An Abduction framework for Handling Incompleteness in First-Order Learning

S. Ferilli and **F. Esposito** and **N. Di Mauro** and **T.M.A. Basile** and **M. Biba**[1]

Abstract. This paper presents the ILP incremental learning system INTHELEX, focusing on its abductive capability. It is based on an abductive proof procedure that aims at attacking the problem of incomplete information by hypothesizing likely facts that are not explicitly stated in the observations. The system implements a framework in which inductive and abductive inference been brought to cooperation, and its performance in experiments on both artificial and real-world dataset is encouraging.

1 INTRODUCTION

Most traditional Machine Learning approaches focus on inductive mechanisms in order to achieve the learning goal. In order to broaden the investigation and the applicability of machine learning schemes, it is necessary to move on to more expressive representations which require more complex inference mechanisms and strategies to work together, taking advantage of the benefits that each approach can bring. In particular, one of the problems of the traditional approach to predicate-learning is the partial relevance of the available evidence, that could be takled by abduction. The problem of integrating an abductive strategy in an inductive learner is made harder in the incremental setting, where hypothesize information is more difficult since the knowledge is not completely available at the beginning.

INTHELEX (INcremental THEory Learner from EXamples) [6] is an incremental learning system for the induction of hierarchical first-order logic theories from positive and negative examples, that works under the Object Identity (OI) assumption [15]. It learns simultaneously multiple concepts, possibly related to each other, and guarantees validity of the theories on all the processed examples. It uses feedback on performance to activate the theory revision phase on a previously generated version of the theory, but learning can also start from scratch. In the learning process, it exploits a previous version of the theory (if any), a graph describing the dependence relationships among concepts, and an historical memory of all the past examples that led to the current theory. Another peculiarity of the system is the integration of multistrategy operators that may help solve the theory revision problem. The purpose of *induction* is to infer regularities and laws (from a certain number of significant observations) that may be valid for the whole population. INTHELEX incorporates two inductive refinement operators, one for generalizing hypotheses that reject positive examples, and the other for specializing hypotheses that explain negative examples.

Deduction is exploited to fill observations with information that is not explicitly stated, but is implicit in their description. Indeed, since the system is able to handle a hierarchy of concepts, some combinations of predicates might identify higher level concepts that are worth adding to the descriptions in order to raise their semantic level. For this reason, the system exploits deduction to recognize such concepts and explicitly add them to the example description. The role of *abduction* in INTHELEX is helping to manage situations where not only the set of all observations is partially known, but each observation could also be incomplete. Indeed, it can be exploited both during theory generation and during theory checking to hypothesize facts that are not explicitly present in the observations. This prevents the refinement operators from being applied, as long as possible, leaving the theory unchanged. Lastly, *abstraction* removes superfluous details from the description of both the examples and the theory. The exploitation of abstraction in the system concerns the shift from the language in which the theory is described to a higher level one according to the framework proposed in [8].

Figure 1 graphically represents the architecture of the system, embodying the cooperation between the different multistrategy operators. In the typical information flow, every incoming example preliminarily undergoes a pre-processing step of abstraction, that eliminates uninteresting details according to the available operators provided in the abstraction theory. Then, the example is checked for correct explanation according to the current theory and the background knowledge, and it is stored in the examples repository. During the coverage (i.e., checking whether the observation is explained by the current theory) and saturation (i.e., identifying higher level concepts and explicitly adding them to the example description) steps, if abduction is enabled, an abductive derivation is used. Otherwise the normal deductive derivation is started to reach the same goal without hypothesizing unseen information. In case the derivation fails, a theory refinement is necessary, and thus the example is (abductively or deductively) saturated and the inductive engine is started in order

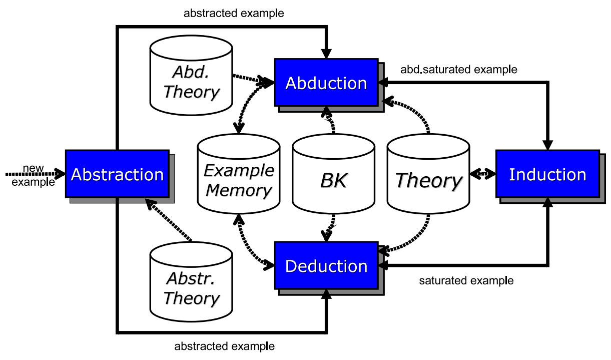

Figure 1. Architecture of the learning system

[1] Università di Bari, Italia, email: {ferilli,esposito,ndm,basile,biba}@di.uniba.it

to generalize/specialize the proper definitions, possibly using the abductive or deductive derivation whenever needed. Specifically, when a positive example is not covered, a revised theory is obtained in one of the following ways (listed by decreasing priority) such that completeness is restored: 1) replacing a clause in the theory with one of its generalizations; 2) adding a new clause to the theory; 3) adding a positive exception. When, on the other hand, a negative example is covered, a revised theory that restores consistency is reached by performing one of the following actions: 1) adding positive literals to clauses; 2) adding a negative literal to a clause; 3) adding a negative exception.

