A Declarative Modeling Language for Concept Learning in Description Logics

Francesca A. Lisi

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

Abstract. Learning in Description Logics (DLs) has been paid increasing attention over the last decade. Several and diverse approaches have been proposed which however share the common feature of extending and adapting previous work in Concept Learning to the novel representation framework of DLs. In this paper, we present a declarative modeling language for Concept Learning in DLs which relies on recent results in the fields of Knowledge Representation and Machine Learning. Based on second-order DLs, it allows for modeling Concept Learning problems as constructive DL reasoning tasks where the construction of the solution to the problem may be subject to optimality criteria.

1 Motivation

Despite the popularity of Machine Learning (ML) and Data Mining (DM) today, it remains challenging to develop applications and software that incorporates ML or DM techniques. This is due to the fact that research in ML and DM has focussed on developing high-performance algorithms for solving particular tasks rather than on developing general principles and techniques. De Raedt *et al.* [11,12] propose to overcome these difficulties by applying the constraint programming methodology to ML and DM and to specify ML/DM problems as Constraint Satisfaction Problems (CSPs) and Optimization Problems (OPs). The goal is to provide the user with a means for specifying declaratively what the ML/DM problem is rather than having to outline how that solution needs to be computed. This corresponds to a model + solver-based approach to ML and DM, in which the user specifies the problem in a *declarative modeling language* and the system automatically transforms such models into a format that can be used by a solver to efficiently generate a solution. This should be much easier for the user than having to implement or adapt an algorithm that computes a particular solution to a specific problem. The model + solver-based approach to ML and DM has been already investigated in De Raedt *et al.*'s work on constraint programming for itemset mining [31,17,18].

In this paper, we are interested in investigating how to model ML problems within the Knowledge Representation (KR) framework of *Description Logics* (DLs). DLs currently play a crucial role in the definition of ontology languages [1]. Learning in DLs has been paid increasing attention over the last

decade. Early work on the application of ML to DLs essentially focused on demonstrating the PAC-learnability for various terminological languages derived from the CLASSIC DL [6,8,7,16]. In particular, Cohen and Hirsh investigate the CORECLASSIC DL proving that it is not PAC-learnable [6] as well as demonstrating the PAC-learnability of its sub-languages, such as C-CLASSIC [8], through the bottom-up LCSLEARN algorithm. These approaches tend to cast supervised Concept Learning to a structural generalizing operator working on equivalent graph representations of the concept descriptions. It is also worth mentioning unsupervised Concept Learning methodologies for DL concept descriptions, whose prototypical example is KLUSTER [21], a polynomial-time algorithm for the induction of BACK terminologies. More recently, algorithms have been proposed that follow the *generalization as search* approach [29] by extending the methodological apparatus of ILP to DL languages [3,13,14,24,25]. Systems, such as YINYANG [20] and $\mathcal{DL}$-LEARNER [26], 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 [15].

In spite of the increasing interest in learning in DLs, induction is still an inference little understood within the KR community. Yet, useful ontology reasoning tasks can be based on the inductive inference. Declarative modeling would be of great help to KR practitioners in specifying ML problems. In this paper, we present a declarative modeling language for Concept Learning in DLs. Based on second-order DLs, it allows to model several variants of the Concept Learning problem in a declarative way. The design of the language is inspired by recent work on those non-standard inferences in DLs which support constructive reasoning, *i.e.* reasoning aimed at constructing a concept. Starting from the assumption that inductive inference is inherently constructive, the proposed language reformulates Concept Learning problems 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 DLs with a particular emphasys on non-standard inferences. Section 3 defines variants of the Concept Learning problem as well as variants of the solution approach for them in the DL context. Section 4 proposes a language based on second-order DLs for modeling the ML problems defined in Section 3. Section 5 concludes the paper with final remarks and directions of future work.

