Building Rules on top of Ontologies? Inductive Logic Programming can help!

Francesca A. Lisi and Floriana Esposito

Dipartimento di Informatica, Università degli Studi di Bari
Via E. Orabona 4, 70125 Bari, Italy
{lisi, esposito}@di.uniba.it

Abstract. Acquiring and maintaining Semantic Web rules is very demanding and can be automated though partially by applying Machine Learning algorithms. In this paper we show that the form of Machine Learning known under the name of Inductive Logic Programming (ILP) can help. In particular, we take a critical look at two ILP proposals based on knowledge representation frameworks that integrate Description Logics and Horn Clausal Logic and draw from them general conclusions that can be considered as guidelines for further ILP research of interest to the Semantic Web.

1 Introduction

The *logical* layer of the Semantic Web architecture is currently one of the major sources of research challenges in the field. Indeed, whereas the mark-up language OWL for *ontologies* is already undergoing the standardization process at W3C, the debate around a unified language for *rules* is still ongoing. Proposals like SWRL[1] extend OWL with constructs inspired to Horn Clausal Logic (HCL) in order to meet the primary requirement of the logical layer: 'to build rules on top of ontologies'. Since the design of OWL has been based on Description Logics (DLs) [1] (more precisely on the $\mathcal{SH}$ family of very expressive DLs [9]), rule languages for the Semantic Web will most likely follow the tradition of *old* hybrid Knowledge Representation (KR) systems such as $\mathcal{AL}$-log [7] and Carin [11] that integrate DLs and HCL.

Acquiring and maintaining Semantic Web rules is very demanding and can be automated though partially by applying Machine Learning (ML) algorithms. The ML approach known under the name of Inductive Logic Programming (ILP) [22] seems particularly promising for the following reasons. ILP was born at the intersection of Concept Learning [19] and Logic Programming [17]. Thus it has been historically concerned with rule induction from examples within the KR framework of HCL and with the aim of prediction. The distinguishing feature of ILP, also with respect to other forms of Concept Learning, is the use of prior knowledge during the induction process. We claim that learning Semantic Web rules can be reformulated as learning rules by having ontologies as prior

[1] `http://www.w3.org/Submission/SWRL/`

knowledge. This may motivate an interest of the Semantic Web community in ILP. In this paper we take a critical look at the only two ILP attempts at learning rules within hybrid DL-HCL KR frameworks, the one for CARIN [28] and the other for $\mathcal{AL}$-log [12]. From the comparative analysis of the two we shall draw general conclusions that can be considered as guidelines for further ILP research of interest to the Semantic Web.

The paper is organized as follows. Section 2 provides the basic notions of DLs and HCL. Section 3 briefly describes different forms of integration of DLs and HCL. Section 4 first provides background information on the ILP methodological apparatus for non informed readers, then compares the two ILP proposals for hybrid DL-HCL formalisms. Section 5 concludes the paper with final remarks.

2 Logics behind Semantic Web Ontologies and Rules

Ontologies and rules for the Semantic Web are logically founded on Description Logics (DLs) and Horn Clausal Logic (HCL) respectively.

2.1 Description Logics

DLs are a family of decidable FOL fragments that allow for the specification of knowledge in terms of classes (*concepts*), binary relations between classes (*roles*), and instances (*individuals*) [2]. Complex concepts can be defined from atomic concepts and roles by means of constructors (see Table 1). E.g., concept descriptions in the basic DL $\mathcal{AL}$ are formed according to only the constructors of atomic negation, concept conjunction, value restriction, and limited existential restriction. The DLs $\mathcal{ALC}$ and $\mathcal{ALN}$ are members of the $\mathcal{AL}$ family. The former extends $\mathcal{AL}$ with (arbitrary) concept negation (also called complement and equivalent to having both concept union and full existential restriction), whereas the latter with number restriction. The DL $\mathcal{ALCNR}$ adds to the constructors inherited from $\mathcal{ALC}$ and $\mathcal{ALN}$ a further one: role intersection (see Table 1).