2 A FRAMEWORK FOR INTEGRATING INDUCTION AND ABDUCTION

Abduction, just like induction, has been recognized as a powerful mechanism for performing hypothetical reasoning in the presence of incomplete knowledge. Indeed, abduction is able to capture *default reasoning* as a form of reasoning which deals with incomplete information [9]. Moreover, abduction can model also *negation as failure* rule (NAF) [3], with simple transformations of logic programs into abductive theories. Thus, abduction gives a uniform way to deal with negation, incompleteness and integrity constraints [12]. The problem of Abduction, defined as *inference to the best explanation* according to a given domain theory, can be formalized as follows[4]: **Given** a theory T, some observations O and some constraints I; **Find** an explanation H such that: $T \cup H$ is consistent, $T \cup H$ satisfies I, $T \cup H \models O$. Candidate abductive explanations H should be described in terms of domain-specific predicates, referred to as *abducibles*, that are not (completely) defined in T, but contribute to the definition of other predicates. The integrity constraints I should provide indirect information about such incompleteness [9]. They can also be exploited to encode preference criteria for selecting the best explanation that may hold in this problem setting.

An abductive proof procedure can find explanations that make hypotheses (abductive assumptions) on the state of the world, possibly involving new abducible concepts. Indeed, when partial relevance is assumed, it could be the case that not only the set of all observations is partially known, but also any single observation may turn out to be incomplete. The procedure is generally goal-driven by the observations that it tries to explain. Preliminary, the top-level goal undergoes a transformation process that converts it into sub-goals. The theory and goals must be transformed into their *positive version*, by converting each literal ¬p into its positive version not_p (*default literals*). Moreover, to embed NAF in such a mechanism, it is necessary to add, for each predicate p, an integrity constraint stating that both p and its negation cannot hold at the same time. This provides a simple and unique modality for dealing with non-monotonic reasoning.

The classic algorithm for an abductive proof procedure [10] is analogous to standard SLD derivations, except that whenever a fact is not known or derivable to be true, before failing an attempt is made to check whether it can be abductively assumed to be true according to the given integrity constraints. Such a check is carried out by a consistency-check subroutine, ensuring that at least one condition of each constraint involving the hypothesized fact is (deductively or abductively) false. Each abductive assumption is considered as known in subsequent processing.

Abductive and Inductive operators address different forms of incompleteness in the theories. Specifically, abduction *extracts from the theory* a hypothesis which is considered to bear incompleteness with respect to some (abducible) predicates but is complete with respect to others. Moreover, the explanations constructed by abduction are specific to the situation of that observation. Hence abduction can be seen as a way to reason with incomplete information, rather than to complete knowledge [4].

```
Revise (T: theory; E: example; $M = M^+ \cup M^-$: historical memory);
$AbsE \leftarrow$ Abstract($E$, $AbsT$)
if Derive($AbsE$, $T$, $D$) succeeds then
    $M \leftarrow M \cup \{AbsE \cup D\}$
else
    $M \leftarrow M \cup AbsE$; $SatE \leftarrow AbsE \cup$ Saturate($AbsE, T \cup BK$)
    if $AbsE$ is a positive example then
        Generalize($T, BK, SatE, M^-$)
    else
        Specialize($T, BK, SatE, M^+$)
    end if
end if
```

```
Derive ($G$: goal; $T$: theory; $D$: abduced literals);
if Abduction is ON at the current stage of processing then
    $D \leftarrow G$
    if success $\leftarrow$ Abduct($G, T \cup BK$, $AbdT$, $D$) succeeds then
        Add to $D$ the abduced literals
    end if
else
    $D \leftarrow \emptyset$; success $\leftarrow$ Deduct($G, T \cup BK$)
end if
return success
```

