Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

Second-Order Description Logics: Semantics, Motivation, and a Calculus

Simona Colucci[1], Tommaso Di Noia[1], Eugenio Di Sciascio[1], Francesco M. Donini[2], Azzurra Ragone[1]

[1]: SisInfLab & DEE, Politecnico di Bari, Bari, Italy
[2]: DISCOM, Università della Tuscia, Viterbo, Italy

1 Introduction

Eight years ago, Tim Berners-Lee, James Hendler and Ora Lassila published their seminal paper [6] describing the evolution of the current web from a human-processable environment to a machine-processable one. The basic idea was to annotate web resources and give them a machine-processable meaning; the Semantic Web was born. Many efforts have been placed in the last years by the Semantic Web community in the attempt to standardize both the language for representing the content of web resources and the production of annotations/metadata. On the one hand, such efforts successfully led to the affirmation of standard languages for machine-processable representation of web pages content, like the recent W3C recommendation OWL2. On the other hand, it produced the Linked Data initiative: a set of best practices for publishing and connecting data on the Web. These two initiatives are tightly connected. In fact, data published following the Linked Data best practices are interpreted thanks to the ontological layer developed using OWL2. Despite a large effort in annotating and representing the semantic content of a resource (in a semi-automatic way) we see the lack of reasoning engines able to fully exploit such representation power. During the last years highly optimized reasoning engines have been developed for classical deductive reasoning tasks such as subsumption/classification, consistency checking and instance retrieval. At the same time, non-standard reasoning tasks have been proposed in the Description Logics literature as an answer to new issues related to knowledge-based domains especially in retrieval scenarios, ontology design and maintenance and automated negotiation. The most relevant reasoning tasks we may cite are: explanation [18], interpolation [23], concept abduction and concept contraction [10], concept unification [3], concept difference [25], concept similarity [8], concept rewriting [2], negotiation [22], least common subsumer [5], most specific concept [1] and knowledge base completion [4].

For each of the above mentioned tasks a specific algorithmic approach has been proposed and very often only for a particular (sometimes simple) Description Logic. Although the need for such reasoning tasks has been widely recognized, there is not yet a unified view—at least from an algorithmic perspective. Indeed, some of the above mentioned tasks share some properties from a computational point of view and sometimes are very related to each other. Moreover, most of the problems in the cited reasoning tasks are of the form: "Find one or more concept(s) C such that $\{$sentence involving $C\}$" and we are really interested in exhibiting such a concept, not just proving its existence. In other words, many of the above mentioned reasoning tasks, known as non-standard reasoning, deal with finding—or constructing—a concept. This is the main reason why we refer to such reasoning as *constructive reasoning*. By contrast, "standard" reasoning is about checking some property (true or false) such as subsumption or satisfiability (also query answering can be reduced to instance checking).

In this paper we propose a new second-order framework and a related calculus able to express, in a uniform way, many of the abovementioned constructive reasoning tasks.

The remainder of the paper is structured as follows: in Section 2 we introduce the framework and its formal semantics. Section 3 is devoted to the reformulation of some relevant contructive reasoning tasks in terms of second order formulas. The general calculus is presented in Section 4, before providing a section on "discussion and future directions".

2 Semantics

We denote by $\mathcal{DL}$ a generic Description Logic. Only in order to exemplify our framework, consider the presentation of the DL $\mathcal{SHIQ}$.

Let N_r be a set of role names. A *general role* R can be either a role name $P \in \mathsf{N}_r$, or its inverse, denoted by P^-. We admit a set of *role axioms*, formed by: (1) a *role hierarchy* $\mathcal{H}$, which is a set of role inclusions of the form $R_1 \sqsubseteq R_2$, and (2) a set of transitivity axioms for roles, denoted by $\mathtt{Trans}(R)$. We denote by $\sqsubseteq^*$ the transitive-reflexive closure of $\mathcal{H} \cup \{R^- \sqsubseteq S^- \mid S \sqsubseteq R \in \mathcal{H}\}$. A role S is *simple* if it is not transitive, and for no R such that $R \sqsubseteq^* S$, R is transitive.

In the following syntax for concepts, let A be a generic concept name in a set N_c of concept names.

$$C \longrightarrow \top \mid \bot \mid A \mid \geqslant n\, S.C \mid \leqslant n\, S.C \mid C_1 \sqcap C_2 \mid \neg C \tag{1}$$

We consider the other well-known constructs as abbreviations: $C_1 \sqcup C_2 = \neg(\neg(C_1) \sqcap \neg(C_2))$, $\exists R.C = \geqslant 1\, R.C$, $\forall R.C = \leqslant 0\, R.\neg C$. For computability reasons, only in $\exists R.C, \forall R.C$ the role R can be a general role (*i.e.*, also a transitive role, or a super-role of a transitive role), while in other number restrictions R must be a *simple* role.