A DL knowledge base (KB) can state both is-a relations between concepts (*axioms*) and instance-of relations between individuals (resp. couples of individuals) and concepts (resp. roles) (*assertions*). Concepts and axioms form the so-called TBox whereas individuals and assertions form the so-called ABox[2]. The semantics of DLs is defined through a mapping to FOL. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for a DL KB consists of a domain $\Delta^{\mathcal{I}}$ and a mapping function $\cdot^{\mathcal{I}}$. In particular, individuals are mapped to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (*Unique Names Assumption* (UNA) [25]). Also the KB represents many different interpretations, i.e. all its models. This is coherent with the *Open World Assumption* (OWA) that holds in FOL semantics. The main reasoning task for a DL KB is the *consistency check* that is performed by applying decision procedures based on tableau calculus.

[2] When a DL-based ontology language is adopted, an ontology is nothing else than a TBox. If the ontology is populated, it corresponds to a whole DL KB, i.e. encompassing also an ABox.

Table 1. Syntax and semantics of the DL $\mathcal{ALCNR}$.

bottom (resp. top) concept	$\bot$ (resp. $\top$)	$\emptyset$ (resp. $\Delta^{\mathcal{I}}$)
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
(abstract) role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
(abstract) individual	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept intersection	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept union	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \; (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \; (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
at least number restriction	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \mid (x,y) \in R^{\mathcal{I}}\}\rvert \geq n\}$
at most number restriction	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \mid (x,y) \in R^{\mathcal{I}}\}\rvert \leq n\}$
role intersection	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
concept equivalence axiom	$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
concept subsumption axiom	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
concept assertion	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$\langle a, b \rangle : R$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

2.2 Horn Clausal Logic

The basic element in HCL is the *atom* of the form $p(t_i, \ldots, t_{k_i})$ such that each p is a predicate symbol and each t_j is a term. A *term* is either a constant or a variable or a more complex term obtained by applying a functor to simpler term. Constant, variable, functor and predicate symbols belong to mutually disjoint alphabets. A *literal* is an atom either negated or not. A *clause* is a universally quantified disjunction of literals. Usually the universal quantifiers are omitted to simplify notation. Alternative notations are a clause as set of literals and a clause as an implication. A *definite clause* is an implication of the form

$$\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m$$

where $m \geq 0$ and α_i are atoms, i.e. a Horn clause with exactly one positive literal. The right-hand side α_0 and the left-hand side $\alpha_1, \ldots, \alpha_m$ of the implication are called *head* and *body* of the clause, respectively. Note that the body is intended to be an existentially quantified conjunctive formula $\exists \alpha_1 \wedge \ldots \wedge \alpha_m$. Furthermore definite clauses with $m > 0$ and $m = 0$ are called *rules* and *facts* respectively.

Definite clauses are at the basis of logic programming [17] and deductive databases [4]. In particular, the language Datalog for deductive databases does not allow functors and recursion. A Datalog *program* D is a set of Datalog clauses. The predicates occurring in D are partitioned into two sets: the *extensional predicates* (EDB-predicates) and the *intensional predicates* (IDB-predicates). It is required that the predicate in the head of each rule in D be an IDB-predicate. Based on this distinction between extensional and intensional predicates, a Datalog program D can be divided into two parts, called extensional and intensional. The *extensional part*, denoted as $EDB(D)$, is the set of facts of D involving the extensional predicates, whereas the *intensional part* $IDB(D)$ is the set of all other clauses of D.

The *model-theoretic* **semantics** of Datalog is based on the notion of Herbrand interpretation. Let D be a Datalog program. The *Herbrand base* HB of D is the set of all atoms of the form $p(c_1, \ldots, c_k)$ such that p is a predicate of D and all the c_i are constants of D. We write EHB (resp. IHB) to denote the atoms of HB whose predicates are extensional (resp. intensional). An *Herbrand interpretation* for D is a subset of the Herbrand base HB. Let H be an Herbrand interpretation for D. A positive ground literal l is satisfied by H if $l \in H$. A negative ground literal $\neg l$ is satisfied by H if $l \notin H$. An Herbrand interpretation H for D is said to be a model of D if for every clause γ of D, for every ground instance $\gamma\prime$ of γ, at least one of the literals of $\gamma\prime$ is satisfied by H. The meaning of a Datalog program D is the set of its models. The intersection of all the models of D is itself a model of D, and in particular is the so-called *least Herbrand model*, i.e. it is the subset of each Herbrand model of D. The corresponding proof-theoretic semantics of Datalog is based on the *Closed World Assumption* (CWA).

