ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010
© 2010 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-60750-606-5-479

A Unified Framework for Non-standard Reasoning Services in Description Logics

Simona Colucci[1] and **Tommaso Di Noia**[2] and **Eugenio Di Sciascio**[3] and **Francesco M. Donini**[4] and **Azzurra Ragone**[5]

Abstract. Non-standard reasoning in Description Logics (DLs) comprises computing a Least Common Subsumer (LCS), a Concept Difference, a Concept Unifier, or an Interpolant Concept, to name a few. Although some reasoning services have been unified already (*e.g.*, LCS and Most Specific Concept), the definition of non-standard problems and the computation that solve them are very different from each other. We propose to unify the definitions of non-standard services as special Second-order sentences in DLs; when the solution concepts are optimal with respect to some preferences, a fixpoint replaces the Second-order quantification. Moreover, we propose to combine the well-known Tableaux calculi for DLs with rules that compute substitutions of Concept Variables. We prove soundness and completeness of the combined calculus and we give a sufficient condition for termination, which covers some non-trivial cases.

1 Introduction

During the last years, several 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 DLs literature as an answer to new issues related to knowledge-based domains, especially in retrieval scenarios, ontology design and maintenance and automated negotiation. Relevant reasoning tasks we may cite are: explanation [19], interpolation [24], concept abduction [11], concept contraction [10], concept unification [3], concept difference [26], concept similarity [7], 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) DL. 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 have 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; and even 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 above mentioned constructive reasoning tasks. In particular we detail as fixed-point reasoning tasks those constructive reasoning tasks that rely on specific optimality criteria to build up the objective concept. In addition to the theoretical result of unifying several reasoning services, the practical importance of this unification is that it paves the way to the construction of *one* system that can solve, with slight adaptations, all the above mentioned non-standard 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 constructive reasoning tasks in terms of second order formulas. In Section 4 we specify such a reformulation for fixed-point reasoning tasks. The general calculus is presented in Section 5, before the conclusive section.

2 Semantics

We denote by $\mathcal{DL}$ a generic DL. 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, $\texttt{Trans}(R)$. 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.

Let N_c be a set of concept names, and let let $A \in \mathsf{N}_c$. Generic concepts C can be described by the following syntax:

$$C \quad \longrightarrow \quad \top \mid \bot \mid A \mid \geqslant n\, S.C \mid \leqslant n\, S.C \mid C_1 \sqcap C_2 \mid \neg C \quad (1)$$

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 [16], 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 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

[1] Politecnico di Bari, Italy, email: s.colucci@poliba.it
[2] Politecnico di Bari, Italy, email: t.dinoia@poliba.it
[3] Politecnico di Bari, Italy, email: disciascio@poliba.it
[4] Università della Tuscia , Viterbo, Italy, email: donini@unitus.it
[5] Politecnico di Bari, Italy, email: a.ragone@poliba.it

$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and extended to concept and role expressions as follows (let $\sharp\{\ldots\}$ denote the cardinality of a set):

$$\begin{array}{rcll} \neg C^{\mathcal{I}} & = & \Delta^{\mathcal{I}} - A^{\mathcal{I}} & (2) \\ \geqslant n\, R.C^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \in \Delta^{\mathcal{I}} \mid \langle a,b\rangle \in R^{\mathcal{I}}, b \in C^{\mathcal{I}}\} \geqslant n\} & \\ \leqslant n\, R.C^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \in \Delta^{\mathcal{I}} \mid \langle a,b\rangle \in R^{\mathcal{I}}, b \in C^{\mathcal{I}}\} \leqslant n\} & \\ (C_1 \sqcap C_2)^{\mathcal{I}} & = & (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} & \\ (P^-)^{\mathcal{I}} & = & \{\langle b,a\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle a,b\rangle \in P^{\mathcal{I}}\} & (3) \end{array}$$

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. For instance, *strict subsumption* between C and D ($C \sqsubset D$) is expressed by the formula $(C \sqsubseteq D) \wedge (D \not\sqsubseteq C)$. We say Γ is *true* iff the composition of truth values of subsumptions and non-subsumptions yields *true*.