Every DL is equipped with a model-theoretic semantics. Again, exemplifying our discussion for $\mathcal{SHIQ}$, an *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a set of individuals, and $\cdot^{\mathcal{I}}$ is an interpretation function mapping $\top$ into $\Delta^{\mathcal{I}}$, $\bot$ into $\emptyset$, each concept name $A \in \mathsf{N}_c$ into a subset of $\Delta^{\mathcal{I}}$, and each role name $P \in \mathsf{N}_r$ into a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and extended to concept and role expressions as follows (let $\sharp\{\ldots\}$ denote the cardinality of a set):

$$\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} - A^{\mathcal{I}} \tag{2}$$

$$\geqslant n\, R.C^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \in \Delta^{\mathcal{I}} \mid \langle a,b\rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \geqslant n\} \tag{3}$$

$$\leqslant n\, R.C^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \in \Delta^{\mathcal{I}} \mid \langle a,b\rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \leqslant n\} \tag{4}$$

$$(C_1 \sqcap C_2)^{\mathcal{I}} = (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} \tag{5}$$

$$(P^-)^{\mathcal{I}} = \{\langle b,a\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle a,b\rangle \in P^{\mathcal{I}}\} \tag{6}$$

As usual, we denote by $C \sqsubseteq D$ the proposition "for every interpretation $\mathcal{I}$ (satisfying role axioms), $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$". We also denote *non-subsumption* by $C \not\sqsubseteq D$, meaning the proposition "there exists an interpretation $\mathcal{I}$ satisfying role axioms such that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$". Observe that $C \sqsubseteq D$, $C \not\sqsubseteq D$ are propositions (true or false), so they can be combined by $\wedge, \vee$ in a propositional formula Γ. We say that Γ is *true* iff the composition of truth values of subsumptions and non-subsumptions yields *true*.

Second-order Concept Expressions. In order to write second-order formulas, we need a set $\mathsf{N}_x = \{X_0, X_1, X_2, \ldots\}$ of concept variables, which we can quantify over.

A *concept term* is a concept formed according to the rules in (1) plus the rule $C \longrightarrow X$ for $X \in \mathsf{N}_x$. For example, $A \sqcap X_0 \sqcap \forall (P^-).(X_1 \sqcap \exists Q.X_2)$ is a concept term. Although also role variables could be conceived, we do not need them here. We stress the fact that concept terms could be defined starting from the syntax of every Description Logic $\mathcal{DL}$, not just $\mathcal{SHIQ}$. We denote by $\mathcal{DL}_X$ the language of concept terms obtained from $\mathcal{DL}$ by adding concept variables.

We use *general semantics* [15]—also acknowledged as Henkin structures [27]—for interpreting concept variables. 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 $\Delta^{\mathcal{I}}$. Note that Baader and Narendran [3] use standard semantics in their paper on concept unification.

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 must coincide with the interpretation $E^{\mathcal{I}}$ of some concept $E \in \mathcal{DL}$. Moreover, since we are interested in particular existential second-order formulas, we limit our definition to such formulas.

Definition 1 (General Semantics). *Let $C_1, \ldots, C_m, D_1, \ldots, D_m \in \mathcal{DL}$ be concept terms containing concept variables $X_0, X_1, \ldots, X_n$, and let Γ be a conjunction of concept subsumptions and non-subsumptions, of the form*

$$\Gamma = (C_1 \sqsubseteq D_1) \wedge \cdots \wedge (C_\ell \sqsubseteq D_\ell) \wedge (C_{\ell+1} \not\sqsubseteq D_{\ell+1}) \wedge \cdots \wedge (C_m \not\sqsubseteq D_m) \qquad (7)$$

for $1 \leq \ell \leq m$. We say that Γ is satisfiable in $\mathcal{DL}$ *if and only if there exist $n+1$ concepts $E_1, \ldots, E_n \in \mathcal{DL}$ such that, extending the semantics (2)–(6) for each interpretation $\mathcal{I}$, with: $(X_i)^{\mathcal{I}} = (E_i)^{\mathcal{I}}$ for $i = 0, \ldots, n$, it holds that*

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

Otherwise, Γ is said to be unsatisfiable in $\mathcal{DL}$ *. Moreover, we say that the formula*

$$\exists X_0 \cdots \exists X_n.\Gamma \qquad (8)$$

is true in $\mathcal{DL}$ *if Γ is satisfiable in $\mathcal{DL}$, otherwise it is* false.

Note that we are considering here only a particular form of closed second-order formulas in Description Logics. This is because we are not interested here in Second-order Description Logics by themselves, but only in their use to express and compute the "constructive" reasoning services of the next section.

3 Modeling Constructive Reasoning Tasks

Hereafter we show how to model some constructive reasoning tasks in trems of formula (8). In this section we only show the constructive formulation of the task and we leave discussion on optimality criteria at the end of the section. The computation of the Most Specific Concept as well as a Knowledge Base Completion could be easily modeled if we allowed in Γ formulas involving an ABox or a TBox.