Deductive **reasoning** with HCL is formalized in its proof theory. In clausal logic *resolution* comprises a single inference rule which, from any two clauses having an appropriate form, derives a new clause as their consequence. Resolution is sound: every resolvent is implied by its parents. It is also refutation complete: the empty clause is derivable by resolution from any set S of Horn clauses if S is unsatisfiable. The main reasoning task in Datalog is query answering. A *query* Q to a Datalog program D is a Datalog clause of the form

$$\leftarrow \alpha_1, \ldots, \alpha_m$$

where $m > 0$, and α_i is a Datalog atom. An *answer* to a query Q is a substitution θ for the variables of Q. An answer is correct with respect to the Datalog program D if $D \models Q\theta$. The *answer set* to a query Q is the set of answers to Q that are correct w.r.t. D and such that $Q\theta$ is ground. In other words the answer set to a query Q is the set of all ground instances of Q which are logical consequences of D. Answers are computed by refutation.

3 Combining Ontologies and Rules

The integration of Ontologies and Rules for the Semantic Web follows the tradition of KR research on *hybrid systems*, i.e. those systems which are constituted by two or more subsystems dealing with distinct portions of a single KB by performing specific reasoning procedures [8]. The motivation for investigating and developing such systems is to improve on two basic features of KR formalisms, namely *representational adequacy* and *deductive power*, by preserving the other crucial feature, i.e. *decidability*. Indeed DLs and HCL are FOL fragments incomparable as for the expressiveness [2] and the semantics [26][3] but combinable under certain conditions. In particular, combining DLs with HCL can easily yield to undecidability if the interface between them is not reduced (*safeness*). A *safe* interaction between the DL and the HCL part of an hybrid KB allows also to solve the semantic mismatch between DLs and HCL [20,27].

$\mathcal{AL}$-log [7] is a safe hybrid KR system that integrates $\mathcal{ALC}$ [29] and DATALOG [4]. In particular, variables occurring in the body of rules may be constrained with $\mathcal{ALC}$ concept assertions to be used as 'typing constraints'. This makes rules applicable only to explicitly named objects. Reasoning for $\mathcal{AL}$-log knowledge bases is based on *constrained SLD-resolution*, i.e. an extension of SLD-resolution with a tableau calculus for $\mathcal{ALC}$ to deal with constraints. Constrained SLD-resolution is *decidable* and runs in single non-deterministic exponential time. Constrained SLD-refutation is a complete and sound method for answering *ground* queries.

A comprehensive study of the effects of combining DLs and HCL (more precisely, Horn rules) can be found in [11]. Here the family **Carin** of hybrid languages is presented. Special attention is devoted to the DL $\mathcal{ALCNR}$. The results of the study can be summarized as follows: (i) answering conjunctive queries over $\mathcal{ALCNR}$ TBoxes is decidable, (ii) query answering in a logic obtained by extending $\mathcal{ALCNR}$ with non-recursive DATALOG rules, where both concepts and roles can occur in rule bodies, is also decidable, as it can be reduced to computing a union of conjunctive query answers, (iii) if rules are recursive, query answering becomes undecidable, (iv) decidability can be regained by disallowing certain combinations of constructors in the logic, and (v) decidability can be regained by requiring rules to be *role-safe*, where at least one variable from each role literal must occur in some non-DL-atom. As in $\mathcal{AL}$-log, query answering is decided using constrained resolution and a modified version of tableau calculus. As opposite to $\mathcal{AL}$-log, the hybridization in CARIN is not safe.

