Fundamenta Informaticae 69 (2006) 1–21
IOS Press

Multistrategy Operators for Relational Learning and Their Cooperation

F. Esposito, N. Fanizzi, S. Ferilli, T.M.A. Basile

N. Di Mauro*
Dipartimento di Informatica
Università degli Studi di Bari
Via Orabona 4, 70125 Bari, Italy
esposito,fanizzi,ferilli,basile,ndm@di.uniba.it

Abstract. Traditional Machine Learning approaches based on single inference mechanisms have reached their limits. This causes the need for a framework that integrates approaches based on abduction and abstraction capabilities in the inductive learning paradigm, in the light of Michalski's Inferential Theory of Learning (ITL). This work is intended as a survey of the most significant contributions that are present in the literature, concerning single reasoning strategies and practical ways for bringing them together and making them cooperate in order to improve the effectiveness and efficiency of the learning process. The elicited role of an abductive proof procedure is tackling the problem of incomplete relevance in the incoming examples. Moreover, the employment of abstraction operators based on (direct and inverse) resolution to reduce the complexity of the learning problem is discussed. Lastly, a case study that implements the combined framework into a real multistrategy learning system is briefly presented.

Keywords: Inductive Learning, Abstraction, Abduction, Multistrategy Learning

1. Introduction

The last decade revealed that traditional Machine Learning approaches have reached their limits [28]. Indeed, most of them exploit simple or constrained knowledge representations for the sake of efficiency

* Address for correspondence: Dipartimento di Informatica, Università degli Studi di Bari, Via Orabona 4, 70125 Bari, Italy

and are based on single (often simple or simplified) inference mechanisms. 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. One possibility is making different inference strategies work together, taking advantage of the benefits that each approach can bring.

This paper overviews and summarizes a number of studies, taken from the literature, aimed at enforcing the integration of multiple inference strategies within a logic programming framework for *relational learning*, a research area that has come evolving in the last years [30, 31, 11]. Indeed, although several learning methodologies based on different inferential mechanisms have been proposed for complex multi-relational scenarios, rarely they have come to a cooperation. Therefore, in the light of Michalski's Inferential Theory of Learning (ITL) [28], there can be a rich ground of interaction between different inferential operators, as regards both the semantic specification and their computation.

The general schema of the inductive predicate-learning paradigm ($BK \cup T \models O$) involves four variables, namely: the language $\mathcal{L}$, for which in this paper the *single representation trick* [8] will be assumed, the background knowledge BK and the theory T that contain concept definitions explaining the occurrence of some observations O. In inductive learning, observations O stand for the extensional representation of concepts, and the aim is building an intensional description T, expressed in the language $\mathcal{L}$, that explains such concepts, supposed that BK is insufficient to give such an explanation. Most approaches focus on inductive mechanisms to fine-tune T in order to achieve the learning goal. This goal has been pursued by identifying desirable properties for the inductive refinement operators, such as *ideality* [23] and *optimality* [9], and studying their mutual relationships and the conditions under which they can be obtained. For instance, a framework for the definition of ideal operators that can exhaustively search the space of candidate concept definitions and find correct ones, when the language $\mathcal{L}$ is able to express them, has been proposed in [13].

In a multistrategy perspective, such a framework can be extended by means of operators descending from other inferential strategies in order to act on the remaining variables of the general inference schema, when problems of different kinds arise. Specifically, two problems of the traditional approach to predicate-learning can be tackled: the partial relevance of the available evidence O and the insolvability of a learning problem when the language $\mathcal{L}$ is not enough powerful to express a proper predicate definition in T. The idea is to employ, respectively, abduction and abstraction to overcome such limitations. The former would pre-process the observations in order to bridge the gap between the observations and the (more operational) instances of the target concepts. The latter should guarantee the shift to a higher language bias whenever in the current one it is impossible to capture the target predicate definition.

From an operational point of view, abduction should in some way complete the observations with unknown facts that are likely to take place in the given situation and that can help in solving the learning problem at hand; it can be carried out by an abductive proof procedure, that shares the falsity-preserving nature with the inductive refinement operators [24]. As regards abstraction, it should be included for dealing with cases when learning can be more effective if it can take place at multiple different levels of complexity, which can be confronted to the language bias shift considered in [9]. A useful perspective for the integration of this inference operator in an inductive learning framework was given in [43]. In this view, concept representation deals with entities belonging to three different levels. Underlying any source of experience there is the *world*, where *concrete* objects (the 'real things') reside. It is not directly known, since any observer's access to it is mediated by his *perception* of it. The percepts reality consists in the 'physical' stimuli produced on the observer. To be available over time, these stimuli must be memorized in an organized *structure*, i.e. an *extensional* representation of the perceived world, in which

stimuli related to each other are stored together. Finally, to reason about the perceived world and communicate with other agents, a *language* is needed, that describes it *intensionally*. World, representation and language make up a *reasoning context*. In the setting dealt with in the following, situations are supposed not to change in time.

Given a reasoning context, it is possible to reason at any of the given levels. Moreover, reasoning at multiple levels can be very advantageous both in effectiveness and efficiency [22]. Then, the learning framework can be extended, for the exploitation of deductive operators based on abstraction, acting on the detail of the information at any level of the reasoning context. Indeed, it could be the case that, by simplifying the problem setting or, equivalently, by shifting the representation, a solution for the learning problem is more easily reachable.

The remainder of this paper is organized as follows. Section 2 recalls a theoretical framework for integrating different learning strategies. Section 3 summarizes notions about first-order representation languages and the problem of the relevance of the examples. Then, Section 4 discusses the integration of abductive reasoning with inductive refinement operators, and Section 5 deals with the support of operators based on abstraction to the learning process. Section 6 introduces a possible integration of the two in a learning system. Lastly, Section 7 concludes the paper.

2. Michalski's Inferential Theory of Learning

Learning is generally defined as a behavior change due to experience. A learning agent must be able to perform *inference* and must have *memory* that supplies the knowledge needed and records the results for future use. Thus, an equation

$$Learning = Inference + Memory$$

can be drawn. In humans, declarative (*conceptual*) and procedural knowledge (*skills*) seem to reside in different neural structures, and are acquired in different ways. Since there is a "conscious" access to the former kind, but not to the latter, acquiring the former is based primarily on an explicit reasoning and memorizing of the results ("studying"), whereas acquiring the latter relies primarily on practice and exercise (without much reasoning). Such a distinction does not happen in computer systems, that store and access both kinds of knowledge in the same way.

More precisely, the *Inferential Theory of Learning* (ILT) defines learning as a process affected by three fundamental factors, that make up a *learning task*: what information is provided to the learner (*input*), what the learner already knows that is relevant to the input (*background knowledge*, BK) and what the learner wants to accomplish (*goal*). The process involved in accomplishing a learning goal can be characterized in terms of high-level inference patterns, called *knowledge transmutations*, each of which takes the input and the learner's BK and generates another piece of knowledge. They represent different reasoning methods, i.e. classes of transformations that can be implemented in many different ways. Depending on the knowledge representation and the computational mechanism, knowledge transmutations are performed explicitly or implicitly (cf. symbolic systems vs. subsymbolic ones).