2 Preliminaries on Description Logics

2.1 Basics

DLs are a family of decidable First Order Logic (FOL) fragments that allow for the specification of structured knowledge in terms of classes (*concepts*), instances (*individuals*), and binary relations between instances (*roles*) [1]. Complex concepts can be defined from atomic concepts and roles by means of constructors. The syntax of some typical DL constructs is reported in Table 1. A DL knowledge base (KB) Σ consists of a so-called *terminological box* (TBox) $\mathcal{T}$ and a so-called *assertional box* (ABox) $\mathcal{A}$. The TBox is a finite set of *axioms* which represent

Table 1. Syntax and semantics of some typical DL constructs.

bottom (resp. top) concept	$\bot$ (resp. $\top$)	$\emptyset$ (resp. $\Delta^{\mathcal{I}}$)
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
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	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concept subsumption axiom	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
concept equivalence axiom	$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

either is-a relations (denoted with $\sqsubseteq$) or equivalence (denoted with $\equiv$) relations between concepts, whereas the ABox is a finite set of *assertions* (or *facts*) that represent instance-of relations between individuals (resp. couples of individuals) and concepts (resp. roles). Thus, when a DL-based ontology language is adopted, an ontology is nothing else than a TBox, and a populated ontology corresponds to a whole DL KB (*i.e.*, encompassing also an ABox).

The semantics of DLs can be defined directly with set-theoretic formalizations as shown in Table 1 or through a mapping to FOL as shown in [4]. 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 $\cdot^{\mathcal{I}}$. Under the *Unique Names Assumption* (UNA)[32], individuals are mapped to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. However UNA does not hold by default in DLs. An interpretation $\mathcal{I}$ is a *model* of a KB $\Sigma = (\mathcal{T}, \mathcal{A})$ iff it satisfies all axioms and assertions in $\mathcal{T}$ and $\mathcal{A}$. In DLs a KB represents many different interpretations, i.e. all its models. This is coherent with the *Open World Assumption* (OWA) that holds in FOL semantics. A DL KB is *satisfiable* if it has at least one model. An ABox assertion α is a *logical consequence* of a KB Σ, written $\Sigma \models \alpha$, if all models of Σ are also models of α.

The main reasoning task for a DL KB Σ is the *consistency check* which tries to prove the satisfiability of Σ. This check is performed by applying decision procedures mostly based on tableau calculus. The *subsumption check* aims at proving whether a concept is included in another one according to the subsumption relationship. Another well known reasoning service in DLs is *instance check*, i.e., the check of whether an ABox assertion is a logical consequence of a DL KB. A more sophisticated version of instance check, called *instance retrieval*, retrieves, for a DL KB Σ, all (ABox) individuals that are instances of the given (possibly complex) concept expression C, i.e., all those individuals a such that Σ entails that a is an instance of C. All these reasoning tasks support so-called

standard inferences and can be reduced to the consistency check. Besides the standard ones, additional so-called *non-standard inferences* have been investigated in DL reasoning [22]. They are treated in more detail in Section 2.2.

2.2 Non-standard reasoning

The motivation for studying non-standard reasoning in DLs has been to support the construction and maintenance of DL KBs [28]. Indeed, although standard inferences help structuring the 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 non-standard inferences are required for building and maintaining large DL KBs. Among them, the first ones to be studied have been the Least Common Subsumer (LCS) of a set concepts [5] and the Most Specific Concept (MSC) of an individual [30,23,2]. The LCS of a set of concepts is the minimal concept that subsumes all of them. The minimality condition implies that there is no other concept that subsumes all the concepts in the set and is less general than (subsumed by) the LCS. The notion of LCS is closely related to that of MSC of an individual, i.e., the least concept description that the individual is an instance of, given the assertions in the KB; the minimality condition is specified as before. More generally, one can define the MSC of a set of assertions about individuals as the LCS of MSC associated with each individual. Based on the computation of the MSC of a set of assertions about individuals one can incrementally construct a DL KB.