4 Learning Rules on top of Ontologies with ILP

4.1 The methodological apparatus of ILP

ILP was born at the intersection between Logic Programming and Concept Learning. From Logic Programming it has borrowed the KR framework. From Concept Learning it has inherited the inferential mechanisms for induction, the most prominent of which is *generalization*. In Concept Learning generalization is traditionally viewed as search through a partially ordered space of inductive hypotheses [19]. According to this vision, an inductive hypothesis is a clausal theory and the induction of a single clause requires (i) structuring, (ii) searching and (iii) bounding the space of clauses. To serve the purposes of this paper we focus on (i) by clarifying the notion of *ordering* for clausal spaces. One such ordering is *θ-subsumption* [23]: Given two clauses C and D, we say that C θ-subsumes D

[3] Remind that the OWA holds for DLs whereas CWA is valid in HCL. Note that the OWA and CWA have a strong influence on the results of reasoning.

if there exists a substitution θ, such that $C\theta \subseteq D$. In θ-subsumption the *background knowledge* that figures prominently in ILP problem settings is left out of consideration. Yet combining the examples with what we already know often allows for the construction of a more satisfactory theory that can be glanced from the examples by themselves. Given the usefulness of background knowledge, orders have been proposed that reckon with it, e.g. Plotkin's *relative subsumption* [24] and Buntine's *generalized subsumption* [3]. Relative subsumption applies to arbitrary clauses and the background knowledge may be an arbitrary finite set of clauses. Generalized subsumption only applies to definite clauses and the background knowledge should be a definite program. Each of these orders is related to some form of deduction. It can be shown by using these two forms of deduction that generalized subsumption is a weaker quasi-order than relative subsumption. Also, it can be shown that both relative and generalized subsumption reduce to ordinary subsumption in case of non-tautologous clauses and empty background knowledge. Generalized subsumption is of major interest to this paper. It is called *semantic generality* in contrast to θ-subsumption which is a purely syntactic generality. In the general case, semantic generality is undecidable and does not introduce a lattice on a set of clauses. Because of these problems, syntactic generality is more frequently used in ILP systems. Yet for DATALOG generalized subsumption is decidable and admits a least general generalization.

Once structured, the space of hypotheses can be searched (ii) by means of refinement operators. The definition of refinement operators presupposes the investigation of the properties of the various quasi-orders. In Shapiro's sense [30], a refinement operator is a function which computes a set of specializations of a clause. Specialization is suited for the direction of search in his approach. His kind of refinement operator has been therefore called a *downward* refinement operator in ILP. Dually, operators can be also defined to compute generalizations of clauses. These can be applied in a bottom-up search, so they have been named *upward* refinement operators. A good refinement operator should satisfy certain desirable properties [32]. We shall illustrate these properties for the case of downward refinement operators but analogous conditions are actually required to hold for upward refinement operators as well. Ideally, a downward refinement operator should compute only a *finite* set of specializations of each clause - otherwise it will be of limited use in practice. This condition is called *local finiteness*. Furthermore, it should be *complete*: every specialization should be reachable by a finite number of applications of the operator. Finally, it is better only to compute *proper* specializations of a clause, for otherwise repeated application of the operator might get stuck in a sequence of equivalent clauses, without ever achieving any real specialization. Operators that satisfy all these conditions simultaneously are called *ideal*. It has been shown that ideal upward and downward refinement operators do not exist for both full and Horn clausal languages ordered by either subsumption or the stronger orders (e.g. implication). In order to define a refinement operator for full clausal languages, it is necessary to drop one of the three properties of idealness. Locally finiteness and completeness are usually considered the most important properties. This means that locally finite and complete, but improper refinement operators can be defined for full clausal languages. On the other hand, in order to retain all the three properties of idealness, it seems that the only possibility is to restrict the search space. Hence, the definition of refinement operators is usually coupled with the specification of a declarative bias for bounding the space of clauses (iii). *Bias* concerns anything which constrains the search for theories [31]. Following [21] we will distinguish three kinds of bias: (a) *Language bias* that specifies constraints on the clauses in the search space; (b) *Search bias* that has to do with the way an ILP system searches its space of permitted clauses; (c) *Validation bias* that concerns the stopping criterion of the ILP system.