Any knowledge transmutation is characterized by the type of underlying inference along the truth-falsity dimension, that determines the validity of its conclusion. Since any kind of inference may be involved in a learning process, a complete learning theory must include a complete theory of inference that accounts for all possible types of knowledge transformations (see Figure 1).

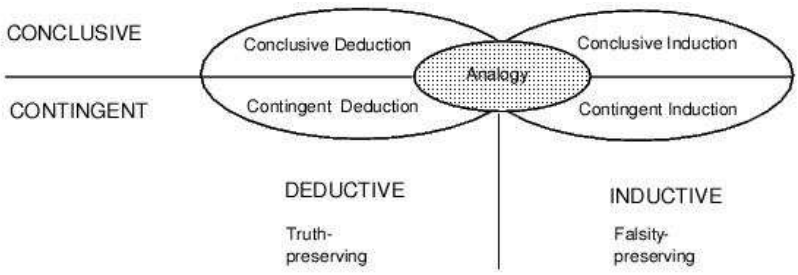

Figure 1. A classification of major types of inference

Inferences can be divided in two fundamentals types: deductive and inductive. To define these kinds of inference in a language-independent way,it is useful to start from the *fundamental equation for inference*:

$$P \cup BK \models C \tag{1}$$

where P is a set of statements (called *premise*), BK is the reasoner's *background knowledge* and C is a set of statements (called *consequent*).

Deductive inference "traces forward" Equation 1, deriving C given P and BK; inductive inference traces it "backward", hypothesizing P given C and BK. Hence, deduction is a truth-preserving inference, while induction is a falsity-preserving one (meaning that if C is false, the P cannot be true).

If Equation 1 is interpreted in an approximate, common-sense way, the "strong" (valid) entailment way may be replaced by a "weak" one, which leads to another basic distinction among types of inference. *Universal* inferences assume the "strong" entailment; *contingent* ones assume the "weak" entailment. According to such a distinction, deduction (resp., induction) is "strongly" or "weakly" truth-preserving (resp., falsity-preserving). Universal deductive inference is strictly truth-preserving, and universal induction is strictly falsity preserving. Contingent deduction (resp., induction) is truth-preserving (resp., falsity-preserving) only to the extent to which contingent dependencies involved in reasoning are true.

Both contingent deduction and contingent induction (or *abduction*) are based on domain-dependent relationships. There is no principal difference between them except if one assumes that $\models$ in Equation 1 indicates a causal ordering (i.e., P is viewed as a cause, and C as a consequence). In this case, contingent deduction yields plausible/likely consequences of given causes; abduction yields plausible/likely causes of given consequences.

The intersection of truth-preserving and falsity-preserving inference represents an equivalence-based inference. *Analogy* can be viewed as an extension of equivalence-based inference. It can be characterized as a combination of induction (involved in detecting an analogical match, i.e. the properties and/or relations similar for the two analogs) and deduction (that uses the hypothesized analogical match to derive unknown properties of the target analog).

3. Representation and Relevance

In the following, we adopt a representation language $\mathcal{L}$ expressing concepts as predicate definitions in *logic programs* made up of Datalog *clauses*. Refer to [25, 31] for the basic notions about clausal

representation. Datalog [6] is the function-free (yet not constant-free) fragment of pure Prolog. Hence, a Datalog program is a finite set of range restricted[1] Horn-clauses without functors of arity $k > 0$. It has been thought for describing deductive databases.

In such a context, typically the *unique names assumption* [33] is adopted, which has been extended by the *Object Identity* assumption [35] that is similar to the *unique substitution semantics* [21]. Such an assumption requires that "*in a clause, terms denoted with different symbols must be distinct*", i.e. they represent different entities of the domain.

Datalog^{OI}, the language resulting from the application of Object Identity to Datalog, has the same expressive power as Datalog [35]. θ-subsumption and implication under Object Identity induce ordering relationships between clauses that turned out to be weaker but more manageable than the standard ones. Hence, it is expected that effective operators from other inferential mechanisms may be defined in this framework. A thorough discussion of these generalization models can be found in [13].

In empirical learning most of the information comes from examples and observations, hence they must be expressive enough to convey the information that is necessary for the inference of general rules.

Many approaches to concept-learning need that each example is completely specified. This property is called *direct relevance* [27]. For instance, the examples could be represented as ground clauses whose body contains all the needed information accounting for the fact stated in the head:

```
bicycle(b) ←frame(f,b), saddle(s,b),
            handlebar(h,b), pedals(p,b),
            wheel(w1,b), wheel(w2,b),
            spikes(k1,w1), spikes(k1,w2),
            rim(r1,w1), rim(r2,w2),
            tire(t1,w1), tire(t2,w2)
```

Another approach, followed by most ILP systems, is *indirect relevance* [9], according to which examples are specified as ground facts, (e.g. `bicycle(b)`), and further related information is to be derived from the current theory and the background knowledge.

Direct relevance simplifies the problem of learning, by assuming that all necessary information is provided. However, it is a strong requirement, since the task of selecting the predicates to be used in the concept description has already been carried out a priori. Nevertheless, since the derivation of the rest of information is accomplished by using both the background knowledge, which is assumed correct, and the current theory, indirect relevance does not seem free of problems. Some learning systems ask directly to the user to decide about feature relevance [39], or require the user to distinguish primitive from derived features. Some other systems handle indirect relevance within a bias. In CLINT [9], the system is able to shift to a higher bias when the current one cannot describe the concept. In some approaches, knowledge about the concept-description language can be exploited to prune the search space [38]. Sometimes this knowledge is expressed in a declarative form or as irrelevance constraints.

4. Abduction: Dealing with Incomplete Observations

Abduction, just like induction, has been recognized as a powerful mechanism for performing hypothetical reasoning in the presence of incomplete knowledge. The problem of Abduction, defined as *inference to*

[1] A clause is a *range-restricted* when all variables occurring in its head also occur in the body.

the best explanation according to a given domain theory, can be formalized as follows[2] [10]:

Given a theory T, some observations O and some constraints I
Find an explanation H such that:

1. $T \cup H$ is consistent
2. $T \cup H$ satisfies I
3. $T \cup H \models O$

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. They carry all the incompleteness of theory T (if it were possible to complete these predicates then the theory would be correctly described). The integrity constraints I should provide indirect information about these abducible predicates [19]. Since several explanations may hold in this problem setting, integrity constraints can also be exploited to encode preference criteria for selecting the best ones. Recent extensions allow to properly treat more general forms of integrity constraints, viewed as active rules [26].

In general, the following schema for *Abductive Logic Programming* (ALP) [24] can be adopted:

Definition 4.1. An *abductive logic theory* is a triple $AT = (T, \mathcal{A}, \mathcal{I})$ where:

- T is a (hierarchical) normal logic program;
- $\mathcal{A}$ is the set of abducible predicates;
- $\mathcal{I}$ is a set of integrity constraints represented as program clauses.

