# Using Abductive Logic Programming as a Representation Formalism for Inductive Logic Programming

Fabrizio Riguzzi
DEIS, Università di Bologna, Viale Risorgimento 2
I-40136 Bologna, Italy
Tel. +39 51 6443033, Fax. +39 51 6443073
{friguzzi}@deis.unibo.it

## 1 Introduction

In this paper we summarize the work done by the nodes of Bologna and Cyprus on the use of Abductive Logic Programming (ALP) as a representation formalism for Inductive Logic Programming. In this case, both the background knowledge and the target program are abductive logic programs and the coverage of examples through the resolution proof procedure of Prolog is replaced by coverage through an abductive proof procedure.

The learning problem of ILP is substituted by the Abductive Concept Learning (ACL) problem defined by Dimopoulos and Kakas [2]:

**Given**
a set $\mathcal{P}$ of possible abductive programs
a set of positive examples $E^+$,
a set of negative examples $E^-$,
an abductive theory $AT = \langle T, A, IC \rangle$ as background theory.
**Find**
A new abductive theory $AT' = \langle T', A, IC' \rangle \in \mathcal{P}$ such that
$\forall e^+ \in E^+$, $AT' \vdash_A e^+$,
$\forall e^- \in E^-$, $AT' \not\vdash_A e^-$.

where $AT' \vdash_A e$ means that $e$ is abductively provable from $AT'$, i.e., there exist an abductive explanation for $e$ from $AT'$. We say that $AT'$ *abductively covers* $e$.

The use of ALP as a representation formalism has two main advantages: the first is the possibility of learning in the presence of incomplete knowledge and the second consists in learning theories for abductive reasoning. As regards learning in the presence of incomplete knowledge, abduction can be used to hypothesize information that are needed for the learning process but are not available. We can use abduction to assume a fact that is missing in the background knowledge and that, if present, would allow to cover a positive example or to rule out a negative one. Moreover, we can use abduction to hypothesize new examples that are needed for the coverage of examples of the same predicate (in the case of recursive clauses) or of other predicates (in the case of multiple predicate learning).

Let us show how we can learn from an incomplete knowledge by means of an example (inspired to [5]):

$$
\begin{array}{ll}
T = & \{parent(john, mary) \\
 & parent(david, steve), \\
 & parent(kathy, ellen), female(kathy)\} \\
A = & \{male, female\} \\
IC = & \{\leftarrow male(X), female(X)\} \\
E^+ = & \{father(john, mary), father(david, steve)\} \\
E^- = & \{father(kathy, ellen), father(john, steve)\}
\end{array}
$$

In this case, the theory

$$father(X, Y) \leftarrow parent(X, Y), male(X).$$

does not cover any positive example because the facts $male(david)$ and $male(john)$ are absent from the background knowledge. Therefore, most ILP algorithm would fail to learn such a theory. However, an algorithm that integrates abduction and induction would learn the previous theory by making the assumptions $\Delta = \{male(john), male(david)\}$, thus successfully solving the ACL problem.

The possibility of learning theories for abductive reasoning has been shown in [3, 8] where a theory for default reasoning through abduction is learned.

The first intensional algorithm for integrating abduction and induction was proposed in [3] and was successively refined in [5, 8, 10, 6]. The algorithm proposed does not solve the full ACL problem but solves an intermediate version of ACL called I-ACL [6] in which the conditions on the negative examples is relaxed to

$$AT' \models_A not\ E^-, \text{ where } not\ E^- = \{not\ e^- | e^- \in E^-\}.$$

Instead of requiring that each negative example is not abductively entailed by the learned theory, we require that the negation of each negative example is abductively entailed by the theory. This means that we can make assumptions in order to avoid the coverage of negative examples.

The integration of abduction and induction in extensional ILP systems has been investigated in [9], while in [7] the authors demonstrate how abduction can be used to introduce some of the advantages of extensionality in intensional ILP systems.

