A Formal Characterization of Concept Learning in Description Logics

Francesca A. Lisi

Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy
lisi@di.uniba.it

Abstract. Among the inferences studied in Description Logics (DLs), induction has been paid increasing attention over the last decade. Indeed, useful non-standard reasoning tasks can be based on the inductive inference. Among them, Concept Learning is about the automated induction of a description for a given concept starting from classified instances of the concept. In this paper we present a formal characterization of Concept Learning in DLs which relies on recent results in Knowledge Representation and Machine Learning.

1 Introduction

Building and maintaining large ontologies pose several challenges to Knowledge Representation (KR) because of their size. In DL ontologies, although standard inferences help structuring the knowledge base (KB), e.g., by automatically building a concept hierarchy, they are, for example, not sufficient when it comes to (automatically) generating new concept descriptions from given ones. They also fail if the concepts are specified using different vocabularies (i.e. sets of concept names and role names) or if they are described on different levels of abstraction. Altogether it has turned out that for building and maintaining large DL KBs, besides the standard inferences, additional so-called *non-standard inferences* are required [27,19]. Among them, the first ones to be studied have been the Least Common Subsumer (LCS) of a set concepts [3] and the Most Specific Concept (MSC) of an individual [32,10,20,1]. Very recently, a unified framework for non-standard reasoning services in DLs has been proposed [8]. It is based on the use of second-order sentences in DLs [7] as the unifying definition model for all those *constructive reasoning* tasks which rely on specific optimality criteria to build up the objective concept. E.g., LCS is one of the cases considered for one such reformulation in terms of optimal solution problems.

Since [27], much work has been done in DL reasoning to support the construction and maintenance of DL KBs. This work has been more or less explicitly related to *induction*. E.g., the notion of LCS has subsequently been used for the bottom-up induction of CLASSIC concept descriptions from examples [5,6]. Induction has been widely studied in Machine Learning (ML). Therefore it does not come as a surprise that the problem of finding an appropriate concept description for given concept instances, reformulated as a problem of inductive

learning from examples, has been faced in ML, initially attacked by heuristic means [6,18,14] and more recently in a formal manner [2,12,13,22] by adopting the methods and the techniques of that ML approach known as Concept Learning.

In this paper, we present a formal characterization of Concept Learning in DLs which relies on recent results in KR and ML. Notably, the proposed formulation can be justified by observing that the inductive inference deals with finding - or constructing - a concept. Therefore, non-standard reasoning services based on induction can be considered as constructive reasoning tasks. Starting from this assumption, and inspired by Colucci *et al*'s framework, Concept Learning is modeled as a second-order concept expression in DLs and reformulated in terms that allow for a construction possibly subject to some optimality criteria.

The paper is structured as follows. Section 2 is devoted to preliminaries on Concept Learning according to the ML tradition. Section 3 defines the Concept Learning problem statement in the KR context. Section 4 proposes a reformulation of Concept Learning as a constructive reasoning task in DLs. Section 5 concludes the paper with final remarks and directions of future work.

2 Preliminaries

2.1 Machine Learning

The goal of ML is the design and development of algorithms that allow computers to evolve behaviors based on empirical data [30]. The automation of the *inductive inference* plays a key role in ML algorithms, though other inferences such as abduction and analogy are also considered. The effect of applying inductive ML algorithms depends on whether the *scope* of induction is discrimination or characterization [28]. *Discriminant induction* aims at inducing hypotheses with discriminant power as required in tasks such as classification. In classification, observations to learn from are labeled as positive or negative instances of a given class. *Characteristic induction* is more suitable for finding regularities in a data set. This corresponds to learning from positive examples only.

Ideally, the ML task is to discover an operational description of a *target function* $f : X \to Y$ which maps elements in the *instance space* X to the values of a set Y. The target function is unknown, meaning that only a set $\mathcal{D}$ (the *training data*) of points of the form $(x, f(x))$ is provided. However, it may be very difficult in general to learn such a description of f perfectly. In fact, ML algorithms are often expected to acquire only some approximation $\hat{f}$ to f by searching a very large space $\mathcal{H}$ of possible hypotheses (the *hypothesis space*) which depend on the representation chosen for f (the *language of hypotheses*). The output approximation is the one that best fits $\mathcal{D}$ according to a *scoring function* $score(f, \mathcal{D})$. It is assumed that any hypothesis $h \in \mathcal{H}$ that approximates f well w.r.t. a large set of training cases will also approximate it well for new unobserved cases. These notions have been mathematically formalized in computational learning theory within the Probably Approximately Correct (PAC) learning framework [36].

