R. Herbrich, and F. Wysotzki, 'Efficient θ-subsumption based on graph algorithms', in *Proceedings of the 6th International Workshop on Inductive Logic Programming*, ed., S. Muggleton, pp. 312–329. Stockholm University, Royal Institute of Technology, (1996).

[15] M. Sebag, 'Resource bounded induction and deduction in fol', in *Multi Strategy Learning*, (1998).

[16] M. Sebag and C. Rouveirol, 'Constraint inductive logic programming', in *Advances In Inductive Logic Programming*, ed., L. De Raedt, 277–294, IOS Press, (1996).

[17] M. Sebag and C. Rouveirol, 'Tractable induction and classification in first order logic via stochastic matching', in *15th Int. Join Conf. on Artificial Intelligence (IJCAI'97), Nagoya, Japon*, pp. 888–893. Morgan Kaufmann, (1997).

[18] B. D. Smith and P. S. Rosenbloom, 'Incremental non-backtracking focusing: A polynomially bounded generalization algorithm for version spaces', in *Proceedings of the 8th National Conference on Artificial Intelligence*, ed., T. S. W. Dietterich, pp. 848–853, Hynes Convention Centre, (July–August 1990). MIT Press.

[19] A. Srinivasan, S. Muggleton, and R.D. King, 'Comparing the use of background knowledge by inductive logic programming systems', in *Proceedings of the 5th International Workshop on Inductive Logic Programming*, ed., L. De Raedt, pp. 199–230. Scientific Report, Departement of Computer Science, K.U.Leuven, (1995).

[20] J.-D. Zucker and J.-G. Ganascia, 'Changes of representation for efficient learning in structural domains', in *Proc. of 13th International Conference on Machine Learning*. Morgan Kaufmann, (1996).

[21] J.-D. Zucker and J.-G. Ganascia, 'Learning strcutural indeterminate clauses', in *Proc. of the 8th International Workshop on Inductive Logic Programming*, ed., D. Page, pp. 235–244. Springer Verlag, (1998).