2.1 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. We stress the fact that concept terms could be defined starting from the syntax of every DL $\mathcal{DL}$, 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 known as Henkin structures [28]—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*

$$(C_1 \sqsubseteq D_1) \wedge \cdots \wedge (C_\ell \sqsubseteq D_\ell) \wedge (C_{\ell+1} \not\sqsubseteq D_{\ell+1}) \wedge \cdots \wedge (C_m \not\sqsubseteq D_m) \quad (4)$$

for $1 \leq \ell \leq m$. We say that Γ is satisfiable in $\mathcal{DL}$ *iff there exist a set of $n+1$ $\mathcal{DL}$ concepts $\mathcal{E} = \langle E_0, \ldots, E_n\rangle$ such that, extending the semantics (2)–(3) for each interpretation $\mathcal{I}$, with: $(X_i)^{\mathcal{I}} = (E_i)^{\mathcal{I}}$ for $i = 0, \ldots, n$, it holds that both*

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

Otherwise, Γ is said to be unsatisfiable in $\mathcal{DL}$ *. If Γ is satisfiable in $\mathcal{DL}$ then we call $\mathcal{E}$ a **solution** for Γ and we define :*

$$SOL(\Gamma) = \{\mathcal{E} = \langle E_0, \ldots, E_n\rangle \mid \mathcal{E} \text{ is a solution for } \Gamma\}$$

as the set of solutions for Γ. Moreover, we say that the formula

$$\exists X_0 \cdots \exists X_n.\Gamma \quad (5)$$

is false in $\mathcal{DL}$ *if $SOL(\Gamma) = \emptyset$* (i.e., *Γ is unsatisfiable), else it is* true.

From now on, when Γ contains only one concept variable, so that solution tuples $\mathcal{E}$ amount to a single concept, we write $\mathcal{E} = E_0$ instead of $\mathcal{E} = \langle E_0\rangle$ to improve readability.

Note that we consider only a particular form of closed second-order formulas in DLs. because we are not interested in Second-order DLs by themselves, but only in their use to express and compute the "constructive" reasoning services presented in next Section.

3 Constructive Reasoning

Hereafter we show how to model some constructive reasoning tasks as second-order concept expressions. We introduce the notion of *signature of a concept* that is used in the following Constructive Reasoning tasks. Given a concept C we define:

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

3.1 Interpolation

Interpolation has been proposed in DLs for different purposes. Given two concepts C and D such that $C \sqsubseteq D$ holds, Schlobach [24] uses an interpolant to explain such a subsumption. Konev *et al.* [17] 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.* [25] 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:*

- *$sign(I) \subseteq sign(C) \cap sign(D)$;*
- *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 (4), when X is interpreted, in General Semantics, in the DL which is the restriction of $\mathcal{DL}$ to $sign(C) \cup sign(D)$.

3.2 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 3 *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 $\mathcal{M}$: $A_1^C \to C_1; \ldots; A_l^C \to C_l, A_1^D \to D_1; \ldots; A_m^D \to D_m$ such that*

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

The Unification problem is solvable iff the following formula (of the form (5)) 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 which are interpreted in the DL which is the restriction of $\mathcal{DL}$ to $sign(C) \cup sign(D)$.

4 Optimality in Constructive Reasoning

In many non-standard reasoning tasks, $SOL(\Gamma)$ has a preference relation $\prec$ between solutions. As an example, we know that a concept $D \in \mathcal{DL}$ is a Common Subsumer of two concepts $C_1, C_2 \in \mathcal{DL}$ if $(C_1 \sqsubseteq D)$ and $(C_2 \sqsubseteq D)$, or, recalling Def.1, if D is a solution of $\Gamma_{LCS} = (C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X)$. The LCS of C_1, C_2 is the least element w.r.t. $\sqsubset$ of $SOL(\Gamma_{LCS})$ and is unique up to equivalence [9]. We generalize this idea as follows.