Summing up, given a hypothesis space $\mathcal{H}$ and a training data set $\mathcal{D}$, ML algorithms are designed to find an approximation $\hat{f}$ of a target function f s.t.:

1. $\hat{f} \in \mathcal{H}$;
2. $\hat{f}(\mathcal{D}) \approx f(\mathcal{D})$; and/or
3. $\hat{f} = argmax_{f \in \mathcal{H}} score(f, \mathcal{D})$.

It has been recently stressed that the first two requirements impose constraints on the possible hypotheses, thus defining a *Constraint Satisfaction Problem* (CSP), whereas the third requirement involves the optimization step, thus turning the CSP into an *Optimization Problem* (OP) [9]. We shall refer to the ensemble of constraints and optimization criterion as the model of the learning task. Models are almost by definition declarative and it is useful to distinguish the CSP, which is concerned with finding a solution that satisfies all the constraints in the model, from the OP, where one also must guarantee that the found solution be optimal w.r.t. the optimization function. Examples of typical CSPs in the ML context include Concept Learning for reasons that will become clearer by reading the following subsection.

2.2 Concept Learning

Concept Learning deals with inferring the general definition of a category based on members (positive examples) and nonmembers (negative examples) of this category. Here, the target is a boolean-valued function $f : X \to \{0, 1\}$, *i.e.* a *concept*. When examples of the target concept are available, the resulting ML task is said *supervised*, otherwise it is called *unsupervised*. The positive examples are those instances with $f(x) = 1$, and negative ones are those with $f(x) = 0$.

In Concept Learning, the key inferential mechanism for induction is *generalization as search* through a partially ordered space of inductive hypotheses [29]. Hypotheses may be ordered from the most general ones to the most specific ones. We say that an instance $x \in X$ satisfies a hypothesis $h \in \mathcal{H}$ if and only if $h(x) = 1$. Given two hypotheses h_i and h_j, h_i is more general than or equal to h_j (written $h_i \succeq_g h_j$, where $\succeq_g$ denotes a *generality relation*) if and only if any instance satisfying h_j, also satisfies h_i. Note that it may not be always possible to compare two hypotheses with a generality relation: the instances satisfied by the hypotheses may intersect, and not necessarily be subsumed by one another. The relation $\succeq_g$ defines a *partial order* (*i.e.*, it is reflexive, antisymmetric, and transitive) over the space of hypotheses.

A hypothesis h that correctly classifies all training examples is called consistent with these examples. For a consistent hypothesis h it holds that $h(x) = f(x)$ for each instance x. The set of all hypotheses consistent with the training examples is called the *version space* with respect to $\mathcal{H}$ and $\mathcal{D}$. Concept Learning algorithms may use the hypothesis space structure to efficiently search for relevant hypotheses. E.g., they may perform a specific-to-general search through the hypothesis space along one branch of the partial ordering, to find the most specific hypothesis consistent with the training examples. Another well known approach, *candidate elimination*, consists of computing the version space by an incremental computation of the sets of maximally specific and maximally general hypotheses. An important issue in Concept Learning is associated with the

so-called *inductive bias*, *i.e.* the set of assumptions that the learning algorithm uses for prediction of outputs given previously unseen inputs. These assumptions represent the nature of the target function, so the learning approach implicitly makes assumptions on the correct output for unseen examples.

Inductive Logic Programming (ILP) was born at the intersection between Concept Learning and the field of Logic Programming [31]. From Concept Learning it has inherited the inferential mechanisms for induction [33]. However, a distinguishing feature of ILP with respect to other forms of Concept Learning is the use of prior knowledge of the domain of interest, called *background knowledge* (BK), during the search for hypotheses. Due to the roots in Logic Programming, ILP was originally concerned with Concept Learning problems where both hypotheses, observations and BK are expressed with first-order Horn rules (usually DATALOG for computational reasons). E.g., FOIL is a popular ILP algorithm for learning sets of DATALOG rules for classification purposes [34]. It performs a greedy search in order to maximize an *information gain* function. Therefore, FOIL implements an OP version of Concept Learning.