We introduce the notion of *signature of a concept* that is used in Interpolation and Concept Unification. Given a concept C we define:

$$\begin{aligned} sign(C)_{\mathsf{N}_c} &= \{A \mid A \in \mathsf{N}_c, A \text{ appears syntactically in } C\} \\ sign(C)_{\mathsf{N}_r} &= \{P \mid P \in \mathsf{N}_r, P \text{ appears syntactically in } C\} \\ sign(C) &= sign(C)_{\mathsf{N}_c} \cup sign(C)_{\mathsf{N}_r} \end{aligned}$$

Least Common Subsumer. A concept $D \in \mathcal{DL}$ is a Common Subsumer of two concepts $C_1, C_2 \in \mathcal{DL}$ if $(C_1 \sqsubseteq D) \wedge (C_2 \sqsubseteq D)$. The Least Common Subsumer (LCS) of C_1, C_2 is the least element w.r.t. $\sqsubseteq$ of the set of concepts which are Common Subsumers of C_1, C_2 and is unique up to equivalence. A concept L is *not* the Least Common Subsumer of C_1, C_2 iff the following formula (of the form (8)) is true in $\mathcal{DL}$:

$$\exists X.(C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X) \wedge (X \sqsubseteq L) \wedge (L \not\sqsubseteq X)$$

that is, L is not the LCS if there exists a concept X which is a Common Subsumer, and is strictly more specific than L. By finding a concept satisfying the above formula, and iterating the process, an algorithm for computing the LCS in a sublanguage of $\mathcal{SHIQ}$ has been proposed [12].

Interpolation. Interpolation have been proposed in Description Logics for different purposes. In [23], the computation of an interpolant is used to explain subsumption, if it exists, between two concepts C and D. Konev *et al.* [16] use the notion of interpolation for a TBox $\mathcal{T}$ in order to *forget* part of the vocabulary adopted in $\mathcal{T}$ and reason on a smaller ontology. Seylan *et al.* [24] need the computation of an interpolant between two concepts to rewrite a query in terms of DBox predicates.

Definition 2 (Interpolation). *Given two concepts C and D in $\mathcal{DL}$ such that $C \sqsubseteq D$, an interpolant of C and D is a concept I such that:*

- *$sign(I) \subseteq sign(C) \cup sign(D)$;*
- *both $C \sqsubseteq I$ and $I \sqsubseteq D$.*

Given two concepts C and D such that $C \sqsubseteq D$, the corresponding interpolant satisfies the formula $(C \sqsubseteq X) \wedge (X \sqsubseteq D)$ of the form (7).

Abduction. Abduction in Description Logics has been recognized as an interesting reasoning procedure for a set of heterogeneous tasks [10, 17, 24, 7, 21]. Here we mainly concentrate on Concept Abduction as defined in [10] and Structural Abduction [11] but the formalization can be easily extended to other abductive procedures [13]. Concept Abduction is a straight adaptation of Propositional Abduction.

Definition 3 (Concept Abduction). *Let C, D, be two concepts in $\mathcal{DL}$ where both C and D are satisfiable. A* Concept Abduction Problem *(CAP) is finding a concept $H \in \mathcal{DL}$ such that $C \sqcap H \not\sqsubseteq \bot$, and $C \sqcap H \sqsubseteq D$.*

Every solution H of a CAP satisfies the formula

$$(C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D)$$

Such solutions can be compared by $\sqsubseteq$, preferring the subsumption-maximal ones, since they are the solutions hypothesizing the least. Moreover, a formula of the form (8) can characterize the complement of being subsumption-maximal. A concept H is *not* a subsumption-maximal solution of a CAP iff the formula is true in $\mathcal{DL}$:

$$\exists X.(C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D) \wedge (H \sqsubseteq X) \wedge (X \not\sqsubseteq H)$$

In order to deal with Abduction for expressive Description Logics, a more fine grained definition of Abduction was introduced in [11] with the name of Structural Abduction. The notion of Structural Abduction relies on the notion of *Adbucible Concept* and *Hypotheses List* we report here for the sake of completeness.

Definition 4 (Abducible Concept – Hypotheses List). *Let C and D be two concepts in $\mathcal{DL}$. We define* abducible concept *$C^h \doteq H_0 \sqcap Rew(C)$, where the rewriting $Rew(C)$ is defined recursively as $Rew(A) = A$; $Rew(\neg A) = \neg A$; $Rew(C_1 \sqcap C_2) = Rew(C_1) \sqcap Rew(C_2)$;$Rew(C_1 \sqcup C_2) = Rew(C_1) \sqcup Rew(C_2)$; $Rew(\exists R.C) = \exists R.(H_{new} \sqcap Rew(C))$; $Rew(\forall R.C) = \forall R.(H_{new} \sqcap Rew(C))$ where by H_{new} we mean a concept variable not yet appearing in the rewriting. We call* hypotheses list *of C^h the list $\overline{\mathcal{H}} = \langle H_0, H_1, H_2, \ldots \rangle$.*

Definition 5 (Structural Abduction). *Let $C, D \in \mathcal{DL}$, be two concepts where both C and D are satisfiable $C \sqcap D \not\sqsubseteq \bot$. Let $\overline{\mathcal{H}} = \langle H_0, \ldots, H_\ell \rangle$ be the hypotheses list of the abducible concept C^h and $\tilde{\mathcal{A}} = \langle \mathcal{A}_0, \ldots, \mathcal{A}_\ell \rangle$ (for Assumptions) be a list of $\mathcal{DL}$ concept sets. A* Structural Abduction Problem *(SAP) for $\mathcal{DL}$ is finding a list of concepts $\mathcal{H} = \langle H_0, \ldots, H_\ell \rangle$ such that*

$$H_i \in \mathcal{A}_i \text{ for every } i = 0, \ldots, \ell \tag{9}$$

$$\mathcal{T} \not\models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq \bot \tag{10}$$

$$\mathcal{T} \models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq D \tag{11}$$

We call a SAP General *when $\mathcal{A}_i = \mathcal{DL}$, for every $i = 0, \ldots, \ell$.*

Let C^x be as C^h with X_i in place of H_i for $i = 0, \ldots, \ell$. Then, $H_0, \ldots, H_\ell$ is a solution of a SAP iff it satisfies the formula $(C^x \not\sqsubseteq \bot) \wedge (C^x \sqsubseteq D)$ by letting $(X_i)^\mathcal{I} = (H_i)^\mathcal{I}$ for every $\mathcal{I}$ and every $i = 0, \ldots, \ell$.