An abductive procedure can be exploited to deal with the problem of relevance and incompleteness. Indeed, abduction is able to capture *default reasoning* as a form of reasoning which deals with incomplete information [19]. Moreover, abduction can model also *negation as failure* rule (NAF) [7], with simple transformations of logic programs into abductive theories. Thus, abduction gives a uniform way to deal with negation, incompleteness and integrity constraints [24], as shown in the following.

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 (see Section 3), 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, in order 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 (represented as $\leftarrow$ `p,not_p`). This provides a simple and unique modality for dealing with non-monotonic reasoning.

[2]Here, the theory T is assumed to include also the background knowledge.

input: T: theory, G: Datalog goal, Δ: initial abductive assumptions;
output: Δ' final abductive assumptions;
begin
$G_1 := G$; $\Delta_1 := \Delta$; $i := 1$;
while $G_i \neq \emptyset$ **do**
  $G_i = \leftarrow L_1, \ldots, L_k$;
  $L_j := R(G_i)$; (**where R is the selection rule**)
  if $L_j \notin \mathcal{A} \cup \mathcal{D}$ **then**
    $G_{i+1} := \mathcal{R}(\{G_i, C\})$ where $C \in T$ and $\{G_i, C\}$ resolve upon L_j;
    $\Delta_{i+1} := \Delta_i$
  elsif $L_j \in \Delta_i$
    $G_{i+1} := \leftarrow L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_k$
    $\Delta_{i+1} := \Delta_i$
  elsif $\overline{L_J} \notin \Delta_i$ and $\exists \Delta_C = consistency(T, \{L_j\}, \Delta_i \cup \{L_j\})$
    $G_{i+1} := \leftarrow L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_k$
    $\Delta_{i+1} := \Delta_C$
  $i := i + 1$;
return Δ_i
end.

Figure 2. Abductive Refutation Algorithm

Figure 2 sketches the classic algorithm for an abductive proof procedure proposed by Kakas and Mancarella [20]. Initially, like in standard SLD derivations, a literal is selected. When it is not abducible or a default one, the procedure continues with a resolution with clauses from T, through the resolution operator $\mathcal{R}$. Else, if the fact has been already assumed abductively (and consistently) as true in previous steps it can be dropped (a case of successful proof). As a further possibility, a new fact may be assumed as true, provided that it is consistent with the current integrity constraints $\mathcal{I}$, which is verified by the consistency-check subroutine (reported in Figure 3). In this subroutine, whenever the literal to be tested is an abducible or default one, but neither it nor its complement have been already abduced (on last if-branch), the abductive procedure is called, in order to check whether it can be hypothesized by abduction. The other branches are similar to derivations except that, when dealing with an abducible or default literal, if it has already been abduced then it is simply dropped (i.e. consistency is trivially proved); otherwise, if its complement has already been abduced or can be abduced, the entire goal is dropped. Thus, the two procedures may call each other both when a new abductive assumption requires further consistency checks against the constraints and vice-versa.

As a mechanism for *knowledge assimilation*, abduction can be employed when observations about the world are given (which does not necessarily represent an explicit learning problem), and they are to be assimilated into a knowledge base [10].

From an inferential point of view abduction and induction are similar since both are falsity-preserving [28]. Hence, their conclusions can be controversial. Accordingly, they are both non-monotonic and deal with forms of incompleteness of the available information. Abduction is generally understood as reasoning from effects to causes (or explanations), while induction concerns the inference of general rules from specific data. The two problems can be regarded as dual to each other since they share the

input: T: theory, $L \in \mathcal{A} \cup \mathcal{D}$: literal, Δ: initial abductive assumptions;
output: Δ' final abductive assumptions;
begin
$C_1 := \{G = \leftarrow L_1, \ldots, L_p |\ G \in \mathcal{R}(\{\{L\}, C\}),\ C \in \mathcal{I},\ p > 0\}$;
$\Delta_1 := \Delta$; $i := 1$;
while $C_i \neq \emptyset$ **do**
  $C_i = C_i' \cup \{\leftarrow M_1, \ldots, M_k\} = C_i' \cup B_i$;
  Consider some $M_j \in B_i,\ j = 1, \ldots, k$;
  if $M_j \notin \mathcal{A} \cup \mathcal{D}$ **then**
    $C_{i+1} := C_i' \cup \mathcal{R}(\{B_i, D\})$, resolved upon M_j, such that $\square \notin C_i'$ and $D \in T$;
    $\Delta_{i+1} := \Delta_i$
  else if $M_j \in \mathcal{A} \cup \mathcal{D}$ and $M_j \in \Delta_i$, $k > 1$ **then**
    $C_{i+1} := C_i' \cup \{\leftarrow M_1, \ldots, M_{j-1}, M_{j+1}, \ldots, M_k\}$;
    $\Delta_{i+1} := \Delta_i$
  else if $M_j \in \mathcal{A} \cup \mathcal{D}$ and $\overline{M}_j \in \Delta_i$ **then**
    $C_{i+1} := C_i'$; $\Delta_{i+1} := \Delta_i$
  else if $M_j \in \mathcal{A} \cup \mathcal{D}$ and ($M_j \notin \Delta_i,\ \overline{M}_j \notin \Delta_i$) **then**
    if $\exists \Delta_A = abduce(T, \leftarrow \overline{M}_j, \Delta_i)$ **then**
      $C_{i+1} := C_i'$; $\Delta_{i+1} := \Delta_A$
  $i := i + 1$
return Δ_i
end.

Figure 3. Consistency Derivation Algorithm

basic formal specification. In the former paradigm, an initial theory is needed containing the conditions that can be involved in the construction of the explanation. These can be made explicit by means of abductive inferences, thus knowledge cannot be induced without a prior abductive explanation. In the latter, the theory is to be synthesized previously from examples by means of inductive mechanisms, hence examples cannot be explained without a prior inductive inference.

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 [10].

Induction builds hypotheses in terms of the concepts possibly present in the theory. On the other hand, the observations concern typically concepts that are not yet defined in the theory, thus, as an effect of abduction, the theory is completed with the definitions learned for these concepts. This suggests that abduction can be employed at an initial phase of the learning task to explain the training data (raw observations about the world), thus providing relevant examples, described at a higher, theory-dependent level of knowledge, for deriving newer hypotheses with standard inductive operators.

The general schema of an inductive learning algorithm can be extended with an abductive proof procedure. The integration of abduction in an inductive learning framework makes this process more knowledge intensive, for the exploitation of the whole information available both in the theory and in the background knowledge.