Over the last decade, ILP has widened its scope significantly, by considering, e.g., learning in DLs (see next section) as well as within those hybrid KR frameworks integrating DLs and first-order clausal languages [35,17,25,26].

3 Learning Concepts in Description Logics

Early work on the application of ML to DLs essentially focused on demonstrating the PAC-learnability for various terminological languages derived from CLASSIC. In particular, Cohen and Hirsh investigate the CORECLASSIC DL proving that it is not PAC-learnable [4] as well as demonstrating the PAC-learnability of its sublanguages, such as C-CLASSIC [5], through the bottom-up LCSLEARN algorithm. It is also worth mentioning unsupervised learning methodologies for DL concept descriptions, whose prototypical example is KLUSTER [18], a polynomial-time algorithm for the induction of BACK terminologies. More recently, algorithms have been proposed that follow the *generalization as search* approach by extending the methodological apparatus of ILP to DL languages [2,11,12,21,22]. Supervised (resp., unsupervised) learning systems, such as YINYANG [16] and $\mathcal{DL}$-LEARNER [23], have been implemented. Based on a set of refinement operators borrowed from YINYANG and $\mathcal{DL}$-LEARNER, a new version of the FOIL algorithm, named $\mathcal{DL}$-FOIL, has been proposed [13]. In $\mathcal{DL}$-FOIL, the information gain function takes into account the Open World Assumption (OWA) holding in DLs. Indeed, many instances may be available which cannot be ascribed to the target concept nor to its negation. This requires a different setting to ensure a special treatment of the unlabeled individuals.

3.1 The Problem Statement

In this section, the supervised Concept Learning problem in the DL setting is formally defined. For the purpose, we denote:

– $\mathcal{T}$ and $\mathcal{A}$ are the TBox and the ABox, respectively, of a $\mathcal{DL}$ KB $\mathcal{K}$
– $\mathsf{Ind}(\mathcal{A})$ is the set of all individuals occurring in $\mathcal{A}$
– $\mathsf{Retr}_{\mathcal{K}}(C)$ is the set of all individuals occurring in $\mathcal{A}$ that are an instance of a given concept C w.r.t. $\mathcal{T}$
– $\mathsf{Ind}_C^+(\mathcal{A}) = \{a \in \mathsf{Ind}(\mathcal{A}) \mid C(a) \in \mathcal{A}\} \subseteq \mathsf{Retr}_{\mathcal{K}}(C)$
– $\mathsf{Ind}_C^-(\mathcal{A}) = \{b \in \mathsf{Ind}(\mathcal{A}) \mid \neg C(b) \in \mathcal{A}\} \subseteq \mathsf{Retr}_{\mathcal{K}}(\neg C)$

These sets can be easily computed by resorting to *retrieval inference services* usually available in DL systems.

Definition 1 (Concept Learning). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{DL}$ KB. Given:*

– *a (new) target concept name C*
– *a set of positive and negative examples $\mathsf{Ind}_C^+(\mathcal{A}) \cup \mathsf{Ind}_C^-(\mathcal{A}) \subseteq \mathsf{Ind}(\mathcal{A})$ for C*
– *a concept description language $\mathcal{DL}_{\mathcal{H}}$*

the Concept Learning *problem is to find a concept definition $C \equiv D$ such that $D \in \mathcal{DL}_{\mathcal{H}}$ satisfies the following conditions*

Completeness $\mathcal{K} \models D(a) \quad \forall a \in \mathsf{Ind}_C^+(\mathcal{A})$ *and*
Consistency $\mathcal{K} \models \neg D(b) \quad \forall b \in \mathsf{Ind}_C^-(\mathcal{A})$

Note that the definition given above provides the CSP version of the supervised Concept Learning problem. However, as already mentioned, Concept Learning can be regarded also as an OP. Algorithms such as $\mathcal{DL}$-Foil testify the existence of optimality criteria to be fulfilled in Concept Learning besides the conditions of completeness and consistency.