Figure 2. Multistrategy Theory Revision in INTHELEX

Figure 2 summarizes the extension of the general schema of the inductive incremental learning system INTHELEX with an abductive proof procedure, derived from the classical one but properly modified to embed the Object Identity assumption. $M = M^+ \cup M^-$ represents the set of all positive and negative processed examples, E is the example currently examined, T represents the theory generated so far according to M. For simplicity, BK (the background knowledge), $AbsT$ (the abstraction theory) and $AbdT$ (the abduction theory), that must be provided by the user, are assumed to be fixed parameters (and hence are not present in the procedure headings). $AbsE$ and $SatE$ represent the example E after the abstraction and saturation phases, respectively; D is the set of literals (facts) returned by the abductive derivation when successfully applied to a goal G in theory T. Procedure *Derive* exploits abduction (through procedure *Abduct*) or deduction (through procedure *Deduct*), according to the specific settings for each step of the revision process, to prove a goal. It returns *true* or *false*, according to the success or failure of the proof procedure. *Saturate* is the procedure that returns all implicit information in the given example. *Generalize* and *Specialize* are the inductive operators used by the system to refine an incorrect theory. The resulting refinement is then implemented in the new version of the theory, and the procedure ends.

The system has been provided with an abductive proof procedure to help it in managing situations in which not only the set of all observations is partially known, but each observation could be incomplete too [6]. Specifically, abduction has been exploited to complete the observations in such a way that the corresponding examples are either covered (if positive) or ruled out (if negative) by the already generated theory, thus avoiding, whenever possible, the use of the generalization/specialization operators above mentioned to modify the theory. The set of abduced literals for each observation is minimal, which ensures that the inductive operators use abducibles only when really needed. Since specific facts are not allowed in the learned theory, the abduced information is attached directly to the observation that generated it, so that the 'completed' examples obtained this way will be available for subsequent refinements of the theory. Such information will also be available to subsequent abductions, in order for them to preserve consistency among the whole set of abduced facts. To sum up, when a new observation is available, the abductive proof procedure is started, parameterized on the current theory, the example and the current set of past abductive assumptions. If the procedure succeeds, the resulting set of assumptions, that were necessary to correctly classify the observation, is added to the example description before storing it (of course, being it minimal by definition, if no assumption is needed for the correct classification, the example description is not affected). Otherwise the usual refinement procedure (generalization or specialization) is performed.

3 EXPERIMENTS

INTHELEX's abduction capability was tested on various domains, both toy and real-world ones. In the following we show the experiments aimed at assessing the quality of the results obtained by the exploitation of the abductive version of the system in handling incomplete data. INTHELEX has been provided with the abductive proof procedure [6] in order to complete the observations in such a way that the corresponding examples are correctly classified by the already generated theory, thus avoiding, whenever possible, the use of the operators to modify the theory.

Multiplexer. The "multiplexer" problem [14] aims at learning the definition of a 6-bits multiplexer. The dataset contains descriptions of all possible configurations of 6 bits, in which the first 2 bits represent the address of one of the subsequent 4 bits, that must be set at 1. Thus, if the bit addressed is actually 1 the example is positive, otherwise it is considered as negative for the target concept. Since a 6-bits multiplexer can assume $2^6 = 64$ possible configurations, the complete training set is made up of 64 examples, 32 positive and 32 negative. The representation language of the observations is the same as in [14]. Starting from scratch with the complete training set containing all the 64 possible configurations, the correct theory was learned in 1.38 secs, performing 12 theory revisions.

Successively, an incomplete dataset was obtained by corrupting 12 examples out of 64 so that only 3 bits out of 6 of the original configuration were specified. Both the examples to be corrupted and their bits to be neglected were randomly selected for 10 times. As described in [14], such an incomplete dataset was exploited for learning theories in two different ways: first using induction only, and then using induction supported by abduction. The theories obtained in the two cases were tested (without using abduction) on the uncorrupted dataset. Table 1 shows the system performance in the two cases, averaged on the 10 corrupted datasets, as regards the number of definitions in the learned theories, the performed theory revisions, the number of exceptions, runtime and predictive accuracy. The Abduction Theory provided to the system included all the predicates as abducibles, and integrity constraints meaning that "if the bit in position N is set to 0 it can't be set to 1, and *vice versa*.
INTHELEX was able to capture the correct definitions but applying less theory revisions, adding less exceptions and in less time with respect to induction alone, while not affecting the predictive accuracy.