Very recently, Colucci *et al.* have proposed a unified framework for non-standard reasoning services in DLs [10]. The framework is based on the use of second-order sentences in DLs [9]. It applies to so-called *constructive inferences*, *i.e.* those non-standard inferences that deal with finding - or constructing - a concept. More precisely, it provides a unifying definition model for all those constructive reasoning tasks which rely on specific optimality criteria to build up the objective concept. Indeed, 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. 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. A reformulation of LCS as optimal solution problem can be found in [10].

The work on non-standard reasoning has been more or less explicitly related to ML. E.g., LCS and MCS have been used for the bottom-up induction of Classic concept descriptions from examples [8,7]. In this paper, we start from the inherently constructive nature of the inductive inference to extend Colucci *et al.*'s framework to Concept Learning in DLs.

3 Learning Concepts in Description Logics

3.1 Variants of the problem

In this section, we formally define several variants of the Concept Learning problem in a DL setting. The variants share the following features: (i) The background knowledge theory is in the form of a DL KB $\mathcal{K}$ composed of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, and (ii) the target theory in the form of concept definitions, i.e. concept equivalence axioms having an atomic concept in the left-hand side. For the purpose, we denote:

- $\mathcal{DL}$ is any DL
- $\mathsf{Ind}(\mathcal{A})$ 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 instance retrieval inference services usually available in DL systems.

The first variant of the Concept Learning problem we consider in this paper is the supervised one. It is the base for the other variants being introduced later and is denoted with Concept Induction.

Definition 1 (Concept Induction). *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 Induction (CI) *problem is to find a concept definition $C \equiv D$ with $D \in \mathcal{DL}_{\mathcal{H}}$ such that*

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 Def. 1 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 [15] testify the existence of optimality criteria to be fulfilled in CI besides the conditions of completeness and consistency. Other algorithms supporting the CI task are YinYang [20] and $\mathcal{DL}$-Learner [26].

In case a previous definition D' for C is already available in $\mathcal{K}$ which however is not correct w.r.t. the training examples then the Concept Learning problem can be cast as a Concept Refinement problem which would amount to searching for a solution D starting from the approximation D'.

Definition 2 (Concept Refinement). *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* $\mathcal{DL}_\mathcal{H}$
– *a concept* $D' \in \mathcal{DL}_\mathcal{H}$ *for which* $\exists a \in \mathsf{Ind}_C^+(\mathcal{A})$ *s.t.* $\mathcal{K} \not\models D'(a)$ *or* $\exists b \in \mathsf{Ind}_C^-(\mathcal{A})$ *s.t.* $\mathcal{K} \not\models \neg D'(b)$

the Concept Refinement (CR) *problem is to find a concept definition* $C \equiv D$ *with* $D \in \mathcal{DL}_\mathcal{H}$ *such that*

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

Note that Def. 2 differs from Def. 1 only for the further constraint (namely, compatibility with a previous concept definition) the CR problem poses on the admissible solutions.

When Concept Learning is unsupervised, the resulting problem is called Concept Formation. Tipically, this problem is decomposed in two subproblems: The former is aimed at clustering individuals in mutually disjoint concepts, whereas the latter concerns the search for a definition for each of these emerging concepts.

Definition 3 (Concept Formation). *Let* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *be a* $\mathcal{DL}$ *KB where* $\mathcal{T}$ *does not contain definitions for all the concepts with assertions in* $\mathcal{A}$*. Given a concept description language* $\mathcal{DL}_\mathcal{H}$*, the* Concept Formation (CF) *problem is to find (i) mutually disjoint concepts* C_i*, and (ii) for each* C_i*, a concept definition* $C_i \equiv D_i$ *with* $D_i \in \mathcal{DL}_\mathcal{H}$ *such that*

Completeness $\mathcal{K} \models D_i(a) \quad \forall a \in \mathsf{Ind}_{C_i}^+(\mathcal{A})$
Consistency $\mathcal{K} \models \neg D_i(b) \quad \forall b \in \mathsf{Ind}_{C_i}^-(\mathcal{A})$

Note that achieving goal (ii) in Def. 3 involves solving as many CI problems as the number of mutually disjoint concepts found in the given KB to fulfill requirement (i). Exemplars of algorithms solving the CF problem are KLUSTER [21] and CSKA [14].