3.2 The Solution Strategy

In Def. 1, we have considered a language of hypotheses $\mathcal{DL}_{\mathcal{H}}$ that allows for the generation of concept definitions in any $\mathcal{DL}$. These definitions can be organized according to the concept subsumption relationship $\sqsubseteq$. Since $\sqsubseteq$ induces a quasi-order (*i.e.*, a reflexive and transitive relation) on $\mathcal{DL}_{\mathcal{H}}$ [2,11], the problem stated in Def. 1 can be cast as the search for a correct (i.e., complete and consistent) concept definition in $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$ according to the *generalization as search* approach in Mitchell's vision. In such a setting, one can define suitable techniques (called *refinement operators*) to traverse $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$ either top-down or bottom-up.

Definition 2 (Refinement operator in DLs). *Given a quasi-ordered search space $(\mathcal{DL}_{\mathcal{H}}, \sqsubseteq)$*

– *a* downward refinement operator *is a mapping $\rho : \mathcal{DL}_{\mathcal{H}} \to 2^{\mathcal{DL}_{\mathcal{H}}}$ such that*

$$\forall C \in \mathcal{DL}_{\mathcal{H}} \quad \rho(C) \subseteq \{D \in \mathcal{DL}_{\mathcal{H}} \mid D \sqsubseteq C\}$$

– *an* upward refinement operator *is a mapping $\delta : \mathcal{DL}_{\mathcal{H}} \to 2^{\mathcal{DL}_{\mathcal{H}}}$ such that*

$$\forall C \in \mathcal{DL}_{\mathcal{H}} \quad \delta(C) \subseteq \{D \in \mathcal{DL}_{\mathcal{H}} \mid C \sqsubseteq D\}$$

Definition 3 (Refinement chain in DLs). *Given a downward (resp., upward) refinement operator ρ (resp., δ) for a quasi-ordered search space $(\mathcal{DL}_\mathcal{H}, \sqsubseteq)$, a* refinement chain *from $C \in \mathcal{DL}_\mathcal{H}$ to $D \in \mathcal{DL}_\mathcal{H}$ is a sequence*

$$C = C_0, C_1, \ldots, C_n = D$$

such that $C_i \in \rho(C_{i-1})$ (resp., $C_i \in \delta(C_{i-1})$) for every $1 \leq i \leq n$.

Note that, given $(\mathcal{DL}, \sqsubseteq)$, there is an infinite number of generalizations and specializations. Usually one tries to define refinement operators that can traverse efficiently throughout the hypothesis space in pursuit of one of the correct definitions (w.r.t. the examples that have been provided).

Definition 4 (Properties of refinement operators in DLs). *A downward refinement operator ρ for a quasi-ordered search space $(\mathcal{DL}_\mathcal{H}, \sqsubseteq)$ is*

- (locally) finite *iff $\rho(C)$ is finite for all concepts $C \in \mathcal{DL}_\mathcal{H}$.*
- redundant *iff there exists a refinement chain from a concept $C \in \mathcal{DL}_\mathcal{H}$ to a concept $D \in \mathcal{DL}_\mathcal{H}$, which does not go through some concept $E \in \mathcal{DL}_\mathcal{H}$ and a refinement chain from C to a concept equal to D, which does go through E.*
- proper *iff for all concepts $C, D \in \mathcal{DL}_\mathcal{H}$, $D \in \rho(C)$ implies $C \not\equiv D$.*
- complete *iff, for all concepts $C, D \in \mathcal{DL}_\mathcal{H}$ with $C \sqsubset D$, a concept $E \in \mathcal{DL}_\mathcal{H}$ with $E \equiv C$ can be reached from D by ρ.*
- weakly complete *iff, for all concepts $C \in \mathcal{DL}_\mathcal{H}$ with $C \sqsubset \top$, a concept $E \in \mathcal{DL}_\mathcal{H}$ with $E \equiv C$ can be reached from $\top$ by ρ.*

The corresponding notions for upward refinement operators are defined dually.