Table 1. System performance on the Multiplexer dataset

	Def	Rev	Exceptions	Time (sec.)	Acc
W/o Abd	4.1	6.05	2.05	4.55	99.38
With Abd	4.1	5.55	0.4	4.36	99.22

Congressional Voting Records. The problem, as reported in [11], consists in classifying a Congressman as a democrat or a republican according to his votes on 16 issues. A certain amount of noise is present in the descriptions, in the form of `unknown` votes. Definitions for the class *democrat* were learned, exploiting first pure induction and then induction plus abduction, starting from the empty theory. The corresponding predictive accuracy was tested according to a 10-fold cross validation methodology, ensuring that each fold contained the same proportion of positive and negative examples. Table 2 shows the system performance on this dataset. It is possible to note that the use of abduction improves all evaluation parameters, except Runtime. This can be explained by taking into account the additional time needed to search for consistent abductive explanations due to the large number of integrity constraints in the abductive theory.

Table 2. System performance on the Congressional Voting Records dataset

	Def	Rev	Exceptions	Time (sec.)	Acc
W/o Abd	12.40	26.90	1.7	30.30	93.33
With Abd	10.10	19.20	0.80	41.36	96.8

Family Relationships. The experiment here described aims at investigating the abductive proof procedure behavior with respect to different degrees of incompleteness. In this case, we followed the same approach adopted by [11] on the same dataset [1]. Only examples about *father* were taken into account: the training set included 36 positive examples and 200 negative ones that were randomly generated. The examples description includes also all the known facts concerning the concepts other than *father* (i.e. *son*, *daugther*, *mother*, etc.), for a total of 742 literals. Progressive corruption of such a complete description was obtained by randomly eliminating facts from it: 100% (no incompleteness, 742 literals), 90% (668 literals), 80%, 70%, 60%, 50% and 40%. For each percentage, the dataset was corrupted in 5 different ways, thus obtaining 5 corresponding learning problems whose performance was averaged according to a 5-fold cross validation methodology, ensuring that each fold contained the same proportion of positive and negative examples. Comparing the performance with and without abduction

Table 3. System Performance on the Family dataset

		Rev/Def	Runtime	Accuracy
100%	noabd	1.6	52.25	99.58
	abd	1.2	47.13	100
90%	noabd	2.2	146.19	96.28
	abd	1.2	69.04	99.17
80%	noabd	2.3	190.12	96.27
	abd	1.2	70.35	100
70%	noabd	1.8	218.03	93.78
	abd	1.2	59.70	100
60%	noabd	1.7	287.57	92.13
	abd	0.5	448.82	100
50%	noabd	1.3	256.91	92.15
	abd	0.5	43.08	100
40%	noabd	1.2	871.51	90.9
	abd	0.5	24.32	98.75

on the corrupted datasets, the benefit becomes very evident with respect to all the parameters taken into account in Table 3. Abduction is able to preserve the theories from being refined (indeed, the number of revisions per clause dramatically decreases). Moreover, lower runtimes (except in one case) prove that the abductive procedure is also efficient. Finally, note that, in spite of the number of clauses being less when using abduction in all corrupted cases, predictive accuracy is always higher than the case without abduction.

Scientific Paper Domain. In the experiment concerning the induction of classification rules for a dataset of scientific paper documents belonging to one of 4 classes [5], the corruption consisted in eliminating 8% of the descriptors for each observation (made up of 112 facts on average (76 min-170 max)) contained in the tuning set. INTHELEX was applied first without exploiting its abductive procedure. Successively, the learning process was repeated, allowing the system to exploit its abductive capability and binary constraints made up of unary and binary predicates, i.e. of the form $(ic([a(X), b(X)], ic([c(X, Y), d(X, Y)])$.

Table 4 reports the system performance as to performed theory revisions, added definitions, predictive accuracy and runtime (secs.). Predictive accuracy and number of theory revisions improve when the abductive procedure is exploited. This means that the system was able to correctly complete the corrupted observations without applying the refinement procedure. As regards runtime, it increases because of the abductive procedure.

Table 4. System performance on the Scientific Papers Domain

	Rev	Clauses	Accuracy (%)	Runtime (sec.)
Without abd	7.72	4.09	96.24	5.16
With abd	5.58	3.18	99.32	40.05

Comparison. The proposed approach does not aim at completing the training data before the learning process starts. Thus, a comparison with systems that propose to overcome the problem of handling missing values by pre-processing the training data before the learning process starts (FOIL [13], LINUS [13], ASSISTANT [2]) would be unfair. Nevertheless, we compare our system to ACL1 [11] and mFOIL [13], the FOIL extension able to deal with incomplete data on the family and congressional votes datasets (the same exploited by [11] for the same purpose). Table 5 reveals that predictive accuracy on the family dataset for progressive corruption (which percentage is reported in the first row of the table) is almost the same as that obtained by the other systems, while on congressional voting INTHELEX turned out to be better with respect to the other systems.