Definition 4 (OSP) *An Optimal Solution Problem (OSP)* ***P*** *is a pair* $\langle \Gamma, \prec \rangle$*, where* Γ *is a formula of the form (4) and* $\prec$ *is a preorder over* $SOL(\Gamma)$*. A solution to* ***P*** *is a concept tuple* $\mathcal{E}$ *such that both* $\mathcal{E} \in SOL(\Gamma)$ *and there is no other* $\mathcal{E}' \in SOL(\Gamma)$ *with* $\mathcal{E}' \prec \mathcal{E}$*.*

To clarify the above definition, consider the LCS problem where $C_1 = A \sqcap \exists R.\top$ and $C_2 = B \sqcap \exists R.\forall R^-.A$, for the simple DL $\mathcal{FL}^-\mathcal{EI}$. Solutions to $\Gamma = (A \sqcap \exists R.\top \sqsubseteq X) \wedge (B \sqcap \exists R.\forall R^-.A \sqsubseteq X)$ are $SOL(\Gamma) = \{\top, A, \exists R.\top, A \sqcap \exists R.\top\}$, up to equivalence. The preference relation for LCS is obviously $\sqsubset$ (strict subsumption).

4.1 Non-standard services in DLs as OSPs

We now show how the above framework can capture five non-standard reasoning tasks, going from the most renowned to the fairly recent—and less well-known—ones. Aiming at a fixpoint computation for solving each of the problems below, we also point out a greatest element (*i.e.*, a least preferred one) w.r.t. $\prec$ which could be used to start the iteration of an inflationary operator.

4.1.1 Least Common Subsumer

Common subsumers of C_1, C_2 satisfy the formula of the form (4) $\Gamma_{LCS} = (C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X)$. Then, the LCS problem can be expressed by the OSP $LCS = \langle (C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X), \sqsubset \rangle$. We note that $\top$ is always a solution which is a greatest element w.r.t. $\sqsubset$.

Such a formulation of LCS was already proposed by Donini *et al.* [12], in a less general way, for a sub-language of $\mathcal{SHIQ}$. That formalization becomes a special case of the one we propose here, since other non-standard reasoning tasks are not considered in that paper.

4.1.2 Concept Difference

Following the algebraic approaches adopted in classical information retrieval, Concept Difference [26] was introduced as a way to measure concept similarity.

Definition 5 *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 define the following formula of the form (4):

$$\Gamma_{DIFF} = (C \sqsubseteq D \sqcap X) \wedge (D \sqcap X \sqsubseteq C)$$

Such a definition causes Concept Difference to be modeled as the OSP $DIFF = \langle \Gamma_{DIFF}, \sqsupset \rangle$. We recall that, is spite of its name, a Concept Difference problem may have several solutions [26]. Note that a greatest solution for Γ_{DIFF} w.r.t. $\sqsupset$ is C itself.

4.1.3 Concept Abduction

Originally introduced in [11], Abduction in DLs has been recognized as an interesting reasoning service for a set of varied tasks [6, 10, 18, 20, 25]. Here we focus on Concept Abduction [11] but the formalization can be easily extended to other abductive procedures [13]. Concept Abduction is a straight adaptation of Propositional Abduction.

Definition 6 (Concept Abduction) *Let* C*,* D*, be two concepts in* $\mathcal{DL}$*, both* C *and* D *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

$$\Gamma_{ABD} = (C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D)$$

The preference relation for evaluating solutions is subsumption-maximality, since less specific solutions should be preferred because they hypothesize the least. According to the proposed framework, we can model Subsumption-maximal Concept Abduction as $ABD = \langle (C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D), \sqsupset \rangle$. Note that a greatest solution of ABD w.r.t. $\sqsupset$ is D, if $C \sqcap D$ is a satisfiable concept (if it is not, then ABD has no solution at all [11, Prop.1]).

4.1.4 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 7 (Concept Contraction) *Let* C*,* D *be two satisfiable concepts in* $\mathcal{DL}$*. 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 of the form (4)

$$\Gamma_{CONTR} = (C \sqsubseteq X_0 \sqcap X_1) \wedge (X_0 \sqcap X_1 \sqsubseteq C) \wedge (X_1 \sqcap D \not\sqsubseteq \bot)$$

Following an information minimal criterion [14], such solutions can be compared by $\sqsubseteq$, preferring the ones whose Ks are subsumption-minimal, since they are the solutions keeping the most. As a consequence, we can define Subsumption-minimal Concept Contraction as $CONTR = \langle \Gamma_{CONTR}, \sqsubset_2 \rangle$, where $\sqsubset_2$ compares by $\sqsubset$ only the second element of each pair of solution concepts. Note that a greatest solution of Γ_{CONTR} w.r.t. $\sqsubset$ is $\langle C, \top \rangle$, that is the solution which gives up the most (the whole C is contracted to $\top$).