Designing a refinement operator needs to make decisions on which properties are most useful in practice regarding the underlying learning algorithm. Considering the properties reported in Def. 4, it has been shown that the most feasible property combination for Concept Learning in expressive DLs such as $\mathcal{ALC}$ is $\{weakly\ complete, complete, proper\}$ [21]. Only for less expressive DLs like $\mathcal{EL}$, *ideal*, i.e. complete, proper and finite, operators exist [24].

4 Concept Learning as Constructive Reasoning in DLs

In this section, we formally characterize Concept Learning in DLs by emphasizing the constructive nature of the inductive inference.

4.1 Second-order Concept Expressions

We assume to start from the syntax of any Description Logic $\mathcal{DL}$ where N_c, N_r, and N_o are the alphabet of concept names, role names and individual names, respectively. In order to write second-order formulas, we introduce a set $\mathsf{N}_x = X_0, X_1, X_2, ...$ of concept variables, which we can quantify over. We denote by $\mathcal{DL}_X$ the language of concept terms obtained from $\mathcal{DL}$ by adding N_x.

Definition 5 (Concept term). *A* concept term *in $\mathcal{DL}_X$ is a concept formed according to the specific syntax rules of $\mathcal{DL}$ augmented with the additional rule $C \longrightarrow X$ for $X \in \mathsf{N}_x$.*

Since we are not interested in second-order DLs as themselves, we restrict our language to particular existential second-order formulas of interest to this paper. In particular, we allow formulas involving an ABox. By doing so, we can easily model the computation of, e.g., the MSC, which was left out as future work in Colucci *et al.*'s framework. This paves the way to the modeling of Concept Learning as shown in the next subsection.

Definition 6 (Concept expression). *Let $a_1, \ldots, a_m \in \mathcal{DL}$ be individuals, $C_1, \ldots, C_m, D_1, \ldots, D_m \in \mathcal{DL}_X$ be concept terms containing concept variables $X_0, X_1, \ldots, X_n$. A* concept expression *Γ in $\mathcal{DL}_X$ is a conjunction*

$$\begin{array}{c}(C_1 \sqsubseteq D_1) \wedge \ldots \wedge (C_l \sqsubseteq D_l) \wedge (C_{l+1} \not\sqsubseteq D_{l+1}) \wedge \ldots \wedge (C_m \not\sqsubseteq D_m) \wedge \\ (a_1 : D_1) \wedge \ldots \wedge (a_l : D_l) \wedge (a_{l+1} : \neg D_{l+1}) \wedge \ldots \wedge (a_m : \neg D_m)\end{array} \tag{1}$$

of (negated or not) concept subsumptions and concept assertions with $1 \leq l \leq m$.

We use *General Semantics*, also called Henkin semantics, for interpreting concept variables [15]. In such a semantics, variables denoting unary predicates can be interpreted only by *some subsets* among all the ones in the powerset of the domain $2^{\Delta^{\mathcal{I}}}$ - instead, in Standard Semantics a concept variable could be interpreted as any subset of $\Delta^{\mathcal{I}}$. Adapting General Semantics to our problem, the structure we consider is exactly the sets interpreting concepts in $\mathcal{DL}$. That is, the interpretation $X^{\mathcal{I}}$ of a concept variable $X \in \mathcal{DL}_X$ must coincide with the interpretation $E^{\mathcal{I}}$ of some concept $E \in \mathcal{DL}$. The interpretations we refer to in the following definition are of this kind.

Definition 7 (Satisfiability). *A concept expression Γ of the form (1) is* satisfiable *in $\mathcal{DL}$ iff there exist $n+1$ concepts $E_0, \ldots, E_n \in \mathcal{DL}$ such that, extending the semantics of $\mathcal{DL}$ for each interpretation $\mathcal{I}$, with: $(X_i)^{\mathcal{I}} = (E_i)^{\mathcal{I}}$ for $i = 0, \ldots, n$, it holds that*

