Integrating Explanatory and Descriptive Learning in ILP

Yannis Dimopoulos
Institut für Informatik
Universität Freiburg
Am Flughafen 17
79110 Freiburg, Germany
yannis@informatik.uni-freiburg.de

Saso Džeroski
Dep. of Intelligent Systems
Jozef Stefan Institute
Jamova 39, SI-1000 Ljublijana,
Slovenia
Saso.Dzeroski@isj.si

Antonis Kakas
Dep. of Computer Science
University of Cyprus
P.O. Box 537, CY-1678
Cyprus
antonis@turing.cs.ucy.ac.cy

Abstract

A learning framework that combines the two frameworks of explanatory and descriptive Inductive Logic Programming (ILP) is presented. The induced hypotheses in this framework are pairs of the form (T, IC) where T is a definite clausal theory and IC is a set of integrity constraints. The two components allow us to combine complementary information from the same data by applying both explanatory and descriptive learning methods. This non-trivial integration is achieved using a nonmonotonic entailment relation for the basic notion of coverage in the combined language of rules and constraints where the constraints can restrict the conclusions derivable by the rules.

We present a semantics for the new framework and then discuss different cases where combining information from explanatory and descriptive ILP could be useful. We present some basic algorithmic frameworks for learning in the new framework, and report on some preliminary experiments with encouraging results.

1 Introduction

Inductive logic programming (ILP, [Muggleton and De Raedt, 1994]) is concerned with learning clausal theories in first-order logic. Two main approaches exist to learning in first-order logic, known under the names of explanatory and descriptive learning. The first [Muggleton, 1995] is also called learning from entailment or normal ILP, the second [De Raedt and Dzeroski, 1994] is also called learning from interpretations or nonmonotonic ILP. The first setting is concerned with the induction of rules that explain (correctly classify) the given observations, whereas the latter is concerned with the induction of constraints that describe the (dependencies in the) given observations.

While attempts have been made to relate the two settings (see e.g. [Muggleton and De Raedt, 1994]), very few (if any) attempts have been made to combine the two frameworks. On the other hand, the combination of rules and integrity constraints is a very powerful modeling or representation framework found in several areas of Computer Science. It forms for example the basic conceptual model of databases where the data must always conform to the properties that the integrity constraints of the database specify. In Artificial Intelligence the combination of a theory with constraints has been extensively studied (e.g. [Poole, 1988; Kakas *et al.*, 1993]) and applied to problems of planning, diagnosis, legal reasoning and many others.

In this paper we bring together work from the areas of Machine Learning (ILP) and Knowledge Representation to propose an integrated learning framework that synthesizes in a non-trivial way the two separate approaches of explanatory and descriptive learning. In this framework the induced hypothesis is a pair of a definite clausal theory (set of rules), T, and a set of integrity constraints, IC. This induced theory is used to reason in a non-monotonic way, where the integrity constraints in IC specialize the rules in T by restricting their conclusions. The integration of the two forms of learned output in T and IC becomes non-trivial by this reasoning. In turn, when this is used as the basic notion of coverage for the learning problem the two processes of explanatory and descriptive learning interact in a strong way.

The integrated learning framework allows us to combine together (during the learning process) complementary information that can be learned from the same data using the two different approaches of explanatory and descriptive learning. In a typical situation, the rules generated by the explanatory learning process can be (informally) understood as sufficient information on the concepts to be learned whereas the output of descriptive learning as necessary information. This combination of information is particularly useful when the learning data or the learning method is incomplete. The integrated learning framework can thus enhance our learning capabilities by allowing us to learn inherently non-monotonic theories (which by definition can not be completely specified). More importantly though it can also be particularly useful (enhancing) for many other learning problems not necessarily non-monotonic in nature as learning by its very nature is often an incompletely specified problem. At the more practical level, the integration of a set of integrity constraints in the learned theory : (i) provides additional complementary information, (ii) offers a new type of specialization operator on the rules of the theory, (iii) offers a complementary notion of classification via the satisfaction of the constraints and (iv) allows us to learn from incomplete data, e.g., from positive data only where the constraints provide implicitly negative data.