4.1.5 DL-based Negotiation

The main aim of a negotiation process is to find an agreement between two competing parties. Both agreement and requirements from the two parties can be represented as (a conjunction of) concepts [22]. In general, when the negotiation starts, the parties requirements are in conflict with each other. Hence, in order to reach an agreement they have to relax their original requirements, until they reach an agreement. However, in every negotiation scenario each party has also some *strict requirements*, which cannot be negotiated and must be enforced in the final agreement. Consider two competing agents c and d, whose strict requirements are, respectively, S_c and S_d, with $S_c \sqcap S_d \not\sqsubseteq \bot$ (otherwise no agreement can be reached). Every agreement A should enforce strict requirements, while being always feasible, *i.e.*, it must be a solution of

$$\Gamma_{NEG} = (X \sqsubseteq S_c) \wedge (X \sqsubseteq S_d) \wedge (X \not\sqsubseteq \bot)$$

Moreover, we are usually interested in those agreements satisfying some economical properties, such as Pareto optimality[6]. Then, we may define a preference relation between concepts in $SOL(\Gamma_{NEG})$ $\prec_{NEG}$ using the notion of utility function. Given two computable utility functions $u_c, u_d : \mathcal{DL} \longrightarrow [0,1]$ let

$$A \prec_{NEG} A' \quad \text{iff} \quad u_c(A) \cdot u_d(A) > u_c(A') \cdot u_d(A')$$

It can be shown that a least concept w.r.t. $\prec_{NEG}$ is a Pareto optimal agreement, so we can define $NEG = \langle \Gamma_{NEG}, \prec_{NEG} \rangle$. Usually [22, 23] the utility functions are non-increasing over subsumption, *i.e.*, if $A_1 \sqsubset A_2$—meaning that agreement A_1 decides strictly more features than A_2—then $u(A_1) \geq u(A_2)$, where equality means that the features additionally decided by A_1 are irrelevant for the agent. Note that a greatest solution of Γ_{NEG} w.r.t. $\prec_{NEG}$ is $S_c \sqcap S_d$, since it barely enforces strict requirements of both parties without deciding any of the features that would increase $u_c \cdot u_d$.

4.2 Optimality by fixpoint

Optimal solutions w.r.t. a preorder might be reached by iterating an inflationary operator. We now specialize the definition of inflationary operators and fixpoints to our setting.

Definition 8 (Inflationary operators and fixpoints) *Given an OSP $\mathbf{P} = \langle \Gamma, \prec \rangle$, we say that the operator $b_{\mathbf{P}} : SOL(\Gamma) \to SOL(\Gamma)$ (for **b**etter) is* inflationary *if for every $\mathcal{E} \in SOL(\Gamma)$, it holds that $b_{\mathbf{P}}(\mathcal{E}) \prec \mathcal{E}$ if $\mathcal{E}$ is not a least element of $\prec$, $b_{\mathbf{P}}(\mathcal{E}) = \mathcal{E}$ otherwise. In the latter case, we say that $\mathcal{E}$ is a fixpoint of $b_{\mathbf{P}}$.*

Intuitively, $b_{\mathbf{P}}(\mathcal{E})$ is a solution better than $\mathcal{E}$ w.r.t. $\prec$, if such a solution exists, otherwise a fixpoint has been reached, and such a fixpoint is a solution to $\mathbf{P}$. Being $b_{\mathbf{P}}$ inflationary, a fixpoint is always reached by the following induction: starting from a solution $\mathcal{E}$, let

$$\begin{aligned} \mathcal{E}_0 &= \mathcal{E} \\ \mathcal{E}_{i+1} &= b_{\mathbf{P}}(\mathcal{E}_i) \text{ for } i = 0, 1, 2, \dots \end{aligned}$$

Then, there exists a limit ordinal λ such that $\mathcal{E}_\lambda$ is a fixpoint of $b_{\mathbf{P}}$. For each of the previous five non-standard reasoning services, we highlighted a greatest solution $\mathcal{E} \in SOL(\Gamma)$ which this iteration can start from. Obviously, when $\prec$ is well-founded (in particular, when $SOL(\Gamma)$ is finite) the fixpoint is reached in a finite number of steps. However, also when after n iterations $\mathcal{E}_n$ is not a fixpoint, $\mathcal{E}_n$ can be considered as an approximation of an optimal solution, since $\mathcal{E}_{i+1} \prec \mathcal{E}_i$ for every $i = 0, \dots, n$.