Table 5. Comparison of Abduction on the Family dataset

	100	90	80	70	60	50	40
INTH.	1	99.17	1	1	1	1	98.75
ACL1	1	1	99.60	1	1	97.20	97.60
mFOIL	1	99.20	98.40	97.50	98.40	98.40	95.10

4 CONCLUSION

This paper presented the ILP incremental learning system INTHELEX, with specific focus on its abductive capability that allows it to takle the problem of relevance within a language bias, that is typical of many real-world domains. After presenting and discussing, an abductive proof procedure that aims at attacking the problem by hypothesizing likely facts that are not explicitly stated in the observations, a framework in which inductive and abductive inference been brought to cooperation, and its implementation in INTHELEX, that make it able to add unseen information that can be consistently hypothesized or deduced, have been mentioned.

The abductive proof procedure exploited in this work requires that an abductive theory for the specific application domain is available. In the current practice, it is in charge of the human expert to specify it, but it is not easy to single out and formally express such parameters. Of course quality, correctness and completeness in the formalization of such meta-information can affect the feasibility of the learning process. To overcome such a bottleneck, we also developed a procedure that can automatically generate such information starting from the same observations that are input to the learning process, thus making the learning system completely autonomous [7]. Actually, the abductive theories provided to INTHELEX for the experiments in Section 3 were automatically learned using our procedure.

REFERENCES

[1] H. Blockeel and L. De Raedt, ‘Inductive database design’, in *ISMIS96*, volume 1079 of *LNAI*, pp. 376–385. SV, (1996).

[2] B. Cestnik, I. Kononenko, and I. Bratko, ‘Assistant 86: A knowledge-elicitation tool for sophisticated users.’, in *EWSL*, pp. 31–45, (1987).

[3] K.L. Clark, ‘Negation as failure’, in *Logic and Databases*, eds., H. Gallaire and J. Minker, 293–322, Plenum Press, (1978).

[4] Y. Dimopoulos and A.C. Kakas, ‘Abduction and learning’, in *Advances in Inductive Logic Programming*, ed., L. De Raedt, 144–171, IOS Press, (1996).

[5] F. Esposito, D. Malerba, and F.A. Lisi, ‘Machine learning for intelligent processing of printed documents’, *Journal of Intelligent Information Systems*, **14**(2/3), 175–198, (2000).

[6] F. Esposito, G. Semeraro, N. Fanizzi, and S. Ferilli, ‘Multistrategy Theory Revision: Induction and abduction in INTHELEX’, *Machine Learning*, **38**(1/2), 133–156, (2000).

[7] S. Ferilli, T.M.A. Basile, N. Di Mauro, and F. Esposito, ‘Automatic induction of abduction and abstraction theories from observations’, in *ILP05*, eds., S. Kramer and B. Pfahringer, volume 3625 of *LNAI*, pp. 103–120. Springer Verlag, (2005).

[8] F. Giunchiglia and T. Walsh, ‘A theory of abstraction’, *Artificial Intelligence*, **57**(2/3), 323–389, (1992).

[9] A.C. Kakas, R. Kowalski, and F. Toni, ‘Abductive logic programming’, *Journal of Logic and Computation*, **2**(6), (1993). 718-770.

[10] A.C. Kakas and P. Mancarella, ‘On the relation of truth maintenance and abduction’, in *Proceedings of the 1st Pacific Rim International Conference on Artificial Intelligence*, Nagoya, Japan, (1990).

[11] A.C. Kakas and F. Riguzzi, ‘Learning with abduction’, *New Generation Computing*, **18**(3), 243, (May 2000).

[12] E. Lamma, P. Mello, M. Milano, F. Riguzzi, F. Esposito, S. Ferilli, and G. Semeraro, ‘Cooperation of abduction and induction in logic programming.’, in *Abductive and Inductive Reasoning: Essays on their Relation and Integration*, eds., A.C. Kakas and P. Flach, Kluwer, (2000).

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

[14] Fabrizio Riguzzi, *Extensions of Logic Programming as Representation Languages for Machine Learning*, Ph.D. dissertation, University of Bologna, Novembre 1998.

[15] G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli, ‘A logic framework for the incremental inductive synthesis of datalog theories’, in *Logic Program Synthesis and Transformation*, ed., N. E. Fuchs, number 1463, pp. 300–321. Springer-Verlag, (1998).