Although there exist several learning systems [De Raedt and Bruynooghe, 1993; De Raedt and Van Laer, 1995; Muggleton, 1995] that learn (or can learn) integrity constraints none of these does so in a strongly integrated fashion as we are proposing here. In fact, most of the constraints learned in practical domains (see e.g. [De Raedt and Dehaspe, 1996]) have the form of definite clauses (the same form as for explanatory ILP), and are even used in the same fashion i.e. for explanation rather than description as is their natural role.

2 An Integrated Learning Framework

In this section we present the basic learning framework that integrates the two different problems of explanatory and descriptive learning. We will assume that the reader is familiar with basic notions of first order logic and logic programming (see e.g. [Lloyd, 1987]).

In the integrated framework the hypothesis space is extended to accommodate the learned output for either setting of explanatory or descriptive learning.

Definition 1 (*Hypothesis Language*): A hypothesis H is a triple $< T, C, IC >$, where C is a set of predicate symbols (the concepts to be learned), T is a Definite Horn theory (of rules defining concepts in C) and IC is a set of first order formulae (integrity constraints on the theory T).

The integrity constraints in IC can in general be any first order formula but in practice (see e.g. [De Raedt and Bruynooghe, 1993]) it is useful to restrict these to be clauses which are also range-restricted. We have also restricted here the theory T to contain only definite Horn clauses with no negation as failure (NAF). The framework can easily be extended to allow NAF. Moreover, the language of definite rules with integrity constraints subsumes NAF [Eshghi and Kowalski, 1989].

The integration of the two forms of learned output (explanatory rules in T and general regularities of the data in IC) becomes non-trivial by combining together the reasoning made with each part and then using this as the basic notion of coverage for the learning problem. This combined notion of entailment is inherited here from work in the area of Knowledge Representation [Poole, 1988], [Kakas *et al.*, 1993]. Reasoning with the theory is done in the usual way using the minimal Herbrand model semantics. For the integrity constraints there are several alternative semantics. In this paper we will use the epistemic or meta-level view [Reiter, 1988] where the constraints are understood as statements at a different level from those in the theory and they specify what must be true of the theory. They are formalized as first order formulae that must be true in any given model of the theory for this model to be accepted.

Definition 2 (*Entailment in the Hypothesis Language*): Let $H =< T, C, IC >$ be a hypothesis. A Generalized model, M, of H is any maximal subset of the minimal Herbrand model, N, of T such that:
(i) $N - M$ contains only atoms with predicates in C
(ii) $M \models IC$
(iii) M is grounded, i.e., $\forall A \in M : T \cup (M - \{A\}) \models A$, where $\models$ denotes truth in the minimal Herbrand model[1].
A ground atom, A, is entailed (covered) by H ($H \vdash A$) iff $M \models A$ for every generalized model M of H.

Informally, this definition allows a conclusion from the theory T only if this does not violate the integrity constraints in IC. The conclusions of the theory T must respect the general properties specified by the integrity constraints, otherwise they are rejected. In other words, the rules of the theory T for the concepts to be learned in C are assumed to be correct unless their conclusions violate the general properties expressed by IC.

We now formulate the general learning problem that can be addressed within the integrated setting of explanatory and descriptive ILP.

Definition 3 (*Learning Problem*):
Given:
(a) background knowledge, B, (definite Horn theory)
(b) a set, C, (possibly empty) of concept predicates to be learned
(c) training data, E, on the predicates of C, containing a set of positive examples E^+ and negative examples E^-, such that $B \cup E$ is consistent.
Find: a hypothesis, $H =< B \cup T, C, IC >$, such that:
(a) $B \cup E^+ \models IC$,
(b) H covers e^+, for every e^+ in E^+, and
(c) H does not cover e^-, for every e^- in E^-,
where $\models$ denotes truth in the minimal Herbrand model.

We often require non-triviality or minimality of the

[1]This technical condition is required for the cases of multiple or recursive predicate learning

learned theory H. There are also several ways in which this problem could be generalized or modified. For example, the background knowledge may already contain some integrity constraints. Another alternative relates to how strong we want the exclusion of negative examples to be. We may for example want to replace the third condition by the stronger condition "there is no generalized model M of H such that $M \models e^-$ for any $e^- \in E^-$". The proper study of these extensions and alternatives is beyond the scope of this paper.