5 Defining $b_{\mathbf{P}}$ through a Calculus

In this section we first set up a calculus that computes a solution of formulas Γ of the form (4), and prove its soundness and completeness. We do not attempt to prove termination, since some of the above problems are known to be undecidable, *e.g.*, unification in $\mathcal{SHI}$ [29]. Then we show for each of the previous OSPs how the preference relation can be embedded inside a formula Γ' extending Γ. Hence, an implementation of such a calculus would provide a uniform method for solving all the above non-standard reasoning problems.

We stress the fact that well-known decidability results of Büchi [8] and Rabin [21] about Monadic Second-order Logic do not apply in our case, since they refer to Standard Semantics.

[6] An agreement is Pareto-optimal if no agent can improve its utility without worsening the other agent's utility.

Definition 9 (Substitutions) *Let $\{i_1, \dots, i_k\} \subseteq \{0, 1, \dots, n\}$ be a set of distinct indexes, $X_{i_1}, \dots, X_{i_k}$ be concept variables, and $D_{i_1}, \dots, D_{i_k} \in (\mathcal{SHIQ})_X$ be concept terms.*

1. *A* substitution *σ is a set of pairs $\{[X_{i_1}/D_{i_1}], \dots, [X_{i_k}/D_{i_k}]\}$. A substitution is* ground *if every D_{i_j} contains no variables,* i.e., *$D_{i_j} \in \mathcal{SHIQ}$.*
2. *Let $C \in (\mathcal{SHIQ})_X$ be a concept term, we 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 (4), $\sigma(\Gamma)$ is the result of applying σ to every subsumption and non-subsumption statement.*

By using substitutions, a formula of the form (5) is true according to Def.1 if and only if there exists a ground substitution making Γ true, as formalized by the theorem below.

Theorem 1 *A formula $\exists X_0 \cdots \exists X_n.\Gamma$ is true in $\mathcal{SHIQ}$ iff there exists a ground substitution $\sigma = \{[X_0/E_0], \dots, [X_n/E_n]\}$ with $E_0, \dots, E_n \in \mathcal{SHIQ}$, 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 [27]. Since inverse roles are present in $\mathcal{SHIQ}$, we use *pairwise blocking* for individuals [27, p.125].

TABLEAUX RULES (T-rules)

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) $y_1, \dots, y_m$ of x with $C \in \mathcal{L}_i(y_j)$ for $j = 1, \dots, m$,
$y, z \in \{y_1, \dots, y_m\}$ 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 (5), 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

$$\mathcal{L}_i(a_i) = \{C_i, \neg D_i\} \tag{6}$$

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 $\langle \theta(\tau_1), \ldots, \theta(\tau_n), \theta \cup \sigma\rangle$.

SUBSTITUTION RULES (S-rules)

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 $\langle \tau_1', \ldots, \tau_m', \sigma'\rangle$.

It is well-known [27] that T-rules are sound and complete, *i.e.*, $C \sqsubseteq D$ is true if and only if the tableau of the form (6) closes. We now extend this property to our combined calculus.

Theorem 2 (Soundness) *Let Γ be as in (4). If the calculus of T- and S-rules, starting with each τ_i as in (6) and $\sigma = \emptyset$, yields a system $\langle \tau_1, \ldots, \tau_m, \sigma\rangle$ in which each τ_i is closed for $i = 1, \ldots, \ell$, and each 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 (4). 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 each τ_i is closed for $i = 1, \ldots, \ell$, and each 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 $j = \ell+1, \ldots, m$. □

Soundness and completeness of the above calculus, together with undecidability results for specific problems such as unification in $\mathcal{SHI}$ [29], imply that (i) there are infinitely many instances in which the calculus does not terminate, (ii) there is no algorithm completely identifying [non-]terminating cases. However, for specific classes of formulas of the form (5), a termination proof can be devised on the basis of *σ-blocking* [12], which prevents the application of S-rules.