1. *for each $j = 1, \ldots, l$, and every interpretation $\mathcal{I}$, $(C_j)^{\mathcal{I}} \subseteq (D_j)^{\mathcal{I}}$ and $(a_j)^{\mathcal{I}} \in (D_j)^{\mathcal{I}}$, and*
2. *for each $j = l+1, \ldots, m$, there exists an interpretation $\mathcal{I}$ s.t. $(C_j)^{\mathcal{I}} \not\subseteq (D_j)^{\mathcal{I}}$ and $(a_j)^{\mathcal{I}} \notin (D_j)^{\mathcal{I}}$*

Otherwise, Γ is said to be unsatisfiable *in $\mathcal{DL}$.*

Definition 8 (Solution). *If a concept expression Γ of the form (1) is satisfiable in $\mathcal{DL}$, then $\langle E_0, \ldots, E_n \rangle$ is a* solution *for Γ. Moreover, we say that the formula*

$$\exists X_0 \cdots \exists X_n . \Gamma \tag{2}$$

is true *in $\mathcal{DL}$ if there exist at least a solution for Γ, otherwise it is* false.

4.2 Modeling Concept Learning with Second-Order DLs

It has been pointed out that the constructive reasoning tasks can be divided into two main categories: tasks for which we just need to compute a concept (or a set of concepts) and those for which we need to find a concept (or a set of concepts) according to some minimality/maximality criteria [8]. In the first case, we have a set of solutions while in the second one we also have a set of sub-optimal solutions to the main problem. E.g., the set of sub-optimal solutions in LCS is represented by the common subsumers. Both MSC and Concept Learning belong to this second category of constructive reasoning tasks. We remind the reader that MSC can be easily reduced to LCS for DLs that admit the one-of concept constructor. However, this reduction is not trivial for the general case. Hereafter, first, we show how to model MSC in terms of formula (2). This step is to be considered as functional to the modeling of Concept Learning.

Most Specific Concept Intuitively, the MSC of individuals described in an ABox is a concept description that represents all the properties of the individuals including the concept assetions they occur in and their relationship to other individuals. Similar to the LCS, the MSC is uniquely determined up to equivalence. More precisely, the set of most specific concepts of individuals $a_1, \ldots, a_k \in \mathcal{DL}$ forms an equivalence class, and if S is defined to be the set of all concept descriptions that have $a_1, \ldots, a_k$ as their instance, then this class is the least element in $[S]$ w.r.t. a partial ordering $\preceq$ on equivalence classes induced by the quasi ordering $\sqsubseteq$. Analogously to the LCS, we refer to one of its representatives by $\mathsf{MSC}(a_1, \ldots, a_k)$. The MSC need not exist. Three different phenomena may cause the non existence of a least element in $[S]$, and thus, a MSC:

1. $[S]$ might be empty, or
2. $[S]$ might contain different minimal elements, or
3. $[S]$ might contain an infinite decreasing chain $[D_1] \succ [D_2] \cdots$.

A concept E is not the MSC of $a_1, \ldots, a_k$ iff the following formula is true in $\mathcal{DL}$:

$$\exists X.(a_1 : X) \wedge \ldots \wedge (a_k : X) \wedge (X \sqsubseteq E) \wedge (E \not\sqsubseteq X) \tag{3}$$

that is, E is not the MSC if there exists a concept X which is a most specific concept, and is strictly more specific than E.

Concept Learning Following Def. 1, we assume that $\mathsf{Ind}_C^+(\mathcal{A}) = \{a_1, \ldots, a_m\}$ and $\mathsf{Ind}_C^-(\mathcal{A}) = \{b_1, \ldots, b_n\}$. A concept $D \in \mathcal{DL}_{\mathcal{H}}$ is a correct concept definition for the target concept name C w.r.t. $\mathsf{Ind}_C^+(\mathcal{A})$ and $\mathsf{Ind}_C^-(\mathcal{A})$ iff it is a solution for the following second-order concept expression:

$$(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (a_1 : X) \wedge \ldots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \ldots \wedge (b_n : \neg X) \tag{4}$$

The CSP version of the task is therefore modeled with the following formula.

$$\exists X.(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (a_1 : X) \wedge \ldots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \ldots \wedge (b_n : \neg X) \tag{5}$$