In the original ALP framework, the theories are to be interpreted according to the Stable Model semantics. Unfortunately, under such semantics it does not hold that

$$T \models P_1,\ T \models P_2,\ \ldots,\ T \models P_n \text{ implies } T \models P_1 \wedge P_2 \wedge \cdots \wedge P_n$$

(think of T as a theory and of the P_i's as positive examples). Thus, the completeness of the theory with respect to the positive examples available cannot be checked by testing each of them separately. The solution proposed in [24] is to test them in a pipeline so that at each step the set of abducibles assumed in the previous tests is taken into account. Another solution could be to adopt the declarative semantics, where the above relation holds, and hence it is possible to test the positive examples separately. Indeed, representing the theories as (hierarchical) normal programs allows to maintain the same semantics (least Herbrand model), coping with negation by means of NAF. From a model-theoretic point of view, there are no problems in adopting this rule if only hierarchical theories are adopted. Indeed, the language of definite clauses with integrity constraints has been proven to subsume NAF [12], thus integrity constraints can be simulated using NAF. Hence, the declarative semantics of hierarchical abductive theories comprises CET and the completion of clauses in the theory [7].

5. Abstraction: Dealing with an Incomplete Representation

Reasoning by abstraction is related to the problem of relevance. Abstraction is a transmutation that reduces the amount of information conveyed by the description of a given *reference set* [28] (i.e. a set of concept instances). Typically, this reduction is lead to such an extent that the information that is relevant to the achievement of a learning goal is maintained, whereas the rest of information is discarded or hidden.

Abstraction can be regarded as a form of deductive transmutation since it preserves the important information about the input and does not hypothesize anything (provided that the background knowledge is complete).

In this case the gap to be abridged concerns different intensional language representations $\mathcal{L}_i$ employed to account for the extensional knowledge of the world representation. When the current language bias $\mathcal{L}_i$ proves not to be expressive enough for representing concept descriptions that can explain the examples, it could be the case to shift to a higher one [41, 9]. Abstraction can provide operators for performing this shift. If learning is modeled as a search process in a space of versions, typically by means of an upper and a lower bounding set of versions [29], the necessity of shifting to a different bias can be detected when the boundaries meet [9].

So far, abstraction has been investigated in the context of theorem proving and problem solving [32, 18]. In this case, we address a different facet, namely the role that abstraction can play in inductive learning. In particular, some studies have already focused on learning concepts in propositional logics or equivalent non structural representations, especially for learning plans for autonomous agents [42] at different task layers [37], and on showing how biologically-inspired Perceptual Learning mechanisms could be used to build efficient lowlevel Abstraction operators that deal with real world data [4]. However, the focus in the present work is on integrating abstraction in a first order concept learning framework as in [17, 3, 2].

Recent developments in conceptual clustering tend to consider the case of classifications that organize *knowledge* [3, 5]. These purely symbolic approaches are not based on distance optimization but on

generalization languages and can take into account the available domain knowledge. Suitable languages for describing structural knowledge can be represented by means of concept triples (like in RDF). The problem is then how to deal with the NP-completeness of graph matching. A possible solution lies in progressively enriching the languages used to organize the same extension of objects. Groups of triplets covering the same extension (*reference* in Michalski's terminology) can be formed as candidate generalizations (most specific ones are preferred). Then nodes can be connected on the ground of inclusion relationships between their extensions. More complex relations represented in a concept graph are obtained by abstracting paths between concepts giving rise to new abstract relations between them [3].

Abstraction is defined as a mapping between representations that are related to the same reference set but contain less detail. Apparently, abstraction and inductive generalization are very similar notions to each other. However, they belong to totally different classes of inference. Indeed, the former is deductive and the latter inductive. Typically, abstraction is a truth-preserving operation, since deduction is the supporting inference. The set of strong inferences that are derivable from the starting description is larger than the one supported by the output one (*strong* abstraction). Nevertheless, it is possible to weaken this setting, by allowing forms of abstraction that are not fully truth-preserving (*weak* abstraction) [28]. Hence, abstraction can be embedded in an inductive generalization framework; for instance, ignoring details about the objects belonging to a class may facilitate the generation of rules for that class.

5.1. Syntactic Abstraction

The process of functional transformation of a representation into another one allows to meet two objectives: helping to solve the learning problem in the starting search space and making the search for a solution more easily manageable [18].

Definition 5.1. Given two clausal theories T and T' built upon different languages $\mathcal{L}$ and $\mathcal{L}'$ (and derivation rules), an *abstraction* is a tuple (T, T', f), where f is a computable total mapping between clauses in $\mathcal{L}$ and those in $\mathcal{L}'$. We will designate T as *ground theory* and T' as *abstract theory*.

The goal of such an abstraction mapping is to preserve some properties and discard others. Among the properties that should be preserved, the most important from the theorem proving viewpoint is *derivability*. A taxonomy of abstractions with respect to such a property can be drawn [18], characterizing the statements of the theory in terms of theoremhood.

Definition 5.2. An abstraction (T, T', f) is said to be:

- *constant* (TC) iff $\forall C \in \mathcal{L} : \; T \vdash C \Leftrightarrow T' \vdash f(C)$;
- *decreasing* (TD) iff $\forall C \in \mathcal{L} : \; T' \vdash f(C) \Rightarrow T \vdash C$;
- *increasing* (TI) iff $\forall C \in \mathcal{L} : \; T \vdash C \Rightarrow T' \vdash f(C)$.

Constant abstractions map theorems to theorems and vice-versa: all the solutions to a derivation problem in the ground theory have a corresponding solution in the abstract theory. In decreasing abstractions, some solutions may be lost because of the detail elimination, but any solution in the abstract theory can be traced back to the ground theory. Conversely, for increasing abstractions, the abstract problem might have more solutions than the ground one and hence a solution cannot always be mapped back to the ground language (*false proof problem* [40]).

Table 1. Uses of abstraction classes

	deductive	**abductive**
positive	TD	TI
negative	TI	TD

Abstractions can be employed in different ways (see Table 1). A possible classification distinguishes between *deductive* and *abductive* uses of these mappings. Deductively, a property for $f(C)$ holding in the abstract theory assures that the same property holds for C (e.g. derivability). Using abstraction abductively gives a suggestion, not a guarantee, on the truth of a property in the ground theory. Another dimension distinguishes between *positive* and *negative* uses. With positive uses, given that $f(C)$ is derivable, then also C is (or may be) provable. In the negative use, if $f(C)$ is unprovable, than C is (may be) unprovable.

A deductive use yields *sound* theorem proving strategies; an abductive one yields *complete* strategies (since it allows to always find proofs). In the inductive learning framework, one should focus primarily on the abductive use of abstraction. Even increasing abstractions can be used for abducting, positively, that something can be true in the ground theory, given the proof for some abstract elements ($T' \vdash f(C) \Rightarrow T \vdash C$). Additionally, one can exploit abstractions abductively to suggest negative information that could be true ($T' \nvdash f(C) \Rightarrow T \nvdash C$).

5.2. Semantic Abstraction

So far, abstraction has been regarded as a mapping between formulæ. Since a mere rewriting cannot guarantee for the preservation of consistency [40], a semantic formulation of abstraction becomes necessary. Semantic abstractions are definable in first order logic because of the soundness of resolution. This defines a TI abstraction once a universe of objects is given for the variables and calculation procedures for functions and predicates (and the standard interpretation for connectives and quantifiers).