The above definition combines elements from the explanatory and descriptive learning problems via the coverage relation of definition 2. The rules in the theory T give sufficient conditions for the concept(s) that we are learning whereas the integrity constraints in IC provide complementary necessary conditions on the concept.

Let us illustrate the potential and possible use of the integrated framework with a few simple examples. As a first example we take a well known domain from knowledge representation which is not possible to express using only the language bias of definite Horn logic.

Example: Consider the background theory B
$bird(x) \leftarrow penguin(x)$
$penguin(x) \leftarrow superpenguin(x)$
$bird(a), bird(b), penguin(c), penguin(d)$,
$superpenguin(e)$, $superpenguin(f)$
with the set of concepts to be learned $C = \{flies(-)\}$. Consider also the training data $E = E^+ \cup E^-$ consisting simply of a set of positive and negative examples as follows: $E^+ = \{flies(a), flies(b), flies(e), flies(f)\}$ and $E^- = \{flies(c), flies(d)\}$.

A solution to this problem is given by the hypothesis $H =< B \cup T, C, IC >$ where $T = \{flies(x) \leftarrow bird(x)\}$ and $IC = \{superpenguin(x) \leftarrow flies(x), penguin(x)\}$. The integrity constraint expresses a general regularity that characterizes the available data is also used to compensate for the overgenerality of the learned rule in T. The conclusion $flies(c)$ is not possible as the constraint is violated: in $B \cup T$ we would have $penguin(c)$ and not $superpenguin(c)$. On the other hand, the conclusion $flies(e)$ is allowed by the learned theory since this does not violate the constraint. Note that if the integrity constraint is considered simply as another clause in the theory T then the resulting hypothesis would be inconsistent covering negative examples.

This learning problem can also be captured using the non-monotonic construct of NAF by extending the language bias of the theory T from definite Horn clauses to normal logic program clauses. A possible solution is given by the rules: $flies(x) \leftarrow bird(x)$,*not* $abnormal(x)$ and $abnormal(x) \leftarrow penguin(x)$,*not* $superpenguin(x)$.

Negation as failure can easily be captured within the framework of definite rules and integrity constraints [Eshghi and Kowalski, 1989]. Hence the integrated framework subsumes the use of NAF in learning systems. Conversely, it is indeed possible to simulate the effect of most forms of integrity constraints using NAF. This, however, would typically require the use of auxiliary new predicates, as in the above example, thus making the learning process unnecessarily more complicated with the need of predicate invention [Bain and Muggleton, 1990]. A more important difference is the fact that the use of NAF alone still relies only on the explanatory (normal) ILP learning criteria. In the integrated framework in addition we can use the second independent learning criterion of describing the data available. This can be significant particularly in problems where the training data is incomplete. In the above example, if we are not given the negative examples the integrated framework can still find the same solution as before since the integrity constraint can be learned independently from the rest of the problem.

In the previous example due to the inherent non-monotonicity of the problem it is not possible to learn the correct theory using each of the two ILP settings separately. We will now consider some examples where although there are solutions in these separate settings these may be difficult to find. Lack of complete information in learning can effectively result in a situation similar to that of a non-monotonic problem.

Let us first consider an example where we are trying to learn the concept *daughter* given complete background information on $parent, father, mother, male$ and $female$ but possibly incomplete information on *daughter*. Suppose that we have learned the rule
$daughter(x, y) \leftarrow parent(y, x)$.
We first note that the usual specialization of this rule by adding the extra condition of $female(x)$ can also be equivalently achieved by learning the integrity constraint
$female(x) \leftarrow daughter(x, y)$.
In fact, any condition in a rule can be simulated by a new constraint of the general form *condition* $\leftarrow$ *concept*, *sub-body-of-rule*. Note though that this alternative way of specializing the rule is also possible even when there is insufficient negative information in the training data to drive the first alternative of explicit specialization of the rule. Suppose now that the background knowledge does not contain the predicate $female$. An explicit specialization of the rule can now only be done using the NAF condition $not(male(x))$. This can again be equivalently achieved via the integrity constraint $\neg(daughter(x, y), male(x))$ but as before this constraint can exist independently of any negative examples.