A simple OP version of the task could be modeled with the formula:

$$\begin{array}{c} \exists X.(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (X \sqsubseteq E) \wedge (E \not\sqsubseteq X) \wedge \\ (a_1 : X) \wedge \ldots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \ldots \wedge (b_n : \neg X) \end{array} \tag{6}$$

which asks for solutions that are compliant with a minimality criterion involving concept subsumption checks. Therefore, a concept $E \in \mathcal{DL}_{\mathcal{H}}$ is not a correct concept definition for C w.r.t. $\mathsf{Ind}^{+}_{C}(\mathcal{A})$ and $\mathsf{Ind}^{-}_{C}(\mathcal{A})$ if there exists a concept X which is a most specific concept, and is strictly more specific than E.

5 Conclusions

In this paper, we have provided a formal characterization of Concept Learning in DLs according to a declarative modeling language which abstracts from the specific algorithms used to solve the task. To this purpose, we have defined a fragment of second-order logic under the general semantics which allows to express formulas involving concept assertions from an ABox. One such fragment enables us to cover the general case of MSC as well. Also, as a minor contribution, we have suggested that the *generalization as search* approach to Concept Learning in Mitchell's vision is just that unifying framework necessary for accompanying the declarative modeling language proposed in this paper with a way of computing solutions to the problems declaratively modeled with this language. More precisely, the computational method we refer to in this paper is based on the iterative application of suitable refinement operators. Since many refinement operators for DLs are already available in the literature, the method can be designed such that it can be instantiated with a refinement operator specifically defined for the DL in hand.

The preliminary results reported in this paper open a promising direction of research at the intersection of KR and ML. For this research we have taken inspiration from recent results in both areas. On one hand, Colucci *et al.*'s work provides a procedure which combines Tableaux calculi for DLs with rules for the substitution of concept variables in second-order concept expressions [8]. On the other hand, De Raedt *et al.*'s work shows that off-the-shelf constraint programming techniques can be applied to various ML problems, once reformulated as CSPs and OPs [9]. Interestingly, both works pursue a unified view on the inferential problems of interest to the respective fields of research. This match of research efforts in the two fields has motivated the work presented in this paper which, therefore, moves a step towards bridging the gap between KR and ML in areas such as the maintenance of KBs where the two fields have already produced interesting results though mostly indipendently from each other. New questions and challenges are raised by the cross-fertilization of these results. In the future, we intend to investigate how to express optimality criteria such as the information gain function within the second-order concept expressions and how the *generalization as search* approach can be effectively integrated with second-order calculus.

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Gottlob, G., Walsh, T. (eds.) IJCAI'03: Proceedings of the 18th International Joint Conference on Artificial intelligence. pp. 319–324. Morgan Kaufmann Publishers (2003)
2. Badea, L., Nienhuys-Cheng, S.: A refinement operator for description logics. In: Cussens, J., Frisch, A. (eds.) Inductive Logic Programming, Lecture Notes in Artificial Intelligence, vol. 1866, pp. 40–59. Springer-Verlag (2000)
3. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Proc. of the 10th National Conf. on Artificial Intelligence. pp. 754–760. The AAAI Press / The MIT Press (1992)
4. Cohen, W.W., Hirsh, H.: Learnability of description logics. In: Haussler, D. (ed.) Proc. of the 5th Annual ACM Conf. on Computational Learning Theory. pp. 116–127. ACM (1992)
5. Cohen, W.W., Hirsh, H.: Learning the CLASSIC description logic: Theoretical and experimental results. In: Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94). pp. 121–133. Morgan Kaufmann (1994)
6. Cohen, W.W., Hirsh, H.: The learnability of description logics with equality constraints. Machine Learning 17(2-3), 169–199 (1994)
7. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: Second-order description logics: Semantics, motivation, and a calculus. In: Haarslev, V., Toman, D., Weddell, G.E. (eds.) Proc. of the 23rd Int. Workshop on Description Logics (DL 2010). CEUR Workshop Proceedings, vol. 573. CEUR-WS.org (2010)
8. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: A unified framework for non-standard reasoning services in description logics. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI 2010 - 19th European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 479–484. IOS Press (2010)
9. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: Fox, M., Poole, D. (eds.) Proc. of the 24th AAAI Conference on Artificial Intelligence. AAAI Press (2010)
10. Donini, F.M., Lenzerini, M., Nardi, D.: An efficient method for hybrid deduction. In: ECAI. pp. 246–252 (1990)
11. Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) The Semantic Web - ISWC 2004: Third International Semantic Web Conference. Lecture Notes in Computer Science, vol. 3298, pp. 441–455. Springer (2004)
12. Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Concept formation in expressive description logics. In: Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) Machine Learning: ECML 2004. Lecture Notes in Computer Science, vol. 3201, pp. 99–110. Springer (2004)
13. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) Inductive Logic Programming. Lecture Notes in Computer Science, vol. 5194, pp. 107–121. Springer (2008)
14. Frazier, M., Pitt, L.: CLASSIC learning. Machine Learning 25(2-3), 151–193 (1996)
15. Henkin, L.: Completeness in the theory of types. Journal of Symbolic Logic 15(2), 81–91 (1950)

16. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence 26(2), 139–159 (2007)
17. Kietz, J.: Learnability of description logic programs. In: Matwin, S., Sammut, C. (eds.) Inductive Logic Programming. Lecture Notes in Artificial Intelligence, vol. 2583, pp. 117–132. Springer (2003)
18. Kietz, J.U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning 14(1), 193–217 (1994)
19. Küsters, R.: Non-Standard Inferences in Description Logics, Lecture Notes in Computer Science, vol. 2100. Springer (2001)
20. Küsters, R., Molitor, R.: Approximating most specific concepts in description logics with existential restrictions. AI Communications 15(1), 47–59 (2002)
21. Lehmann, J., Hitzler, P.: Foundations of refinement operators for description logics. In: Blockeel, H., Ramon, J., Shavlik, J.W., Tadepalli, P. (eds.) Inductive Logic Programming. Lecture Notes in Artificial Intelligence, vol. 4894, pp. 161–174. Springer (2008)
22. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning 78(1-2), 203–250 (2010)
23. Lehmann, J.: DL-Learner: Learning concepts in description logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
24. Lehmann, J., Haase, C.: Ideal downward refinement in the $\mathcal{EL}$ description logic. In: De Raedt, L. (ed.) Inductive Logic Programming. Lecture Notes in Computer Science, vol. 5989, pp. 73–87 (2010)
25. Lisi, F.A.: Building rules on top of ontologies for the semantic web with inductive logic programming. Theory and Practice of Logic Programming 8(03), 271–300 (2008)
26. Lisi, F.A.: Inductive logic programming in databases: From Datalog to $\mathcal{DL}$+log. Theory and Practice of Logic Programming 10(3), 331–359 (2010)
27. McGuinness, D., Patel-Schneider, P.: Usability issues in knowledge representation systems. In: Mostow, J., Rich, C. (eds.) Proc. of the 15th National Conf. on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference. pp. 608–614. AAAI Press / The MIT Press (1998)
28. Michalski, R.S.: A theory and methodology of inductive learning. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning: an artificial intelligence approach, vol. I. Morgan Kaufmann (1983)
29. Mitchell, T.: Generalization as search. Artificial Intelligence 18, 203–226 (1982)
30. Mitchell, T.: Machine Learning. McGraw Hill (1997)
31. Muggleton, S.: Inductive logic programming. In: Arikawa, S., Goto, S., Ohsuga, S., Yokomori, T. (eds.) Proc. of the 1st Conf. on Algorithmic Learning Theory. Springer/Ohmsha (1990)
32. Nebel, B.: Reasoning and revision in hybrid representation systems, Lecture Notes in Computer Science, vol. 422. Springer (1990)
33. Nienhuys-Cheng, S., de Wolf, R.: Foundations of inductive logic programming, Lecture Notes in Artificial Intelligence, vol. 1228. Springer (1997)
34. Quinlan, J.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
35. Rouveirol, C., Ventos, V.: Towards learning in CARIN-$\mathcal{ALN}$. In: Cussens, J., Frisch, A. (eds.) Inductive Logic Programming. Lecture Notes in Artificial Intelligence, vol. 1866, pp. 191–208. Springer (2000)
36. Valiant, L.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)