An original approach has been formalized in [17]. It preserves many important properties such as completeness, consistency, extensionality and the hierarchy of the representations. The central point is the construction of abstract data types starting from the objects of the ground theory, extending Tenenberg's idea of restricted abstract mapping, by allowing abstractions as semantic mappings between models. Some of these definitions and results are briefly presented in the following, adapted to a first-order clausal representation.

Definition 5.3. An *abstraction theory* AT from $\mathcal{L}$ to $\mathcal{L}'$ is a consistent set of formulæ:

$$c(X) \leftrightarrow (d_1(Y_1) \wedge X = f_1(Y_1)) \vee \cdots \vee (d_n(Y_n) \wedge X = f_n(Y_n))$$

where c is a predicate in $\mathcal{L}'$, $d_i,\ i = 1, \ldots, n$ are predicates of $\mathcal{L}$ and the functions $f_i,\ i = 1, \ldots, n$ map the arguments of the d_is onto that of c.

i.e., it is a collection of intermediate concepts represented as a disjunction of alternative definitions. Each concept can be equivalently expressed as a set of clauses with the same head $c(X)$, implicitly assuming the double entailment yielded by the completion semantics [7]. This allows for an easier integration with the abductive operator defined in the previous section.

Definition 5.4. Given two languages $\mathcal{L}$ and $\mathcal{L}'$, a clausal theory $T \subseteq 2^{\mathcal{L}}$ and an abstraction theory AT that maps predicates of $\mathcal{L}$ to predicates of $\mathcal{L}'$, we designate *abstraction* of T, the theory $T' \subseteq 2^{\mathcal{L}'}$, such that: $T \cup AT \vdash T'$.

Any least Herbrand model $\mathcal{M}'$ of T' can be considered as the abstraction of a model $\mathcal{M}$ of T, hence the consistency is preserved. Also the generality relationship among clauses is preserved, and various other properties fulfilled. In particular, preservation of the entailment corresponds to preservation of correctness of the descriptions. Indeed, an abstract concept is defined as entailed by a series of characteristics (descriptions). Thus, if a concept has been defined from a series of examples, and the ground predicates cover these examples, the same does the abstract concept, since the entailment is preserved (correctness). In fact, the abstract concept may cover more examples, making the abstraction theory a generalizing operator.

A notion of completeness of the abstraction theory can be given as follows.

Definition 5.5. An abstraction theory AT is *completeness preserving* iff it holds:
$T' \cup \{C'\} \vdash D'$ implies $T \cup \{C\} \vdash D$ for any $C, D \in T$ and $C', D' \in T'$ such that: $T \cup AT \cup \{C\} \vdash C'$ and $T \cup AT \cup \{D\} \vdash D'$.

Thus, a completeness preserving abstraction theory covers exactly those examples that were covered by the ground one. In the following, the focus is set on the problem of building abstraction theories through proper operators.

5.3. Abstraction Operators

Among the three levels of a reasoning context (see Section 1) there may be a complex interplay. If they are generated upwards, the world-perception $P(W)$ is the very source of information, that is recorded into the structure S and then described by the language L. Modifications to the structure and language are merely a consequence of differences in the perception of the world, that may occur for a number of reasons (e.g., the medium used and the focus-of-attention), even though the world itself does not change[3]. Thus, abstraction should be defined as a mapping at the level of perceived world [43].

Abstraction takes place at level $P(W)$ by means of a set of operators Ω and then propagates to higher levels, where it is possible to identify operators corresponding to the previous ones. At level S we have Σ, whereas at level L we have Λ. These sets can be reduced or augmented according to domain-specific abstraction types, but in general contain operators for performing the following operations: grouping indistinguishable objects into equivalence classes; grouping a set of ground objects to form a new compound object[4] that replaces them in the abstract world; ignoring terms that can be in the abstract world, where they disappear; merging a subset of values that are considered indistinguishable; dropping a subset of arguments, thus reducing the arity or a relation; eliminating *all* arguments, so that the function (or relation) moves from a predicate logic to a propositional logic setting (which corresponds to a *propositional abstraction* at the language level).

Modifications are performed by mappings among perceived world, and cause corresponding modifications to the structure and language only as side-effects. Hence, given a ground reasoning context,

[3] In ML, this corresponds to the phase of *feature selection*.

[4] It is called term construction [17], and offers the most significant promises for limiting the complexity of learning in a first order logic setting, since it simplifies the matching process between hypotheses and examples.

knowing the operators in Ω it is possible not only to obtain the perceived world abstraction P_a, but also the abstract structure S_a and language L_a by exploiting the correspondent operators in Σ and Λ, respectively.

When concepts and examples are defined at ground level, the problem is to determine the features of an abstraction mapping that make the corresponding abstract concepts more *easily* learnable in the abstract space using L_a. Besides, it should be specified how to map back an abstract concept definition learnt in L_a to one in the ground language L_g.

The definition of an unknown concept is searched for in a space, defined by the underlying language, whose most important structure is the generality relationship among hypotheses[5]. Thus, in our case it is desirable that abstractions preserve such a relationship (*Generality-preserving Language Abstraction Mapping*, or *GLAM*). The only restriction is that the abstract structure must be used for evaluating the truth of the abstracted hypotheses.

As regards the second problem, it should be noted that while syntactic TD abstractions are deprecated for loosing theorems, when shifting to the ground space from the abstract one, this may no longer be a limitation in our case, as long as the generality relation is preserved. Indeed, in the presented framework, some abstractions defined at the domain level may not be expressible at the language one.

5.4. A Framework for Abstraction Operators in ILP

In inductive learning, the shift to a higher level representation (*shift of language bias* [41]) can be performed directly when the abstraction theory is given. Thus it can be exploited to shift in a hierarchy of search spaces. More interestingly, when the abstraction theory is not given, it has to be learned during the process of learning. Unfortunately, it has been proven that the shift is not always decidable [36].

Inverse resolution operators [30], by tracking back resolution steps, can suggest new salient properties and relations of the learning domain. They are divided in V-operators and W-operators depending on the form of the resolution step(s) inverted. In particular the W-operators may introduce new predicates. Inverse resolution operators can be a valuable mechanism to build abstraction theories, as introduced in [16]. To this purpose, the absorption, inter-construction and intra-construction operators can be exploited, also in the case of first order clauses. This paper is interested in the case of a Datalog program as ground space of the abstraction, as in [34], where clauses are *flattened*, hence function-free.

Definition 5.6. (Inversion Resolution Operators)

absorption: let C and D be two Datalog clauses. If there exists a unifier θ such that $\exists S \subset body(C), S = body(D)\theta$, then applying the absorption operator yields the new clause C' such that:

- $head(C') = head(C)$
- $body(C') = (body(C) \setminus S) \cup \{head(D)\theta\}$.

inter-construction: let $\{C_i | i = 1, \ldots, n\}$ be a set of Datalog clauses. If there exists a set of literals R and a unifier θ_i for each clause C_i, such that $\exists S_i \subset body(C_i),\ S_i = R\theta_i$, then we define:

[5] Given two hypotheses h_1 and h_2, h_1 is more general than h_2 if the set of models of h_2 is included in that of h_1.