Concept Contraction. Gärdenfors [14] distinguishes three main kinds of belief changes: (i) *expansion*, (ii) *revision*, (iii) *contraction*. Given two concepts C and D such that $C \sqcap D \sqsubseteq \bot$, Concept Contraction is the DL-based version of contraction.

Definition 6 (Concept Contraction). *Let C, D both satisfiable. A* Concept Contraction Problem *(CCP) is finding a pair of concepts $\langle G, K \rangle$ (Give up, Keep) such that $C \equiv G \sqcap K$, and $K \sqcap D \not\sqsubseteq \bot$. We call K a* contraction *of C according to D.*

Every solution $\langle G, K\rangle$ of a CCP satisfies the formula

$$(C \equiv X_0 \sqcap X_1) \wedge (X_1 \sqcap D \not\sqsubseteq \bot)$$

Such solutions can be compared by $\sqsubseteq$, preferring the ones whose G's are subsumption-maximal, since they are the solutions contracting the least. Moreover, a formula of the form (8) can characterize the non-preferred contractions. A pair $\langle G, K\rangle$ is *not* a preferred solution of a CCP iff the following formula is true in $\mathcal{DL}$:

$$\exists X_0 \exists X_1.(C \equiv X_0 \sqcap X_1) \wedge (X_1 \sqcap D \not\sqsubseteq \bot) \wedge (G \sqsubseteq X_0) \wedge (X_0 \not\sqsubseteq G)$$

Concept Unification. Concept Unification [3] between two concepts C and D arises when one wants to rewrite some concept names occurring in C and D in order to make the relation $C \equiv D$ true.

Definition 7. *Let C and D be two concepts in $\mathcal{DL}$ such that $C \not\equiv D$. We define the two sets $\mathcal{X}^C = \{A_i^C \mid i = 1, \ldots, l\}$ and $\mathcal{X}^D = \{A_j^D \mid j = 1, \ldots, m\}$ such that $\mathcal{X}^C \subseteq sign(C)_{\mathsf{N}_c}$ and $\mathcal{X}^D \subseteq sign(D)_{\mathsf{N}_c}$. A Unification Problem is finding the set of rewriting rules $\mathcal{M}$: $A_1^C \rightarrow C_1; \ldots; A_l^C \rightarrow C_l, A_1^D \rightarrow D_1; \ldots; A_m^D \rightarrow D_m$ such that*

$$\begin{aligned} sign(C_i) &\subseteq sign(C) \cup sign(D), \textit{ with } i = 1, \ldots, l \\ sign(D_j) &\subseteq sign(C) \cup sign(D), \textit{ with } j = 1, \ldots, m \\ C &\equiv_{\mathcal{M}} D \end{aligned}$$

The Unification problem is solvable iff the following formula (of the form (8)) is true in $\mathcal{DL}$:

$$\exists A_1^C, \ldots, A_l^C, A_1^D, \ldots, A_m^D.(C \sqsubseteq D) \wedge (D \sqsubseteq C)$$

treating $\mathcal{X}^C, \mathcal{X}^D$ as concept variables interpreted in General Semantics.

Concept Difference. Following the algebraic approaches adopted in classical information retrieval, Concept Difference [25] was introduced as a way to measure concept similarity.

Definition 8. *Let C and D be two concepts such that $C \sqsubseteq D$. The Concept Difference $C - D$ is defined by $max_{\sqsubseteq}\{B \in \mathcal{DL}$ such that $D \sqcap B \equiv C\}$.*

We can use a formula of the form (8) to check whether a concept B is *not* a difference between C and D, namely, B is not a Difference iff the formula below is true:

$$\exists X.(C \sqsubseteq D \sqcap X) \wedge (D \sqcap X \sqsubseteq C) \wedge (X \sqsubseteq B) \wedge (B \not\sqsubseteq X)$$