The information provided by the integrity constraints may not be captured by the rules of the theory and is thus complementary to that of the rules. Suppose for example that we do not have any direct information on the predicates $male$ and $female$ and we have learned the rule $daughter(x, y) \leftarrow parent(y, x)$. It is still possible to specialize this rule with the use of integrity constraints. This can happen using other information available in the background knowledge and/or the training data that we learn (using the descriptive criteria) is necessarily related to the concept of daughter. Consider for example that the background knowledge also contains information about the predicates $child(-, -)$ and $sister(-, -)$. Then we could learn the integrity constraint: $sister(x, z) \leftarrow daughter(x, y), child(z, y), z \neq x$. This will provide (some) specialization to the general rule excluding all persons which are not sisters to their sibling from being classified as daughters. Similarly other integrity constraints such as $mother(x, z); aunt(x, z) \leftarrow daughter(x, y), grandparent(y, z)$ etc could provide additional specialization of the rule.

This type of specialization comes from learned dependencies in the data relating the concept we are trying to learn with other known relations in the background knowledge. This information encoded by the integrity constraint should not necessarily be seen as part of the rule definition of the concept. It is instructive to compare the behavior of the integrated framework with that of normal ILP in these situations. Firstly, for the latter framework to capture this type of information it will need explicit negative information to drive this specialization, e.g., a negative example of a sibling that is not a daughter. If this is given then normal ILP will try to specialize the general rule by adding extra conditions in its body. In the example above it will specialize the rule to $daughter(x, y) \leftarrow parent(y, x), child(z, y), z \neq x, sister(x, z)$ resulting in an arguably artificial definition.

Apart from the non-naturality of the rules another important difference of encoding this type of information in rules instead of integrity constraints is the fact that the extra conditions added to the rules (e.g., here $child(z, y), z \neq x$) now become necessary for the conclusion to hold whereas this is not the case with the integrity constraints. But these conditions may not be relevant in all cases. They may not be known by the theory (e.g., in the case where we have multiple predicate learning and child is another predicate that we are learning) or they simply do not hold (e.g., cases where the daughter does not have a sibling). This then means that in addition, the normal ILP system must now generate another rule (or rules) to re-cover all the positive examples that are lost by these extra conditions. This is the phenomenon of rule splitting in normal ILP systems that can result in many overspecific rules containing conditions not directly relevant to the proper definition of the concept.

Learning integrity constraints differs significantly from learning more clauses in the theory. The integrity constraints are not simply some extra clauses of the theory learned on some additional concepts that (possibly) appear in the heads of the integrity constraints. Although the integrity constraints could be used to provide a partial definition for these predicates their main purpose is to specialize the concept rules in the theory T. To be more specific, if the above integrity constraint is considered simply as another clause in the theory for a new concept $sister$, this will not have any effect on the rule of the theory which will remain overgeneral. Similarly, if we add the "corresponding clause"
$daughter(x, y) \leftarrow sister(x, z), child(z, y)$
to the theory, this will not have any specialization effect on the other rule for daughter. The choice of the integrity constraints is not independent from the rules of the theory and part of the difficulty is to find the "relevant" constraints that would compensate correctly for the rules of the theory.

Summarizing, the integrity constraints can sometimes provide implicitly the required specialization of the defining rules without the need of explicit negative training data. They effectively form additional implicit training data for the explanatory part of the problem allowing us to learn from positive data only. In addition, the integrity constraints offer new possibilities of specialization of the rules when these express regularities of the concept with other predicates not directly involved in the definition of the concept.

3 Learning algorithms and Experiments

Several approaches to developing learning algorithms in the integrated framework are possible. One is to separate the generation of the two components (rules or constraints). A top-level description of an algorithm that generates the rules first, which then drive the rest of the learning process, is:

Algorithm 1
Find a set of rules R that cover all +ve examples;
Decide-bias-of-ics($R, E; Bias$);
Generate-ics($Bias, E; IC$);
Choose-ics($IC, R, E; C$);
Return (R, C);