Induction with ILP generalizes from individual instances/observations in the presence of background knowledge, finding *valid hypotheses*. Validity depend on the underlying *setting*. At present, there exist several formalizations of induction in clausal logic. Two orthogonal dimensions are usually taken into account when classifying these formalizations [6]: the *scope of induction* (discrimination versus characterization) and the *representation of observations* (ground definite clauses versus ground unit clauses). *Discriminant induction* (also called *predictive induction*) aims at inducing hypotheses with discriminant power as required in tasks such as classification. In classification, observations encompass both positive and negative examples. *Characteristic induction* (also called *descriptive induction*) is more suitable for finding regularities in a given set of unclassified examples. This corresponds to learning from positive examples only. For a thorough discussion of differences between discriminant and characteristic induction see [18]. The second dimension affects the notion of *coverage*, i.e. the condition under which a hypothesis explains an observation. In *learning from entailment* (also called *learning from implications*), hypotheses are clausal theories, observations are ground definite clauses, and a hypothesis covers an observation if the hypothesis logically entails the observation. In *learning from interpretations*, hypotheses are clausal theories, observations are Herbrand interpretations (ground unit clauses) and a hypothesis covers an observation if the observation is a model for the hypothesis. A deeper investigation of learning from entailment and learning from interpretations can be found in [5].

4.2 ILP and DL-HCL formalisms

Learning in DL-HCL hybrid languages has very recently attracted some attention in the ILP community. Two ILP frameworks have been proposed that adopt a hybrid representation for both hypotheses and background knowledge.

In [28], the chosen language is CARIN-$\mathcal{ALN}$. The framework focuses on discriminant induction and adopts the ILP setting of learning from interpretations. The target concept is a unary DATALOG predicate, therefore hypotheses are represented as CARIN-$\mathcal{ALN}$ rules with a DATALOG literal in the head. The coverage relation of hypotheses against examples and the generality relation between two hypotheses are based on the existential entailment algorithm of CARIN. In particular, the generality relation is defined as an extension of Buntine's generalized subsumption [3]. Following [28], Kietz studies the learnability of CARIN-$\mathcal{ALN}$, thus providing a pre-processing method which enables ILP systems to learn Carin-$\mathcal{ALN}$ rules [10].

In [12], the representation and reasoning means come from $\mathcal{AL}$-log. Hypotheses are represented as constrained Datalog clauses that are linked, connected (or range-restricted), and compliant with the bias of Object Identity (OI)[4]. Note that this framework is general, meaning that it is valid whatever the scope of induction (description/prediction) is. Therefore the literal in the head of hypotheses represents a concept to be either discriminated from others (*discriminant induction*) or characterized (*characteristic induction*). The generality relation for one such hypothesis language is an adaptation of generalized subsumption [3], named $\mathcal{B}$-subsumption, to the $\mathcal{AL}$-log KR framework. It gives raise to a quasi-order and can be checked with a decidable procedure based on constrained SLD-resolution [14]. Coverage relations for both ILP settings of learning from interpretations and learning from entailment have been defined on the basis of query answering in $\mathcal{AL}$-log [13]. As opposite to [28], the framework has been implemented in an ILP system [16]. More precisely, an instantiation of it for the case of *characteristic induction from interpretations* has been considered. Indeed, the system supports a variant of a very popular data mining task - frequent pattern discovery - where rich prior conceptual knowledge is taken into account during the discovery process in order to find patterns at multiple levels of description granularity. The search through the space of patterns represented as unary conjunctive queries in $\mathcal{AL}$-log and organized according to $\mathcal{B}$-subsumption is performed by applying an ideal downward refinement operator [15].