Negotiation. The aim of a negotiation process is to find an agreement between two competitive parties. Both agreement and requirements from the two parties can be represented as (a conjunction of) concepts [22]. Usually, in a negotiation the parties requirements are in conflict with each other. Hence, in order to reach an agreement they have to give up some parts of their requirements. During a negotiation the two parties have to agree on and follow a protocol (*i.e.*, a set of rules that characterize the specific process). Here we define a simple protocol where given the initial requirements W_0^c and W_0^d, if they are in conflict with each other, then the two parties c and d propose a relaxed version $\overline{W}^c$ and $\overline{W}^d$ of W_0^c and W_0^d. The final aim of the protocol is to satisfy as much as possible both agents with the final agreement. At the first round they relax their requirements keeping the minimal information they want to be satisfied by the final agreement and propose $W_\top^c$ and $W_\top^d$ such that $W_0^c \sqsubseteq W_\top^c$ and $W_0^d \sqsubseteq W_\top^d$. For each following round i they propose least relaxed version of W_0^c and W_0^d which are more specific of the proposals at round $i-1$ [1]. The protocol stops either if the max number MAX of rounds has been reached or when it does not exist a concept both more specific than the one found at the previous round and less specific than the initial requirements.

```
Input: concepts $W_0^c, W_0^d$ such that $W_0^c \sqcap W_0^d \sqsubseteq \bot$
Output: the final outcome of the negotiation after $n$ rounds. If $c$ and $d$ reach an agreement
        the returned value is $\langle W_n^c, W_n^d \rangle$, $NULL$ otherwise.
1  begin
2      $W^c = W_\top^c$;
3      $W^d = W_\top^d$;
4      $i = 0$; $flag = continue$;
5      while $(i < MAX) \wedge (flag == continue)$ do
6          if $\exists \overline{W}^c, \overline{W}^d.(\overline{W}^c \sqcap \overline{W}^d \not\sqsubseteq \bot) \wedge (\overline{W}^c \sqsubseteq W^c) \wedge (\overline{W}^d \sqsubseteq W^d) \wedge (W_0^c \sqsubseteq$
           $\overline{W}^c) \wedge (W_0^d \sqsubseteq \overline{W}^d)$ then
7              $W^c = \overline{W}^c$ ;
8              $W^d = \overline{W}^d$ ;
9          else
10             $flag = stop$;
11         $i = i + 1$;
12     if $flag == stop$ then
13         return $\langle W^c, W^d \rangle$;
14     else
15         return $NULL$;
16 end
```

Algorithm 1: A simple negotiation protocol

Optimal Solutions. We may classify the above reasoning tasks into two main categories: tasks for which we *just* need to compute a concept (or a set of concepts) as Concept Unification and Interpolation and those for which we need to find a concept (or a set of concepts) according to some minimality/maximality criteria such as LCS, Concept Difference, Concept Abduction, Concept Contraction and Negotiation. 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. As an example, for LCS we have the set of sub-optimal solution represented by "common subsumers". Based on this observation, we may think of a procedure that computes a sub-optimal solution X^i at step i and then

[1] The way $\overline{W}^c$ and $\overline{W}^d$ are computed at each step should take into account also agents' preferences. For the sake of conciseness we omit such details.

iteratively computes a better solution X^{i+1} at the next step. The procedure stops (if decidable) when no better solution can be found according to the minimality/maximality criterion. In case the procedure is not decidable, we may decide to iterate for a maximum number of steps. In this case, the procedure returns a sub-optimal solution to the problem. In other words, this means that we may apply a procedure similar to the negotiation protocol described above to other constructive reasoning every time we need to satisfy optimal criteria.

4 A Calculus

Definition 9 (Substitutions).

1. *Let $\mathcal{DL}$ be a Description Logic, $\{i_1, \ldots, i_k\} \subseteq \{0, 1, \ldots, n\}$ be a set of distinct indexes, $X_{i_1}, \ldots, X_{i_k}$ be concept variables, and $D_{i_1}, \ldots, D_{i_k} \in \mathcal{DL}_X$ be concept terms. A* substitution *σ is a set of pairs $\{[X_{i_1}/D_{i_1}], \ldots, [X_{i_k}/D_{i_k}]\}$. A substitution is* ground *if every D_{i_j} contains no variables,* i.e., *$D_{i_j} \in \mathcal{DL}$.*
2. *For a concept term $C \in (\mathcal{SHIQ})_X$, we inductively define $\sigma(C)$ as $\sigma(X_i) = D_i$, $\sigma(\neg X_i) = \neg(\sigma(D_i))$, $\sigma(A) = A$, $\sigma(C_1 \sqcap C_2) = \sigma(C_1) \sqcap \sigma(C_2)$, $\sigma(\bowtie nR.C) = \bowtie nR.\sigma(C)$ for $\bowtie \in \{\leqslant, \geqslant\}$.*
3. *For concept terms C, D, we define also $\sigma(C \sqsubseteq D) = \sigma(C) \sqsubseteq \sigma(D)$, $\sigma(C \not\sqsubseteq D) = \sigma(C) \not\sqsubseteq \sigma(D)$, and for a boolean conjunction Γ of the form (7), $\sigma(\Gamma)$ is the result of applying σ to every subsumption and non-subsumption statement.*

By using substitutions, a formula of the form (8) is true according to Def.1 if and only if there exists a ground substitution making it valid, as formalized by the theorem below.

Theorem 1. *A formula $\exists X_0 \cdots \exists X_n.\Gamma$ is true in $\mathcal{DL}$ iff there exists a ground substitution $\sigma = \{[X_0/E_0], \ldots, [X_n/E_n]\}$ with $E_0, \ldots, E_n \in \mathcal{DL}$, such that $\sigma(\Gamma)$ is true.*

Observe that since σ is ground, and substitutes every variable in Γ, $\sigma(\Gamma)$ is just a boolean combination of [non-]subsumptions in $\mathcal{SHIQ}$. Observe also that if *standard* semantics is adopted for concept variables [3] instead of Def.1—that is, if $X^{\mathcal{I}}$ can be any subset of $\Delta^{\mathcal{I}}$—then the "only if" part of the above theorem no longer holds, since there can be statements for which $X^{\mathcal{I}}$ is not expressible in the target $\mathcal{DL}$, yielding no substitution. For example, formula $\exists X.(A \sqsubseteq X) \wedge (B \sqsubseteq X) \wedge (\top \not\sqsubseteq X)$ is false in a DL without $\sqcup$ (disjunction), but it would be true in standard semantics: just let for every $\mathcal{I}$, $X^{\mathcal{I}} = A^{\mathcal{I}} \cup B^{\mathcal{I}}$.