After the first step that generates the rules of the theory, the algorithm goes into a constraint generation and selection phase. We first select the general IC schema(ta) (or bias) that will be given as input to the descriptive inductive procedure. We have identified two types of analysis that help decide on the bias (a) analysis of the form of the rules (b) analysis on the examples misclassified. Other information that helps in finding an appropriate schema for the constraints is their complexity (roughly the number of literals they involve) and

more importantly their independence from the existing rules (i.e. they should not be subsumed by the rules). In the experiments we report below we have restricted this kind of analysis to constraints of the form $literal1 \leftarrow concept, literal2$ or $literal1 \leftarrow concept, sub\text{-}body\text{-}of\text{-}rule, literal2$ where $literal1$ is a literal which can be the "false" literal and $literal2$ is a conjunction of up to 2 literals.
Then we call a descriptive system that generates constraints with the decided bias. In our experiments we have used the CLAUDIEN system which offers a strong declarative language for expressing the bias.
The last step in the constraint computation phase is to choose among the constraints produced by the descriptive system, those that will be actually added to the theory. Here we use a combination of the usefulness of the constraints in specializing the theory (the number of negative example misclassifications they correct) together with descriptive criteria, e.g., the degree of applicability in the data.

In algorithm 1 above, the generation of rules is done outside the process of generating the ICs of the theory. In an "interleaved" approach the generation of rules and ICs are combined together and one dynamically influences the other. Constraints again offer an extra possibility of specialization.

Algorithm 2
$IC := \emptyset$; $Rules := \emptyset$;
repeat
Find-rule(R);
While (R, IC) covers E^-
Extended-Specialization($R; R', NIC$);
$R := R'$; $IC := IC \cup NIC$;
$Rules := Rules \cup R$;
until all positive examples are covered;
Return $(Rules, IC)$.

The procedure Extended-Specialization(R; R', NIC) can use either the normal specialization step of adding a new literal in the body of R or call on a descriptive ILP algorithm to generate more constraints NIC. As above this will involve deciding on the bias of constraints to look for and choosing amongst the generated ones. The form of the integrity constraints can now be more specific relating strongly to the rules they are trying to specialize. The general bias of the constraints again follows the schema $literal1 \leftarrow concept, literal2$ where $literal1$ can be the "false" literal.

Another possibility is to generate (and select) in a first phase the integrity constraints and then use these in a second phase of rule generation. The integrity constraints are used primarily as additional (negative) training data and thus this strategy is particularly appropriate for problems where there is no or limited explicitly given negative training data. A top level shell for such an algorithm is the following:

Algorithm 3
Generate a set of ICs C;
(use descriptive criteria e.g. degree of applicability)
repeat
Generate rule R;
While ($covers(R, E^+) >$ thresh.) and not $satisf(R, C)$
Specialize-rule(R);
$R' := R' \cup R$;
until all positive examples are covered;
C' := set of ICs violated by the generated rules;
Return (R', C').

Note that here specialization of the rules does not come form the negative examples only, but also from the ICs that are violated. These constraints force the specialization of a rule until either some minimum threshold on the number of positive examples that it covers is reached or the integrity constraints are indeed satisfied by all the consequences of the generated rule. If the constraints can not be fully satisfied within the threshold, they are returned in the output. The Generate-rule and Specialize-rule are the usual procedures from top-down normal ILP algorithms.

3.1 Initial Experiments

We report here on some initial experiments with integrated learning, the main purpose of which was to provide an initial confirmation of the theoretical ideas and a basis for the future development of a system for integrated learning.

Several experiments were carried out on the real-life problem of characterizing river water quality. The task is to interpret a biological sample taken from a river in terms of five quality classes (see [Dzeroski *et al.*, 1994]). A fact of the form $family(X)$ (resp. $family(X, A)$) indicates that the bioindicator $family$ is present in sample X (resp. at abundance level A). The five classes are denoted $class0(X)$ to $class4(X)$. Several machine learning systems have been applied on this problem, including CN2 [Clark and Boswell, 1991], CLAUDIEN [De Raedt and Bruynooghe, 1993] and GOLEM [Muggleton and Feng, 1990]. Making use of abundance level data CN2 achieves accuracy of 0.62 on unseen cases, while without abundance levels it achieves 0.64 accuracy. CLAUDIEN and GOLEM were only used to generate rules from the entire dataset. In the following, a theory correctly classifies a sample if it predicts the correct class, and no other class is predicted by it. This is different from the classification procedure in CN2, which adds the numbers of examples of each class covered by each rule applicable to an example. The majority class is predicted by CN2 if no rule applies.