We now show how to define $b_{\mathbf{P}}$ when $\mathbf{P} = LCS, DIFF, ABD, CONTR, NEG$. The idea is to add to $\Gamma_{\mathbf{P}}$ the conditions enforcing a better solution, yielding a new formula $\Gamma_{\mathbf{P}}'$. In all cases, $b_{\mathbf{P}}$ has the following form:

$$b_{\mathbf{P}}(\mathcal{E}) = \begin{cases} \mathcal{E}' \in SOL(\Gamma_{\Gamma_{\mathbf{P}}'}) \text{ if such an } \mathcal{E}' \text{ exists} \\ \mathcal{E} \text{ otherwise} \end{cases}$$

and the conditions added to $\Gamma_{\mathbf{P}}$ enforce that $\mathcal{E}' \prec_{\mathbf{P}} \mathcal{E}$. To shorten formulas, we use the abbreviation $C \equiv D$ for $(C \sqsubseteq D) \wedge (D \sqsubseteq C)$, and $C \sqsubset D$ to mean $(C \sqsubseteq D) \wedge (D \not\sqsubseteq C)$. In all the problems below, but for Concept Contraction, $\mathcal{E}$ is just the singleton E.

$$\begin{aligned}
\Gamma'_{LCS} &= (C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X) \wedge (X \sqsubset E) \\
\Gamma'_{DIFF} &= (D \sqcap X \equiv C) \wedge (E \sqsubset X) \\
\Gamma'_{ABD} &= (C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D) \wedge (E \sqsubset X) \\
\Gamma'_{CONTR} &= C \equiv (X_0 \sqcap X_1)) \wedge (X_1 \sqcap D \not\sqsubseteq \bot) \wedge (X_1 \sqsubset E_1) \\
&\quad (\text{given } \mathcal{E} = \langle E_0, E_1 \rangle)
\end{aligned}$$

For DL-based Negotiation, we exploit the property that utilities are non-increasing over strict subsumption, *i.e.*, $E' \sqsubset E$ implies that $u_x(E') \geq u_x(E)$, for $x = c, d$.

$$\Gamma'_{NEG} = (X \sqsubseteq S_c) \wedge (X \sqsubseteq S_d) \wedge (X \sqsubset E)$$

In order to avoid adding insignificant details to E' (that would imply $u(E') = u(E)$, with no real improvement towards an optimum), we should delve into the details of u. For instance, if each agent assigns a worth $w_1, \ldots, w_n$ to some concepts $P_1, \ldots, P_n$, and u is defined by $u(E) = \sum_{E \sqsubseteq P_i} w(P_i)$ [22], then it is sufficient to add to the above formula the conjunct $X \sqsubseteq (\sqcup_{E \not\sqsubseteq P_i} P_i)$, expressing that X should subsume at least one preferred concept P_i not yet subsumed by E.

Observe that the number of occurrences of variables in each of the above formulas Γ is a small constant (6 in the worst case, Γ'_{CONTR}, considering equivalence as two axioms), while the proof of undecidability of unification in $\mathcal{SHI}$ [29] requires a large number of variable occurrences in (the analogous of) Γ. Hence deciding the satisfiability of the above formulas is an open problem.

6 Conclusion and Future Directions

This paper proposed an innovative approach exploiting the commonalities shared by several non-standard reasoning services in DLs to model them according to a unified framework. It is noteworthy that such a uniform view involves both the definition and the computation of the investigated tasks: on the one hand we propose a unique model to express the services as special Second-order sentences in DLs; on the other hand we provide a calculus for solving them according to a unified algorithmic approach. The unification potential of the proposed framework is shown in the paper w.r.t. several non-standard reasoning tasks apparently far from each other like Interpolation, Concept Unification, LCS, Concept Difference, Concept Abduction, Concept Contraction and Negotiation. We note that the framework is easily extensible for the computation of further reasoning tasks, like the Most Specific Concept and Knowledge Base Completion, by simply allowing formulas involving an ABox or a TBox in the definition model. The framework underlines how most non-standard services share the property of being devoted to the exhibition of one or more concepts and therefore names them "constructive reasoning tasks". Furthermore, constructive reasoning problems whose solution obeys to some optimality criteria, are more specifically modelled as "Optimal Solution Problem". The proposed unification effort will be finally capitalized by the construction of a unique system able to solve any non-standard reasoning task; the design and the implementation of such a system will be object of our future research work.