We present now a simple calculus, obtained by combining Analytic Tableaux for ordinary concept constructors, and substitutions for concept variables. Then we prove its soundness and completeness. Again, we present the calculus for the DL $\mathcal{SHIQ}$, but only for sake of clarity; the same framework could be adopted for other DLs. We borrow Tableaux rules (T-rules; see below) from well-known results of Tobies [26]. Since inverse roles are present in $\mathcal{SHIQ}$, we use *pairwise blocking* for individuals [26, p.125].

All rules are applicable only if x is *not blocked*. For each $i = 1, ..., n$, $\mathcal{L}_i$ is a branch in τ_i. Rules above the separating line have precedence over rules below it.

$\sqcap$-rule : **if** $C \sqcap D \in \mathcal{L}_i(x)$,
then add both C and D to $\mathcal{L}_i(x)$

$\sqcup$-rule : **if** $C \sqcup D \in \mathcal{L}_i(x)$,
then add either C or D to $\mathcal{L}_i(x)$

$\forall$-rule : **if** $\forall R.C \in \mathcal{L}_i(x)$, and there exists an individual y such that y is an R-successor of x,
then add C to $\mathcal{L}_i(y)$

$\leqslant$-rule : **if** $\leqslant n\, S.C \in \mathcal{L}_i(x)$ with $n \geqslant 1$, and
there are $m > n$ S-neighbors (say) $y_1, \ldots, y_m$ of x with $C \in \mathcal{L}_i(y_j)$ for $j = 1, \ldots, m$,
$y, z \in \{y_1, \ldots, y_m\}$ with y being an S-successor of x and not $y \neq z$
then (1) add $\mathcal{L}_i(y)$ to $\mathcal{L}_i(z)$,
(2) for every $R \in \mathcal{L}_i(x, y)$ if z is a predecessor of x then add R^- to $\mathcal{L}_i(z, x)$ else add R to $\mathcal{L}_i(x, z)$,
(3) let $\mathcal{L}_i(x, y) = \emptyset$, and
(4) for all u with $u \neq y$, set $u \neq z$

$\forall_+$-rule : **if** $\forall S.C \in \mathcal{L}_i(x)$, with $\mathtt{Trans}(R)$ and $R \sqsubseteq^* S$, there exists an individual y such that y is an R-successor of x, and $\forall R.C \notin \mathcal{L}_i(y)$,
then add $\forall R.C$ to $\mathcal{L}_i(y)$

choose-rule : **if** $\bowtie nS.D \in \mathcal{L}_i(x)$, with $\bowtie \in \{\geqslant, \leqslant\}$ and there is an S-neighbor y of x
then add either D or $\neg D$ to $\mathcal{L}_i(y)$

$\exists$-rule : **if** $\exists R.C \in \mathcal{L}_i(x)$, and x has no R-successor y with $C \in \mathcal{L}_i(y)$,
then pick up a new individual y, add R to $\mathcal{L}(x, y)$, and let $\mathcal{L}_i(y) := \{C\}$

$\geqslant$-rule : **if** $\geqslant n\, S.C \in \mathcal{L}_i(x)$, and x has not n S-neighbors $y_1, \ldots, y_n$ with $y_\ell \neq y_j$ for $1 \leqslant \ell < j \leqslant n$,
then create n new successors $y_1, \ldots, y_n$ of x with $\mathcal{L}_i(x, y_\ell) = \{S\}$, $\mathcal{L}_i(y) := \{C\}$, and $y_\ell \neq y_j$, for $1 \leqslant \ell < j \leqslant n$

A branch $\mathcal{L}$ is closed if, for some individual x, either $\bot \in \mathcal{L}(x)$, or $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some concept name A, or $\leqslant n\, S.C \in \mathcal{L}(x)$ and x has in $\mathcal{L}$ m S-neighbors $y_1, \ldots, y_m$ with $m > n$, with $C \in \mathcal{L}(y_j)$ and $y_i \neq y_j$ for $1 \leqslant i < j \leqslant m$. We call such a situation a *clash*. A tableau is closed if all its branches are closed. A branch is *open* if it is not closed, and no T-rule can be applied to it. A tableau is open if it has at least one open branch.

In order to prove a formula of the form (8), each [non-]subsumption in Γ is associated with a tableau. For a sentence $C_i \sqsubseteq D_i$, the calculus aims at closing the tableau τ_i that starts with the single branch

$$\mathcal{L}_i(a_i) = \{C_i, \neg D_i\} \tag{12}$$

with a_i being an individual. For a sentence $C_i \not\sqsubseteq D_i$, the calculus, starting with τ_i as before, aims at obtaining an open tableau. We call *system* the $n+1$-tuple $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, made of the n tableaux and the substitution on the variables. The system always starts with $\sigma = \emptyset$. Substitution rules (S-rules) are presented below. We denote the application of the substitution θ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ by $\theta\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ and its result is $\langle \theta(\tau_1), \ldots, \theta(\tau_n), \theta \cup \sigma \rangle$.