Starting from a set of rules that GOLEM induced from data with abundances, we have tried (following algorithm 1) to specialize and complement these rules with integrity constraints. In fact, in this process a different analysis for the different classes is appropriate. We concentrated only on the first three classes, since GOLEM rules for the last two classes are not available. The bias for constraints for class1 was generated by analyzing the actual GOLEM rules of class0 and class1. Four constraints, e.g., $ancylidae(X) \leftarrow class1(X), perlodidae(X)$, were selected which can correct the misclassification of class0 examples as class1 examples while at the same time not affecting significantly the correct classification of class1 examples. In contrast, for class2 constraints, the bias was found by analyzing the actual misclassifications of other examples as class2. The results were similar to that for class1. Overall, a small number of constraints were generated and the performance of the integrated theory was comparable to that of the rules alone with just a marginal improvement. This was expected due to the very specific nature of the GOLEM rules. The point to note though is that the constraints learned form complementary information to the rules as they do not affect the coverage of the correct class examples. In fact, some of these constraints were useful in the other two experiments.

Experim. 1	Accur.	Experim. 2	Accur.
Original CN2 Rules	0.53	Original GOLEM Rules	0.80
CN2 Rules without NAF	0.43	GOLEM rules without abund.	0.22
CN2 Rules without NAF + ICs	0.59	GOLEM rules without abund. + ICs	0.72

Table 1

The next two experiments (see table 1), that also concern the first three classes, involve learning a theory from training data with information only on the presence or absence of each family (no abundance levels). In the first experiment we started with the CN2 rules whose accuracy was found to be 0.53 on unseen cases (compare this to the 0.64 accuracy of the same rule set under the CN2 classification procedure) and removed all NAF literals form these to produce an overgeneral theory. The task was then to specialize these rules with appropriate constraints by following algorithm 1 and applying the same types of analysis as above for generating constraint bias. It was again possible to learn an integrated theory whose accuracy (0.59 on testing examples) is comparable to that of the full CN2 rules with NAF although this time more constraints were necessary.

In the second experiment, we removed the abundance information from the GOLEM rules discussed above to produce an overgeneral set of rules. Examples of these rules are: $class0(X) \leftarrow chironomidae(X), heptageniidae(X), perlodidae(X)$ for class0, $class1(A) \leftarrow ancylidae(A)$, $erpoddellidae(A)$, $hydropsychidae(A)$ for class1 and $class2(A) \leftarrow asellidae(A)$, $erpoddellidae(A)$, $physidae(A)$. To specialize these rules, we used some of the constraints from the first experiment, together with some new constraints that were added because negative examples were still covered by the theory. Examples of these constraints are: $false \leftarrow class1(X)$, $leuctridae(X)$, $physidae(X)$ and $lymnaeidae(X) \leftarrow class1(X)$, $asellidae(X)$, $gammaridae(X)$, $corixidae(X)$ restricting class1 and $false \leftarrow class2(X)$, $heptageniidae(X)$, $perlodidae(X)$ and $baetidae(X) \leftarrow class2(X)$, $hydrobiidae(X)$, $leptoceridae(X)$. Most of the new constraints were selected based on the number of misclassifications they correct. The resulting theory has accuracy 0.72 on unseen cases. This compares well with the accurary of 0.80 of the original GOLEM rules as these rules were generated from the whole data including the testing data.

The overall conclusions drawn from these experiments is that it is indeed possible to generate complementary information in the form of constraints with comparable accuracy as that of a given explanatory theory. Furthermore, it is possible to learn simpler overgeneral explanatory rules, which when augmented with constraints in an integrated theory have comparable and sometimes better accuracy.

4 Conclusions and Related work

We have presented a new framework for learning that integrates explanatory and descriptive learning. This framework allows us to synthesize complementary information from the same learning data by applying interactively both explanatory and descriptive learning methods and is particularly useful when the given learning problem is incompletely specified.

The need to strongly integrate learning of rules and integrity constraints (ICs) was proposed recently in [Dimopoulos and Kakas, 1996] in their study of the relationship of abductive and inductive reasoning. As mentioned though in the introduction, ICs have been often used in explanatory ILP. The interactive theory revision system CLINT [De Raedt, 1992] introduced the use of ICs in explanatory ILP. Intermediate revisions of the theory are tested for consistency with the ICs, and further revised if found inconsistent.