- a new predicate $L \leftarrow R$
- for all $i = 1, \ldots, n$ $body(C_i)$ can be rewritten as $(body(C_i) \setminus S_i) \cup \{L\theta_i\}$.

intra-construction: let $\{C_i | i = 1, \ldots, n\}$ be a set of Datalog clauses. If there is a subset $S \subset body(C_i)$ for all i, then we define:

- a new predicate with the clauses: $L \leftarrow R_i$ $(i = 1, \ldots, n)$ where $R_i = body(C_i) \setminus S$
- the predicate in $head(C_i)$ can be defined by a single clause: $head(C_1) \leftarrow S, L$.

Absorption may generalize the clause it is applied to. Since normally absorption should not generalize the clauses, it is important to eliminate the direct or indirect generalizations that absorption might cause. This could be solved by using an "oracle" or the CWA and completion semantics. However, a better solution is to limit the absorption operator as proposed by Giordana and Saitta, by introducing the *Non-Generalizing absorption* operator (*NG-Absorption*) [16]. Like for absorption, also the limitation of NG-absorption makes it not applicable to recursive clauses. However, it is possible to see that abstracting a theory T by using NG-absorption and the constructive operators (inter and intra-construction), the resulting abstract theory T' is consistent with the models of T (even non minimal ones). These abstraction operators realize the so-called *Non-Generalizing CP-abstraction*. In the actual integration of abstraction in an inductive learning system, a more conservative operator, named *saturation* [34], can be used.

A different way to cope with generalizing abstraction operators is to employ an abstraction operator which has the effect of specializing the theory. The underlying idea is that, similarly to the previous operators, based on the inversion of resolution, one could use an operator that partially performs the resolution step. In fact this operator, known as *unfolding* [1], has been studied as a mechanism for program transformation, aiming at the improvement of logic programs in terms of efficiency and comprehensibility, which is a typical case of *theory restructuring*. Unfolding can be applied directly to abstraction theories. Indeed, a natural way to specialize an overly general abstraction theory AT is specializing a clause $C \in AT$ by substituting it with its resolvents upon some atoms in the body. It is straightforward to notice that the new abstraction theory AT' is implied by AT. Some of these resolvents can be deleted from AT', thus specializing the theory and ruling out some unwanted possible generalizations.

6. A Case Study for a Combined Framework

Two problems of the traditional approach to predicate-learning are the partial relevance of the available evidence and the infeasibility of a learning problem when the language is not enough powerful to express a proper predicate definition. Integration of different inference strategies can help to carry out a broader approach to inductive *predicate-learning* [9]. Indeed, there can be a rich ground of interaction among them, both as regards the semantic specification and their computation. In the following we analyze a system that is endowed with multistrategy capabilities, according to the Inferential Theory of Learning framework presented in Section 2, in order to improve effectiveness and efficiency of the learning task. Such a system, named INTHELEX, originally developed as a purely inductive learner, was first provided with abductive capabilities in [15], and successively extended with abstraction and deduction operators, showing interesting performance when applied to complex real-world problems [14]. It is an incremental learning system for the induction of hierarchical theories from positive and negative examples, using

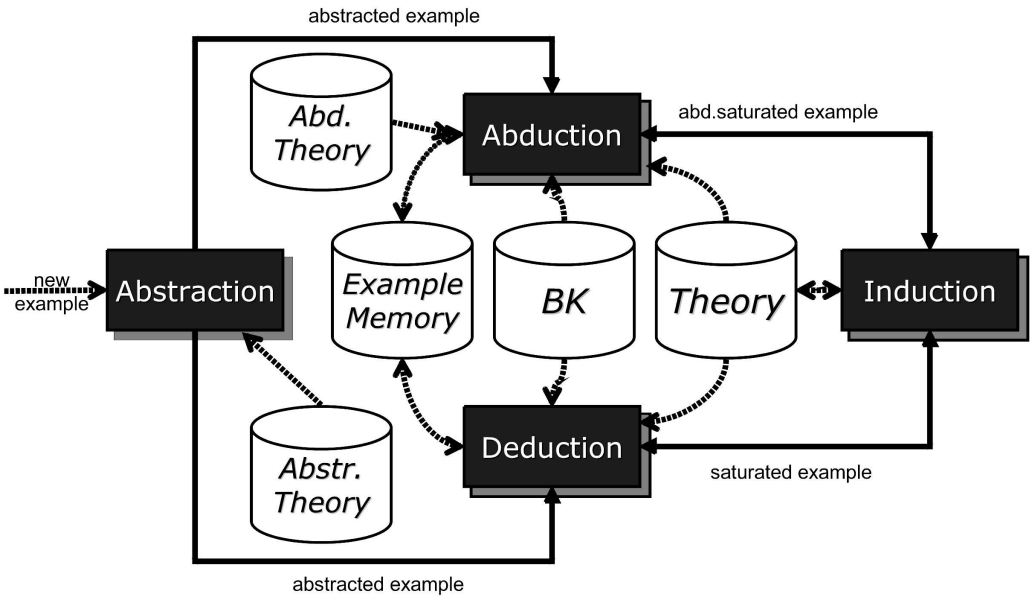

Figure 4. New INTHELEX architecture

Datalog^{OI} (see Section 3) as a representation language. Specifically, theories (and background knowledge) are represented as sets of constant-free clauses whose body describes the definition of the concept in the head. As to the relevant evidence, it will be represented by ground clauses whose body describes the observations accounting for the example in the head. Examples regard directly the concepts to be learned, yet no complete relevance is assumed. Observations concern concepts that are "perceived" from the world.

The new version results in a novel integration scheme for coordinating and harmonizing different inference strategies aimed at a profitable cooperation. Namely, *deduction* exploits the provided Background Knowledge (i.e. some partial concept definitions known to be correct, and hence not modifiable) to recognize known objects in an example description and explicitly add them to the description itself. *Abstraction* can be cast as the process of focusing on what is relevant in an observation. Indeed, ignoring the details about the objects belonging to a class may facilitate the generation of rules for that class. *Abduction* is used to complete the observations, whenever possible, in such a way that the examples they represent are explained (if positive) or rejected (if negative). This prevents the refinement operators from being applied, as long as possible, leaving the theory unchanged. Implementing the theoretical combined framework obviously raises new issues due both to the representation formalism used by the system and to the procedures according to which it manipulates the available data. Both abduction and abstraction are exploited to perform a pre-processing of incoming information. Even if using opposite approaches, both aim at reducing the computational effort required to learn a correct theory with respect to the incoming examples.