All rules are applicable only if $\mathcal{L}$ is *open*, and the substitution is *not σ-blocked*. Rules above the separating line have precedence over rules below it.

$\sigma\top$**-rule** : apply $[X/\top]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$
σN**-rule** : apply $[X/A]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$

$\sigma\neg$**-rule** : apply $[X/\neg Y]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y denotes a concept variable not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$
$\sigma\geqslant$**-rule** : apply $[X/\geqslant m\, R.Y]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y denotes a concept variable not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, and if $m > 1$ then R is a simple role
$\sigma\leqslant$**-rule** : apply $[X/\leqslant n\, R.Y]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y denotes a concept variable not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, and if $n > 0$ then R is a simple role

---

$\sigma\sqcap$**-rule** : apply $[X/Y_1 \sqcap Y_2]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y_1, Y_2 denote concept variables not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$

Note that T-rules are applied separately to each branch of each tableau, while S-rules are applied to all branches of all tableaux at the same time.

An S-rule r is *σ-blocked* for $X \in \mathcal{L}_i(x)$ in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ if $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ derives from some $\langle \tau'_1, \ldots, \tau'_m, \sigma' \rangle$, in which there is some individual x' such that: (i) $X' \in \mathcal{L}'_i(x')$, (ii) $\mathcal{L}_i(x) = \mathcal{L}'_i(x')$, (iii) for every R-successor y of x in $\mathcal{L}_i$, there exists an R-successor y' of x' in $\mathcal{L}'_i$ such that $\mathcal{L}_i(y) = \mathcal{L}'_i(y')$, (iv) for every S, the number of different S-neighbors of x in $\mathcal{L}_i$ is the same as the number of different S-neighbors of x' in $\mathcal{L}'_i$, and (v) Rule r has been applied to $\mathcal{L}'_i$ in $\langle \tau'_1, \ldots, \tau'_m, \sigma' \rangle$.

Theorem 2 (Soundness). *Let Γ be as in (7). If the calculus of T- and S-rules, starting with τ_i as in (12) and $\sigma = \emptyset$, yields a system $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ in which τ_i is closed for $i = 1, \ldots, \ell$, and τ_j is open for $j = \ell+1, \ldots, m$, then there exists a substitution σ' such that $\sigma'(\Gamma)$ is true.*

Proof. Let σ' be σ in which every remaining unsubstituted concept variable is substituted with a different concept name A never appearing in Γ. Since T-rules are sound, each closed tableau τ_i for $i = 1, \ldots, \ell$ is a proof that $\sigma(C_i) \sqsubseteq \sigma(D_i)$, and the same is also a proof for $\sigma'(C_i) \sqsubseteq \sigma'(D_i)$. Moreover, since T-rules are complete, each open tableau τ_j for $j = \ell+1, \ldots, m$ is a proof that $\sigma(C_j) \not\sqsubseteq \sigma(D_j)$, and the same remains a proof for $\sigma'(C_j) \not\sqsubseteq \sigma'(D_j)$, since remaining variables are substituted by unconstrained concept names. □

Theorem 3 (Completeness). *Let Γ be as in (7). If there exists a substitution σ such that $\sigma(\Gamma)$ is true, then there is a way of applying T- and S-rules that yields a system $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ in which τ_i is closed for $i = 1, \ldots, \ell$, and τ_j is open for $j = \ell + 1, \ldots, m$.*

Proof. Since S-rules mimic $\mathcal{SHIQ}$ syntax (1), every ground substitution σ can be reconstructed by repeated applications of S-rules. If one decides to apply all these S-rules at once, one gets a system $\langle \tau_1', \ldots, \tau_m', \sigma' \rangle$ in which each τ_i has one branch $\mathcal{L}_i(a_i) = \{\sigma(C_i), \sigma(\neg D_i)\}$, and $\sigma' = \sigma$. Now since T-rules are sound and complete, their application yields closed tableaux τ_i for $i = 1, \ldots, \ell$, and open tableaux τ_j for $j = \ell + 1, \ldots, m$. $\square$

Soundness and completeness of the above calculus, together with undecidability results for specific problems such as unification in $\mathcal{SHI}$ [28], imply that there are infinite instances in which the calculus does not terminate. However, for specific classes of formulas of the form (8), a termination proof can be devised on the basis of *σ-blocking* [12], which prevents the application of S-rules.

5 Discussion and Future Work

Some related work [3, 12] has been already compared within the technical sections of the paper. In addition, some researchers proposed and studied the use of Higher-order DLs for meta-modeling purposes. More specifically, Pan& Horrocks [20] propose a stratified Higher-order DL (OWL FA) to cope with meta-assertions about concepts and roles; OWL FA is incomparable with any $\mathcal{DL}_X$, since on one side, higher-order assertions can be made, but on the other side, concept variables are not admitted. Motik [19] proves that satisfiability in Higer-order $\mathcal{ALCO}$, which is a fragment of OWL Full, is undecidable; his proof could not be rephrased in $(\mathcal{SHIQ})_X$, since it exploits the feature $\mathcal{O}$ to construct concepts starting from individuals. Motik also proposes a Higher-order DL with two possible semantics, but again, he does not consider concept variables. De Giacomo *et al.* [9] augment a DL with variables that may be interpreted—in a Henkin semantics—as individuals, concepts, and roles at the same time, obtaining a new logic $Hi(\mathcal{DL})$. Also this extension is incomparable with any $\mathcal{DL}_X$, since on one side one can express in $Hi(\mathcal{DL})$ arbitrarily higher-order concepts that are not expressible in $\mathcal{DL}_X$, while in $\mathcal{DL}_X$ one can form complex concept terms that are not allowed in $Hi(\mathcal{DL})$, such as $\exists R.X$.