Most explanatory ILP systems that learn starting from an empty theory on the target predicate(s) do not make use of ICs. The MONIC-SKILit system [Jorge and

Brazdil, 1996] is a notable exception. Clauses proposed for addition to the theory are checked for consistency with the ICs. This is done in a stochastic fashion: a number of consequences of the theory are randomly generated; if none of these violate the ICs the theory is considered consistent.

PROGOL [Muggleton, 1995] learns from entailment. While it has been mostly used to learn definite clauses, it can also learn integrity constraints in the form of denials. CLAUDIEN ([De Raedt and Bruynooghe, 1993], [De Raedt and Dehaspe, 1996]) learns from interpretations: it takes as input a set of interpretations P and learns a set of constraints C that hold in each of the given interpretations. ICL [De Raedt and Van Laer, 1995] takes an additional set of interpretations N requiring that at least one constraint in IC is violated by each interpretation in N.

To sum up, some explanatory ILP systems can take into account ICs given by the user. These are used in the learning phase, but seldom (if ever) together with the learned rules of the theory. On the other hand, ILP systems that can learn constraints do not use them (if at all) as constraints in their coverage (classification) relation as we are proposing in this paper.

Several issues remain for future work. On the theoretical level the relationship with nonmonotonic reasoning needs to be explored further to study problems of learnability and compactness of the learned theory in this new framework. On the practical level, the development of an integrated learning system is needed for the application of the framework to other real life problems.

Acknowledgments

Part of this work was supported by the European project ESPRIT IV No 20237 ILP^2 and the Graduiertenkolleg "Menschliche und maschinelle Intelligenz", University of Freiburg.

References

[Bain and Muggleton, 1990] M. Bain and S. Muggleton. Non-monotonic learning. In J.E. Hayes-Michie and E. Tyugu, editors, *Machine Intelligence 12*, 1990.

[Clark and Boswell, 1991] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. Fifth European Working Session on Learning*, pages 151–163, 1991.

[De Raedt and Bruynooghe, 1993] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proc. 13th International Joint Conference on Artificial Intelligence*, pages 1058–1063, 1993.

[De Raedt and Dehaspe, 1996] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 1996. To appear.

[De Raedt and Dzeroski, 1994] L. De Raedt and S. Dzeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.

[De Raedt and Van Laer, 1995] L. De Raedt and V. Van Laer. Inductive constraint logic. In *Proc. Sixth International Workshop on Algorithmic Learning Theory*, pages 80–94, 1995.

[De Raedt, 1992] L. De Raedt. *Interactive Theory Revision: an Inductive Logic Programming Approach.* Academic Press, 1992.

[Dimopoulos and Kakas, 1996] Y. Dimopoulos and A. Kakas. Abduction and inductive learning. In L. De Raedt, editor, *Advances in Inductive Llogic Programming*, pages 144–171, 1996.

[Dzeroski *et al.*, 1994] S. Dzeroski, L. Dehaspe, B. Ruck, and W.J. Walley. Classification of river water quality data using machine learning. In P. Zannetti, editor, *Computer Techniques in Environmental Studies V Vol. I: Pollution modelling*, pages 129–137, 1994.

[Eshghi and Kowalski, 1989] K. Eshghi and R. A. Kowalski. Abduction compared with negation by failure. In *Proc. 6th Inter. Conference on Logic Programming*, pages 234–255, 1989.

[Jorge and Brazdil, 1996] A. Jorge and P. Brazdil. Integrity constraints in ILP using a Monte Carlo approach. In *Proc. 6th International Workshop on Inductive Logic Programming*, pages 137–151, 1996.

[Kakas *et al.*, 1993] A. Kakas, R. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[Lloyd, 1987] J. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1987.

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

[Muggleton and Feng, 1990] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proc. First Conference on Algorithmic Learning Theory*, pages 368–381, 1990.

[Muggleton, 1995] S. Muggleton. Inverse entailment and PROGOL. *New Generation Computing*, 13:245–286, 1995.

[Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Reiter, 1988] R. Reiter. On integrity constraints. In *Proc. of TARK'88*, 1988.