Figure 4 graphically represents the architecture of the new version of INTHELEX, embodying the cooperation between the different multistrategy operators. The presence throughout the diagram of dotted arrows, representing (possibly abductive) derivations, demostrates how abduction is pervasive in the proposed behaviour. Specifically, abduction can be inhibited by the user in any of these derivations, trans-

forming them into normal derivations, in order to make the system more cautios in relying on facts whose truth value is not guaranteed to be correct. Let us now describe the typical information flow inside the architecture. 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 (possibly hypothesizing unknown facts, if necessary). 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. In the coverage and saturation steps, an abductive derivation that exploits the abductive theory (containing the abducibles and the integrity constraints) is used if abduction is turned on, 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 to generalize/specialize the proper definitions, possibly using the abductive or deductive derivation whenever needed. The resulting refinement is then implemented in the new version of the theory, and the procedure ends.

Figure 5 summarizes the new behavior of the system when it is applied to each new example. $M = M^+ \cup M^-$ represents the set of all positive (M^+) and negative (M^-) 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 are to be provided by the user, are assumed to be fixed parameters (and hence are not present in the procedure headings). $AbsE$ and $SatE$ are, respectively, the examples generated by the abstraction and saturation phases from the example; D is the set of literals 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 process of theory revision, as performed by the system, is now briefly summarized. 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: replacing a clause in the theory with one of its generalizations; adding a new clause to the theory; 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: adding positive literals to clauses; adding a negative literal to a clause; adding a negative exception.

Deduction is performed by a saturation operator that exploits a *dependency graph* describing the dependencies among the concepts to be learned. Whenever a new example is taken into account, and before it is stored in the historical memory, it undergoes a saturation phase. If any of its (direct or indirect) sub-concepts in the dependency graph can be recognized in its description according to the definitions learned thus far and the Background Knowledge, literals concerning those concepts are added (properly instantiated) to the example description. The background knowledge rules cannot be modified by the refinement operators.

According to the framework described in Section 5, abstraction consists in the shift from a description language to a higher level one, and an abstraction theory is used to perform such a shift. The abstraction theory, if any, must be given by the Expert (i.e. it has not to be learned by the system), and the system will automatically apply it to each example before processing it (thus, the shift to the higher level language always occurs, and it occurs just once for each example). Indeed, incremental systems cannot change

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($T, BK, SatE, M^+$)

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
else
  $D \leftarrow \emptyset$
  success $\leftarrow$ Deduct(G, $T \cup BK$)
return *success*

Figure 5. Multistrategy Theory Revision in INTHELEX

the abstraction theory in the middle of a learning task, because the result of abstracting separately the partial theory already learnt and the past examples would not be the same as learning a theory directly from examples described at the new abstraction level. As already pointed out, abstraction at the language level is only a consequence of an abstraction step occurred at the perception level and memorized at the structure level. This is reflected in the system by the fact that examples and related observations are stored in the historical memory, and used for undergoing the normal processing, in the new (abstract) form, and not as they are provided to it, which corresponds to application of operators in the set Σ. Conversely, the abstraction theory contains information for performing the shift specified by the operators of the set Λ. In detail, the abstraction operator of Absorption, proposed in the theoretical framework in Section 5.4, is applied to the example description exploiting the abstraction theory. Unlike deduction, there is no reference to the dependency graph in the application of abstraction. Since the structure level must be abstracted, and the learned theory must be abstracted too, the abstraction step must precede the storage of the incoming examples (and hence the possible inductive refinement). In general, application of these operators yields a TD abstraction, in that they eliminate information that in principle could be useful for proving other facts. Assuming that the provider of the abstraction theory knows that no detail that it eliminates is necessary for any proof that the system needs to carry out, all the 'important' things that the system could prove are preserved also after the language shift, and hence it can be considered a TC abstraction.

According to the framework in Section 4, 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 [15]. 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 operators 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. The negative literals that might appear in the set of abduced literals are not added to the description, but are handled by means of the CWA. Since observations are described exclusively in terms of basic predicates in the description language, only such predicates have been considered as abducibles. These predicates do not have a definition in the theory, which also complies with the requirements for abducibles. No explicit integrity constraints for default negation is needed: they are implicitly assumed, and default negation is simulated by means of NAF (thus embedding it in the normal Prolog derivation). 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.

7. Conclusions

This paper presented a survey of relevant research in the literature aimed at studying the possibile integration of different inference strategies, in order to carry out a broader approach to *inductive learning* than the classical single-strategy framework. Indeed, there can be a rich ground of interaction between different operators, both as regards the semantic specification and their computation.

In particular, two important problems of inductive learning have been considered, namely the problem of relevance within a language bias and the shift of language bias. Initially, an abductive proof procedure has been discussed that aims at attacking the former problem by hypothesizing likely facts that are not explicitly stated in the observations. Successively, a way for extending the framework with the integration of deductive operators based on abstraction has been presented, allowing to switch to more suitable description languages when the adopted one proves unable to express the target concept(s) to be learnt.

Finally, a framework in which these methodologies have been brought to cooperation has been mentioned. It is implemented in an inductive learning system, by extending it with features that make it able to handle observation descriptions both by eliminating details that are not significant to the learning process, and by adding unseen information that can be consistently hypothesized or deduced.

References

[1] Boström, H., Idestam-Almquist, P.: Specialization of Logic Programs by Pruning SLD-trees, *Proceedings of the 4th International Workshop on Inductive Logic Programming* (S. Wrobel, Ed.), GMD-Studien, vol. 237 of *GMD-Studien*, Bad Honnef/Bonn, 1994, 31–48.

[2] Bournaud, I., Courtine, M., J.-D., Z.: Propositionalization for Clustering Symbolic Relational Descriptions, *Proceedings of the Twelfth International Conference on Inductive Logic Programming (ILP 2002)*, Sydney Australia, July 2002.

[3] Bournaud, I., Courtine, M., Zucker, J.-D.: Abstractions for Knowledge Organization of Relational Descriptions, *Proceedings of Abstraction, Reformulation, and Approximation, 4th International Symposium, SARA 2000* (B. Y. Choueiry, T. Walsh, Eds.), Lecture Notes in Computer Science, vol. 1864 of *Lecture Notes in Computer Science*, Springer, 2000, 87–106.

[4] Bredeche, N., Zucker, J.-D., Zhongzhi, Z.: Perceptual Learning and Abstraction in Machine Learning, *Proceedings of the 2nd IEEE International Conference on Cognitive Informatics (ICCI'03)*, 2003, 18–25.

[5] Carpineto, C., Romano, G.: Galois: an order-theoric approach to conceptual clustering, *Proceedings of the International Conference on Machine Learning*, Ahmerst, 1993, 33–40.

[6] Ceri, S., Gottlob, G., Tanca, L.: *Logic Programming and Databases*, Springer, 1990.

[7] Clark, K.: Negation as Failure, in: *Logic and Databases* (H. Gallaire, J. Minker, Eds.), Plenum Press, 1978, 293–322.

[8] Cohen, P., Feigenbaum, E., Eds.: *The Handbook of Artificial Intelligence*, vol. 3, Morgan Kaufmann, 1981.

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

[10] Dimopoulos, Y., Kakas, A.: Abduction and Learning, in: *Advances in Inductive Logic Programming* (L. D. Raedt, Ed.), IOS Press, 1996, 144–171.