The intensional algorithm I-ACL, defined in [6], extends the basic top down intensional ILP algorithm by using the abductive proof procedure defined in [4] for the coverage of examples. I-ACL is able to learn rules containing abducibles but not new integrity constraints. An extension of the algorithm for learning constraints has been proposed in [5] and consists in employing a non-monotonic learner, such as CLAUDIEN [11] or ICL [13], to learn integrity constraints starting from the input data of the learning problem plus the output data, i.e. the learned theory and the literals abduced for covering examples. The constraints learned should ensure that the more restrictive condition of ACL on negative examples is verified by prohibiting the abduction of literals that can justify negative examples. This extension is still in an early phase of development and is the subject of current work.

The first experiments on the application of the I-ACL algorithm on incomplete data has been reported in [6]. The performances of I-ACL are compared to those of ICL-Sat [12] on the multiplexer data [1]: I-ACL generates more accurate theories than ICL-Sat in all the three experiments considered. At the moment, I-ACL is being applied on data from market research questionnaires in which the incompleteness is due to unanswered questions or to do not know answers.

In conclusion, the use of ALP as a representation formalism for ILP has been investigated. The utility of this formalism for learning in presence of incomplete knowledge seems promising. An algorithm for an intermediate version of ACL has been implemented and it is currently being experimented on real cases. An algorithm able to perform full ACL is the next goal in this line of research.

# References


[1] W. Van de Velde. Idl, or taming the multiplexer problem. In Morik K., editor, *Proceedings of the 4th European Working Session on Learning*. Pittman, 1989.

[2] Y. Dimopoulos and A. Kakas. Abduction and Learning. In *Advances in Inductive Logic Programming*. IOS Press, 1996.

[3] F. Esposito, E. Lamma, D. Malerba, P. Mello, M.Milano, F. Riguzzi, and G. Semeraro. Learning Abductive Logic Programs. In *Proceedings of the ECAI96 Workshop on Abductive and Inductive Reasoning*, 1996.

[4] A.C. Kakas and P. Mancarella. On the relation between Truth Maintenance and Abduction. In *Proceedings of PRICAI90*, 1990.

[5] A.C. Kakas and F. Riguzzi. Abductive Concept Learning. Technical Report TR-96-15, University of Cyprus, Computer Science Department, 1996.

[6] A.C. Kakas and F. Riguzzi. Learning with abduction. In *ILP97*, 1997.

[7] E. Lamma, P. Mello, M. Milano, and F. Riguzzi. Integrating extensional and intensional ilp systems through abduction. In *Proceedings of the 7th International Workshop on Logic Program Synthesis and Transformation (LOPSTR'97)*, 1997.

[8] E. Lamma, P. Mello, M. Milano, and F. Riguzzi. Integrating induction and abduction in logic programming. In P. P. Wang, editor, *Prooceedings of the Third Joint Conference on Information Sciences*, volume 2, pages 203–206, 1997.

[9] E. Lamma, P. Mello, M. Milano, and F. Riguzzi. Introducing Abduction into (Extensional) Inductive Logic Programming Systems. In *Proceedings of the 5th Congress of the Italian Association for Artificial Intelligence (AI*IA97)*, 1997.

[10] E. Lamma, P. Mello, M. Milano, and F. Riguzzi. A system for learning abductive logic programs. In M. Falaschi, editor, *Prooceedings of APPIA-GULP-PRODE 97*, 1997.

[11] L. De Raedt and M. Bruynooghe. A Theory of Clausal Discovery. In *Proceedings of IJCAI93*, 1993.

[12] L. De Raedt and L. Dehaspe. Learning from satisfiability. Technical report, Katholieke Universiteit Leuven, 1996.

[13] L. De Raedt and W. Van Lear. Inductive Constraint Logic. In *Proceedings of the 5th International Workshop on Algorithmic Learning Theory*, 1995.