3.2 Variants of the solution

In Section 3.1, 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 relation $\sqsubseteq$. Indeed, $\sqsubseteq$ is a reflexive and transitive binary relation, *i.e.* a quasi-order. Thus, $(\mathcal{DL}_\mathcal{H}, \sqsubseteq)$ is a quasi-ordered set of $\mathcal{DL}$ concept definitions which defines a search space to be traversed either top-down or bottom-up by means of suitable refinement operators according to the *generalization as search* approach in Mitchell's vision [3,13].

Definition 4 (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* $\delta : \mathcal{DL}_{\mathcal{H}} \rightarrow 2^{\mathcal{DL}_{\mathcal{H}}}$ *such that*

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

Definition 5 (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 6 (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. 6, it has been shown that the most feasible property combination for Concept Learning in expressive DLs such as $\mathcal{ALC}$ is $\{\textit{weakly complete}, \textit{complete}, \textit{proper}\}$ [24]. Only for less expressive DLs like $\mathcal{EL}$, *ideal*, i.e. complete, proper and finite, operators exist [27].

4 A Modeling Language based on Second-Order DLs

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 7 (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 variants as shown in Section 4.2.

Definition 8 (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 $X_0, \ldots, X_n$. 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 $1 \leq l \leq m$.

Definition 9 (Formula). *A* formula *Φ in $\mathcal{DL}_X$ has the form*

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

where Γ is a concept expression of the form (1) and $X_0, \ldots, X_n$ are concept variables.

We use *General Semantics*, also called Henkin semantics, for interpreting concept variables [19]. 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 10 (Satisfiability of concept expressions and formulas). *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 $\mathcal{DL}$. If Γ is satisfiable in $\mathcal{DL}$, then $\langle E_0, \ldots, E_n \rangle$ is a* solution *for Γ. A formula Φ of the form (2) is* true *in $\mathcal{DL}$ if there exist at least a solution for Γ, otherwise it is* false.

4.2 Modeling with Second-Order Concept Expressions

The second-order logic fragment introduced in Section 4.1 can be used as a declarative modeling language for Concept Learning problems in DLs. Hereafter, first, we show how to model MSC. This step is to be considered as functional to the modeling of the variants of Concept Learning considered in Section 3.1.

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 assertions they occur in and their relationship to other individuals. 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 $\sqsubseteq$. 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 $[S]$ might be empty, or contain different minimal elements, or 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 Φ_{MSC} 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 Induction 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:

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

that is, iff D can be an assignment for the concept variable X. The CI problem can be therefore modeled with the following formula Φ_{CI}:

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

which covers only the CSP version of the problem. A simple OP version of CI could be modeled by asking for solutions that are compliant with a minimality criterion involving concept subsumption checks as already done for MSC. More precisely, 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 $\mathsf{Ind}_C^-(\mathcal{A})$ iff the following formula is true in $\mathcal{DL}_{\mathcal{H}}$:

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

that is, iff there exists a concept X which is a most specific concept, and is strictly more specific than E.

Concept Refinement Concerning Def. 2, we assume that $D' \in \mathcal{DL}_{\mathcal{H}}$ is a partially correct definition for the target concept name C w.r.t. $\mathsf{Ind}^+_C(\mathcal{A}) = \{a_1, \ldots, a_m\}$ 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 C w.r.t. $\mathsf{Ind}^+_C(\mathcal{A})$ and $\mathsf{Ind}^-_C(\mathcal{A})$ iff it makes the following formula Φ_{CR}:

$$\begin{array}{c} \exists X.(C \sqsubseteq X) \wedge (X \sqsubseteq C) \wedge (X \sqsubseteq D') \wedge \\ (a_1 : X) \wedge \ldots \wedge (a_m : X) \wedge (b_1 : \neg X) \wedge \ldots \wedge (b_n : \neg X) \end{array} \tag{7}$$

true in $\mathcal{DL}_{\mathcal{H}}$. Note that Φ_{CR} is similar in the spirit to formula (6).