[11] Džeroski, S., Lavrač, N., Eds.: *Relational Data Mining*, Springer, Berlin, 2001.

[12] Eshghi, K., Kowalski, R.: Abduction Compared to Negation by Failure, *Proceedings of the 6th International Conference on Logic Programming* (G. Levi, M. Martelli, Eds.), The MIT Press, 1989, 234–255.

[13] Esposito, F., Fanizzi, N., Ferilli, S., Semeraro, G.: A generalization model based on OI-implication for ideal theory refinement, *Fundamenta Informaticae*, **47**, 2001, 15–33.

[14] Esposito, F., Ferilli, S., Fanizzi, N., Basile, T., Di Mauro, N.: Incremental Multistrategy Learning for Document Processing, *Applied Artificial Intelligence*, **17**(8/9), 2003, 859–883.

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

[16] Giordana, A., Roverso, D., Saitta, L.: Abstracting Concepts with Inverse Resolution, *Proceedings of the 8th International Workshop on Machine Learning*, Morgan Kaufmann, Evanston, IL, 1991, 142–146.

[17] Giordana, A., Saitta, L.: Abstraction: A General Framework for Learning, *Working Notes of the Workshop on Automated Generation of Approximations and Abstractions*, Boston, MA, 1990, 245–256.

[18] Giunchiglia, F., Walsh, T.: A Theory of Abstraction, *Artificial Intelligence*, **57**(2/3), 1992, 323–389.

[19] Kakas, A., Kowalski, R., Toni, F.: Abductive Logic Programming, *Journal of Logic and Computation*, **2**(6), 1993, 718–770.

[20] Kakas, A., Mancarella, P.: On the Relation between Truth Maintenance and Abduction, *Proceedings of the 1st Pacific Rim International Conference on Artificial Intelligence*, Nagoya, Japan, 1990, 438–443.

[21] Khardon, R.: Learning function-free Horn expressions, *Machine Learning*, **37**(3), December 1999, 241–275.

[22] Kramer, S., Lavrač, N., Flach, P.: Propositionalization Approaches to Relational Data Mining, in: *Relational Data Mining* (S. Džeroski, N. Lavrač, Eds.), Springer, September 2001, 262–291.

[23] van der Laag, P.: *An Analysis of Refinement Operators in Inductive Logic Programming*, Ph.D. Thesis, Erasmus University, Rotterdam, The Netherlands, 1995.

[24] Lamma, E., Mello, P., Milano, M., Riguzzi, F., Esposito, F., Ferilli, S., Semeraro, G.: Cooperation of Abduction and Induction in Logic Programming., in: *Abductive and Inductive Reasoning: Essays on their Relation and Integration* (A. Kakas, P. Flach, Eds.), Kluwer, 2000, 233–252.

[25] Lloyd, J.: *Foundations of Logic Programming*, 2nd edition, Springer, 1987.

[26] Mancarella, P., Terreni, G.: An Abductive Proof Procedure Handling Active Rules, *AI*IA 2003: Advances in Artificial Intelligence, 8th Congress of the Italian Association for Artificial Intelligence, Pisa, Italy, September 23-26, 2003, Proceedings* (A. Cappelli, F. Turini, Eds.), Lecture Notes in Computer Science, vol. 2829 of *Lecture Notes in Computer Science*, Springer, 2003.

[27] Michalski, R.: A Theory and Methodology of Inductive Learning, in: *Machine Learning: an artificial intelligence approach* (R. Michalski, J. Carbonell, T. Mitchell, Eds.), vol. I, Morgan Kaufmann, San Mateo, CA, 1983, 83–134.

[28] Michalski, R.: Inferential Theory of Learning. Developing Foundations for Multistrategy Learning, in: *Machine Learning. A Multistrategy Approach* (R. Michalski, G. Tecuci, Eds.), vol. IV, Morgan Kaufmann, San Mateo, CA, 1994, 3–61.

[29] Mitchell, T.: Generalization as Search, *Artificial Intelligence*, **18**, 1982, 203–226.

[30] Muggleton, S., De Raedt, L.: Inductive Logic Programming, *Journal of Logic Programming*, **19/20**, 1994, 629–679.

[31] Nienhuys-Cheng, S., de Wolf, R.: *Foundations of Inductive Logic Programming*, vol. 1228 of *Lecture Notes in Artificial Intelligence*, Springer, 1997.

[32] Plaisted, D.: Theorem Proving with Abstraction, *Artificial Intelligence*, **16**, 1981, 47–108.

[33] Reiter, R.: Equality and Domain Closure in First Order Databases, *Journal of ACM*, **27**, 1980, 235–249.

[34] Rouveirol, C., Puget, J.: Beyond Inversion of Resolution, *Proceedings of the 7th International Conference on Machine Learning*, Morgan Kaufmann, Austin, TX, 1990, 122–130.

[35] Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., Ferilli, S.: A Logic Framework for the Incremental Inductive Synthesis of Datalog Theories, *Proceedings of the 7th International Workshop on Logic Program Synthesis and Transformation* (N. Fuchs, Ed.), LNCS, vol. 1463 of *LNCS*, Springer, 1998, 300–321.

[36] Stahl, I.: Predicate Invention in Inductive Logic Programming, *Advances in Inductive Logic Programming* (L. D. Raedt, Ed.), IOS Press, Amsterdam, 1996, 34–47.

[37] Stone, P., Veloso, M. M.: Layered Learning, *Machine Learning: ECML 2000, Proceedings of the 11th European Conference on Machine Learning*, vol. 1810, Springer, Berlin, 2000, 369–381.

[38] Subramanian, D., Genesereth, M.: The relevance of irrelevance, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1987, 416–422.

[39] Tecuci, G., Kodratoff, Y.: Apprenticeship learning in nonhomogeneous domain theories, in: *Machine Learning. A Multistrategy Approach* (Y. Kodratoff, R. Michalski, Eds.), vol. III, Morgan Kaufmann, San Mateo, CA, 1990, 514–552.

[40] Tenenberg, J.: Preserving Consistency Across Abstraction Mappings, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987, 1011–1014.

[41] Utgoff, P.: Shift of Bias for Inductive Concept Learning, in: *Machine Learning: an artificial intelligence approach* (R. Michalski, J. Carbonell, T. Mitchell, Eds.), vol. II, Morgan Kaufmann, Los Altos, CA, 1986, 107–148.

[42] Zucker, J., Bredeche, N., Saitta, L.: Abstracting Visual Percepts to Learn Concepts, *Proceedings of Abstraction, Reformulation and Approximation, 5th International Symposium, SARA 2002* (S. Koenig, R. C. Holte, Eds.), Lecture Notes in Computer Science, vol. 2371 of *Lecture Notes in Computer Science*, Springer, 2002, 256–273.

[43] Zucker, J.-D.: Semantic Abstraction for Concept Representation and Learning, *Proceedings of the 4th International Workshop on Multistrategy Learning* (R. S. Michalski, L. Saitta, Eds.), 1998, 157–164.