The innovative potential of the paper mainly lays in the investigation on non-standard reasoning services apparently far from each other under a unifying lens. Such a unification effort paves the way to important generalization results both in the definition and in the solution of problems expressible according to the proposed framework. In particular, on the one hand we exploit the property that many non-standard reasoning services are devoted to the "construction of an objective concept" in order to model all of them as Constructive Reasoning Tasks trough special Second-order sentences in DLs; on the other hand we propose a unified calculus aimed at the design and implementation of a unique system able to solve any non-standard reasoning tasks, whose investigation is object of our current and future research work.

References

1. F. Baader, ‘Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles’, in *IJCAI 2003*.
2. F. Baader, R. Küsters, and R. Molitor, ‘Rewriting concepts using terminologies’, in *KR 2000*.
3. F. Baader and P. Narendran, ‘Unification of concept terms in description logics’, *J. of Symbolic Computation*, **31**, 277–305, (2001).
4. F. Baader and B. Sertkaya, ‘Usability issues in description logic knowledge base completion’, in *ICFCA-2009*.
5. F. Baader, B. Sertkaya, and A.-Y. Turhan, ‘Computing the least common subsumer w.r.t. a background terminology’, *J. of Applied Logic*, **5**(3), 392–420, (2007).
6. T. Berners-Lee, J. Hendler, and O. Lassila, ‘The semantic web’, *Scient. Amer.*, (2001).
7. M. Bienvenu, ‘Complexity of abduction in the EL family of lightweight description logics’, in *KR 2008*.
8. A. Borgida, T. Walsh, and H. Hirsh, ‘Towards measuring similarity in description logics’, in *DL 2005*.
9. G. De Giacomo, M. Lenzerini, and R. Rosati, ‘On higher-order description logics’, in *DL 2009*. CEUR-WS.org.
10. T. Di Noia, E. Di Sciascio, and F. M. Donini, ‘Semantic matchmaking as non-monotonic reasoning: A description logic approach’, *J. of Artif. Intell. Res.*, **29**, 269–307, (2007).
11. T. Di Noia, E. Di Sciascio, and F. M. Donini, ‘Computing information minimal match explanations for logic-based matchmaking’, in *WI/IAT 2009*.
12. F. M. Donini, S. Colucci, T. Di Noia, and E. Di Sciascio, ‘A tableaux-based method for computing least common subsumers for expressive description logics’, in *IJCAI 2009*.
13. C. Elsenbroich, O. Kutz, and U. Sattler, ‘A case for abductive reasoning over ontologies’, in *OWLED Workshop*.
14. P. Gardenfors, *Knowledge in Flux*, Mit Press, Bradford Book, 1988.
15. L. Henkin, ‘Completeness in the theory of types’, *J. of Symbolic Logic*, **15**(2), 81–91, (1950).
16. B. Konev, D. Walther, and F. Wolter, ‘Forgetting and uniform interpolation in large-scale description logic terminologies’, in *IJCAI 2009*.
17. F. Lécué, A. Delteil, and A. Léger, ‘Applying abduction in semantic web service composition’, in *ICWS 2007*.
18. D. L. McGuinness and A. Borgida, ‘Explaining subsumption in description logics’, in *IJCAI’95*.
19. B. Motik, ‘On the properties of metamodeling in OWL’, *J. of Log. and Comp.*, **17**(4), 617–637, (2007).
20. J. Z. Pan and I. Horrocks, ‘Owl fa: a metamodeling extension of OWL DL’, in *WWW’06*. ACM.
21. I. S. E. Peraldi, A. Kaya, and R. Möller, ‘Formalizing multimedia interpretation based on abduction over description logic aboxes’, in *DL 2009*.
22. A. Ragone, ‘OWL-DL as a power tool to model negotiation mechanisms with incomplete information’, in *ISWC/ASWC 2007*.
23. S. Schlobach, ‘Explaining subsumption by optimal interpolation’, in *JELIA 2004*.
24. I. Seylan, E. Franconi, and J. de Bruijn, ‘Effective query rewriting with ontologies over dboxes’, in *IJCAI 2009*.
25. G. Teege, ‘Making the difference: A subtraction operation for description logics’, in *KR’94*.
26. S. Tobies, *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*, Ph.D. dissertation, RWTH Aachen, 2001.
27. J. Väänänen, ‘Second-order logic and foundations of mathematics’, *The Bulletin of Symbolic Logic*, **7**(4), 504–520, (2001).
28. F. Wolter and M. Zakharyaschev, ‘Undecidability of the unication and admissibility problems for modal and description logics’, *ACM Trans. on Computational Logic*, **9**, (2008).