5 Final remarks

Building rules on top of ontologies for the Semantic Web poses several challenges not only to KR researchers investigating suitable hybrid DL-HCL formalisms but also to the ML community which has been historically interested in application areas where the Knowledge Acquisition bottleneck is particularly severe. In this paper, we have provided a brief survey of ILP literature dealing with hybrid DL-HCL formalisms. From the comparative analysis of [28] and [12], a common feature emerges. Both proposals resort to Buntine's generalized subsumption, being it a *semantic* generality relation. Note that in both the extension of [3] to hybrid DL-HCL formalisms is not trivial. Following the guidelines of [28] and [12], new ILP frameworks can be designed to deal with more expressive hybrid DL-HCL languages. The augmented expressive power may be due to a more expressive DL (than $\mathcal{ALC}$ and $\mathcal{ALN}$), or a more expressive HCL fragment (than Datalog), or a looser integration between the DL and the HCL parts. We would like to emphasize that the *safeness* and the *decidability* of these formalisms are two desirable properties which are particularly appreciated both in ILP and in the Semantic Web application area.

[4] The OI bias can be considered as an extension of the UNA from the semantic level to the syntactic one of $\mathcal{AL}$-log. It can be the starting point for the definition of either an equational theory or a quasi-order for constrained Datalog clauses.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.
3. W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
4. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.
5. L. De Raedt. Logical Settings for Concept-Learning. *Artificial Intelligence*, 95(1):187–201, 1997.
6. L. De Raedt and L. Dehaspe. Clausal Discovery. *Machine Learning*, 26(2–3):99–146, 1997.
7. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
8. A.M. Frisch and A.G. Cohn. Thoughts and afterthoughts on the 1988 workshop on principles of hybrid reasoning. *AI Magazine*, 11(5):84–87, 1991.
9. I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
10. J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 117–132. Springer, 2003.
11. A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
12. F.A. Lisi. Principles of Inductive Reasoning on the Semantic Web: A Framework for Learning in $\mathcal{AL}$-log. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3703 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2005.
13. F.A. Lisi and F. Esposito. Efficient Evaluation of Candidate Hypotheses in $\mathcal{AL}$-log. In R. Camacho, R. King, and A. Srinivasan, editors, *Inductive Logic Programming*, volume 3194 of *Lecture Notes in Artificial Intelligence*, pages 216–233. Springer, 2004.
14. F.A. Lisi and D. Malerba. Bridging the Gap between Horn Clausal Logic and Description Logics in Inductive Learning. In A. Cappelli and F. Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence*, volume 2829 of *Lecture Notes in Artificial Intelligence*, pages 49–60. Springer, 2003.
15. F.A. Lisi and D. Malerba. Ideal Refinement of Descriptions in $\mathcal{AL}$-log. In T. Horvath and A. Yamamoto, editors, *Inductive Logic Programming*, volume 2835 of *Lecture Notes in Artificial Intelligence*, pages 215–232. Springer, 2003.
16. F.A. Lisi and D. Malerba. Inducing Multi-Level Association Rules from Multiple Relations. *Machine Learning*, 55:175–210, 2004.
17. J.W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.
18. R.S. Michalski. A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume I. Morgan Kaufmann, San Mateo, CA, 1983.
19. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

20. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In S.A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *The Semantic Web*, volume 3298 of *Lecture Notes in Computer Science*, pages 549–563. Springer, 2004.
21. C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, 1996.
22. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
23. G.D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
24. G.D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6:101–121, 1971.
25. R. Reiter. Equality and domain closure in first order databases. *Journal of ACM*, 27:235–249, 1980.
26. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1), 2005.
27. R. Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.
28. C. Rouveirol and V. Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.
29. M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
30. E.Y. Shapiro. Inductive inference of theories from facts. Technical Report 624, Dept. of Computer Science, Yale University, 1981.
31. P.E. Utgoff and T.M. Mitchell. Acquisition of appropriate bias for inductive concept learning. In *Proceedings of the 2nd National Conference on Artificial Intelligence*, pages 414–418, Los Altos, CA, 1982. Morgan Kaufmann.
32. P.R.J. van der Laag. *An Analysis of Refinement Operators in Inductive Logic Programming*. Ph.D. Thesis, Erasmus University, Rotterdam, The Netherlands, 1995.