Concept Formation As regards Def. 3, the problem decomposition of the CF problem allows us to model the first subproblem with Φ_{MSC} and the second subproblem with a formula which substantially rely on Φ_{CI} as many times as the number of the mutually disjoint concepts found as a result of the solution to the first subproblem.

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 study reported in this paper opens 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. 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. 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., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications (2nd ed.). Cambridge University Press (2007)
2. 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: Proc. of the 18th Int. Joint Conf. on Artificial intelligence. pp. 319–324. Morgan Kaufmann Publishers (2003)
3. 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)
4. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artificial Intelligence 82(1–2), 353–367 (1996)
5. 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)
6. Cohen, W.W., Hirsh, H.: Learnability of description logics. In: Haussler, D. (ed.) Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992. ACM (1992)
7. Cohen, W.W., Hirsh, H.: The learnability of description logics with equality constraints. Machine Learning 17(2-3), 169–199 (1994)
8. Cohen, W.W., Hirsh, H.: Learning the CLASSIC description logic: Thoretical 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)
9. 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. CEUR Workshop Proceedings, vol. 573. CEUR-WS.org (2010)
10. 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)
11. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: Fox, M., Poole, D. (eds.) Proc. of the Twenty-Fourth AAAI Conference on Artificial Intelligence. AAAI Press (2010)
12. De Raedt, L., Nijssen, S., O'Sullivan, B., Van Hentenryck, P.: Constraint programming meets machine learning and data mining (dagstuhl seminar 11201). Dagstuhl Reports 1(5), 61–83 (2011)
13. 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. Lecture Notes in Computer Science, vol. 3298, pp. 441–455. Springer (2004)

14. 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)
15. 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)
16. Frazier, M., Pitt, L.: CLASSIC learning. Machine Learning 25(2-3), 151–193 (1996)
17. Guns, T., Nijssen, S., De Raedt, L.: Evaluating pattern set mining strategies in a constraint programming framework. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) Advances in Knowledge Discovery and Data Mining, Part II. Lecture Notes in Computer Science, vol. 6635, pp. 382–394. Springer (2011)
18. Guns, T., Nijssen, S., De Raedt, L.: Itemset mining: A constraint programming perspective. Artificial Intelligence 175(12-13), 1951–1983 (2011)
19. Henkin, L.: Completeness in the theory of types. Journal of Symbolic Logic 15(2), 81–91 (1950)
20. 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)
21. Kietz, J.U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning 14(1), 193–217 (1994)
22. Küsters, R.: Non-Standard Inferences in Description Logics, Lecture Notes in Computer Science, vol. 2100. Springer (2001)
23. Küsters, R., Molitor, R.: Approximating most specific concepts in description logics with existential restrictions. AI Communications 15(1), 47–59 (2002)
24. 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)
25. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning 78(1-2), 203–250 (2010)
26. Lehmann, J.: DL-Learner: Learning Concepts in Description Logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
27. 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)
28. McGuinness, D.L., Patel-Schneider, P.F.: Usability issues in knowledge representation systems. In: Mostow, J., Rich, C. (eds.) Proc. of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference. pp. 608–614. AAAI Press / The MIT Press (1998)
29. Mitchell, T.: Generalization as search. Artificial Intelligence 18, 203–226 (1982)
30. Nebel, B.: Reasoning and Revision in Hybrid Representation Systems, Lecture Notes in Computer Science, vol. 422. Springer (1990)
31. Nijssen, S., Guns, T., De Raedt, L.: Correlated itemset mining in ROC space: a constraint programming approach. In: Elder IV, J.F., Fogelman-Soulié, F., Flach, P.A., Zaki, M.J. (eds.) Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009. pp. 647–656. ACM (2009)
32. Reiter, R.: Equality and domain closure in first order databases. Journal of ACM 27, 235–249 (1980)