ACKNOWLEDGEMENTS

Work partially supported by Apulia projects PS_092 and PS_025

REFERENCES

[1] F. Baader, 'Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles', in *IJCAI 2003*, pp. 319–324, (2003).
[2] F. Baader, R. Küsters, and R. Molitor, 'Rewriting concepts using terminologies', in *KR 2000*, pp. 297–308, (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*, pp. 1–21, (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] M. Bienvenu, 'Complexity of abduction in the EL family of lightweight description logics', in *KR 2008*, pp. 220–230, (2008).
[7] A. Borgida, T. Walsh, and H. Hirsh, 'Towards measuring similarity in description logics', in *DL 2005*, (2005).
[8] J. R. Büchi, 'On a decision method in restricted second order arithmetic', in *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*, eds., E. Nagel et al., pp. 1–11. Stanford University Press, (1960).
[9] W. Cohen, A. Borgida, and H. Hirsh, 'Computing least common subsumers in description logics', in *AAAI'92*, eds., P. Rosenbloom and P. Szolovits, pp. 754–761, (1992). AAAI Press.
[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, F. M. Donini, and M. Mongiello, 'Abductive matchmaking using description logics', in *IJCAI 2003*, pp. 337–342, (2003).
[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*, pp. 739–745, (2009).
[13] C. Elsenbroich, O. Kutz, and U. Sattler, 'A case for abductive reasoning over ontologies', in *OWLED Workshop*, (2006).
[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] I. Horrocks, U. Sattler, and S. Tobies, 'Practical reasoning for very expressive description logics', *Logic J. of the IGPL*, **8**(3), (2000).
[17] B. Konev, D. Walther, and F. Wolter, 'Forgetting and uniform interpolation in large-scale description logic terminologies', in *IJCAI 2009*, pp. 830–835, (2009).
[18] F. Lécué, A. Delteil, and A. Léger, 'Applying abduction in semantic web service composition', in *ICWS 2007*, pp. 94–101, (2007).
[19] D. L. McGuinness and A. Borgida, 'Explaining subsumption in description logics', in *IJCAI'95*, pp. 816–821, (1995).
[20] I. S. E. Peraldi, A. Kaya, and R. Möller, 'Formalizing multimedia interpretation based on abduction over description logic aboxes', in *DL 2009*, (2009).
[21] M. O. Rabin, 'Decidability of second-order theories and automata on infinite trees', *Trans. of the Am. Math. Society*, **141**, 1–35, (1969).
[22] A. Ragone, 'OWL-DL as a power tool to model negotiation mechanisms with incomplete information', in *ISWC/ASWC 2007*, pp. 941–945, (2007).
[23] A. Ragone, T. Di Noia, F. M. Donini, E. Di Sciascio, and M. Wellman, 'Weighted description logics preference formulas for multiattribute negotiation', in *SUM'09*, LNAI, Springer-Verlag, (2009).
[24] S. Schlobach, 'Explaining subsumption by optimal interpolation', in *JELIA 2004*, pp. 413–425, (2004).
[25] I. Seylan, E. Franconi, and J. de Bruijn, 'Effective query rewriting with ontologies over dboxes', in *IJCAI 2009*, pp. 923–925, (2009).
[26] G. Teege, 'Making the difference: A subtraction operation for description logics', in *KR'94*, pp. 540–550, (1994).
[27] S. Tobies, *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*, Ph.D. dissertation, RWTH Aachen, 2001.
[28] J. Väänänen, 'Second-order logic and foundations of mathematics', *The Bulletin of Symbolic Logic*, **7**(4), 504–520, (2001).
[29] F. Wolter and M. Zakharyaschev, 'Undecidability of the unification and admissibility problems for modal and description logics', *ACM Trans. on Computational Logic*, **9**, (2008).