A Logical Framework for Modeling and Reasoning about Semantic Web Services Contract

Hai Liu[1,2,3], Qing Li[2,3], Naijie Gu[1,2] and An Liu[1,2,3]
[1]Department of Computer Science and Technology,
University of Science and Technology of China, Hefei, China.
[2]Joint Research Lab of Excellence, CityU-USTC Advanced Research Institute, Suzhou, China.
[3]Department of Computer Science, City University of Hong Kong, Hong Kong, China.
hail@mail.ustc.edu.cn, itqli@cityu.edu.hk, gunj@ustc.edu.cn, liuan@ustc.edu

ABSTRACT

In this paper, we incorporate concrete domain and action theory into a very expressive Description Logic (DL), called $\mathcal{ALCQO}$. Notably, this extension can significantly augment the expressive power for modeling and reasoning about dynamic aspects of services contracting. Meanwhile, the original nature and advantages of classical DLs are also preserved to the extent possible.

Categories and Subject Descriptors:
I.2.4 [Knowledge Representation Formalisms and Methods]: Representation languages

General Terms: Languages, Theory

Keywords: Semantic Web Services, DLs, Services Contract

1. INTRODUCTION

A key aspect in Web services behavior (described by services choreography) is contracting them to (i) make all parties compatible, (ii) guarantee expected QoS parameters, and (iii) ensure every participating service to terminate in a correct state. It is generally accepted that semantic Web services (SWS) should be based on a formalism with a well-defined model-theoretic semantics, particularly, on some sort of logics [2]. Given that DLs are usually considered as the underpinning of many W3C standard languages, it is quite natural and promising to resort to a variant of DLs to model a semantic Web services contract (SWSc). Our present work extends the classical DLs with concrete domain and action theory, which makes it feasible to integrate planning and process modeling into DLs yet still retain decidable reasoning ability.

2. THE FRAMEWORK

Intuitively, there exist multiple-parties constraints of web services in a SWSc, which bring to existing DLs more challenges. In particular, we consider the following aspect(s):

A1. The actions specified by a services contract are required to be coordinated (e.g., ordered) and all the constraints need to be represented. Meanwhile, the underlying logic should also retain the decidable reasoning ability for all the represented "knowledge". The case of possessing sufficient expressive power for establishing QoS requirements is also analogous.

Example 1. Consider the motivation example as in Figure 1. Apparently, the completion of a deposit by the user should precede the computer delivery, which is followed by the completion of the entire payment. Moreover, the user must complete the entire payment within 12 hours after the computer delivery.

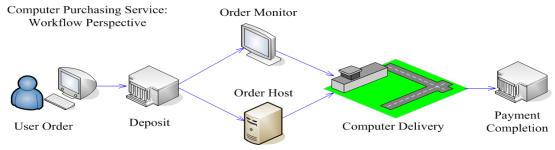

Figure 1: Motivating Example

Theoretically, combining original DLs with so-called "concrete domain" is a sound approach to satisfy the requirements as depicted in **A1**. However, [3] demonstrates that in the context of general TBoxes, if the concrete domain provides a unary predicate for equality with 0, a binary equality predicate, and a binary predicate for incrementation, then concept satisfiability and subsumption are undecidable. This result actually rules out the possibility of combining general TBoxes with more powerful concrete domains. In the sequel, we disallow general TBoxes in our framework but adopt acyclic TBoxes instead. The concrete domain in our framework sits on the basis of [3] and [4], and is defined as follows:

Definition 1. (Arithmetic concrete domain $\mathcal{Q}^*$) Our concrete domain is denoted as $\mathcal{Q}^*$ and is a pair $(\triangle_\mathcal{D}, \Phi_\mathcal{D})$, where $\triangle_\mathcal{D}$ refers to the rational numbers $\mathbb{Q}$, and $\Phi_\mathcal{D}$ consists of the following predicates:
(1). unary predicates P_q for each $P \in \{\leqslant, <, =, \neq, >, \geqslant\}$ and each $q \in \mathbb{Q}$ where $(P_q)^\mathcal{D} = \{q' \in \mathbb{Q} \mid q'Pq\}$, two unary predicate $\top_{\mathcal{Q}^*}$ with $(\top_{\mathcal{Q}^*})^\mathcal{D} = \mathbb{Q}$, and $\bot_{\mathcal{Q}^*}$ with $(\bot_{\mathcal{Q}^*})^\mathcal{D} = \emptyset$;
(2). binary predicates $P_b = \{\leqslant, <, =, \neq, >, \geqslant\}$;
(3). ternary predicates $P_t = \{+, \mp, *, \circledast\}$ with $(+)^\mathcal{D} = \{(q, q', q'') \in \mathbb{Q}^3 \mid q+q' = q''\}$, $(\mp)^\mathcal{D} = \mathbb{Q}^3 \backslash (+)^\mathcal{D}$, $(*)^\mathcal{D} = \{(q, q', q'') \in \mathbb{Q}^3 \mid q*q' = q''\}$, and $(\circledast)^\mathcal{D} = \mathbb{Q}^3 \setminus (*)^\mathcal{D}$;

We denote our underlying logic as $\mathcal{ALCQO}(\mathcal{Q}^*)$ with its syntax and Tarski-style semantics shown in Table 1 (f, g and U denote abstract feature, concrete feature, and feature path, respectively).

THEOREM 1. *(Decidability) An $\mathcal{ALCQO}(\mathcal{Q}^*)$-concept satisfiability is decidable.*

Example 1 revisited. At first, we introduce three types of concrete features: atTime, lBound, and rBound, whose intuitive meanings are the occurring time, the left and right bound of occurring time interval. We denote an abstract feature *c_Deli* as an abbreviation for computer delivery. Similarly, we denote *c_Pay* and *interval* as abbreviations for complete payment and the allowed time interval between *c_Deli* and *c_Pay*, respectively. Therefore, from the perspective of computer purchase service (CP), the interactions

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

Table 1: Syntax and Semantics of $\mathcal{ALCQO}(\mathcal{Q}^*)$

Syntax	Semantics
$\{o\}$	$\{o\}^I \subseteq \triangle^I$, $\sharp(\{o\}^I) = 1$
$\neg C$	$(\neg C)^I = \triangle^I \setminus C^I$
$C \sqcap D$	$(C \sqcap D)^I = C^I \cap D^I$
$C \sqcup D$	$(C \sqcup D)^I = C^I \cup D^I$
$\exists R.C$	$(\exists R.C)^I = \{a \mid \exists b. < a, b > \in R^I$, and $b \in C^I\}$
$\forall R.C$	$(\forall R.C)^I = \{a \mid \forall b.(< a, b > \in R^I \rightarrow b \in C^I)\}$
$\leqslant nR.C$	$(\leqslant nR.C)^I = \{a \mid \sharp(\{b \mid < a, b > \in R^I\} \cap C^I) \leqslant n\}$
$\geqslant nR.C$	$(\geqslant nR.C)^I = \{a \mid \sharp(\{b \mid < a, b > \in R^I\} \cap C^I) \geqslant n\}$
$U = f_1 \cdots f_n g$	$U^I = \{< a, x > \mid \exists b_1, \cdots b_n \in \triangle^I, x \in \triangle^I_{\mathcal{D}}.(< a, b_1 > \in f_1^I, \cdots < b_{n-1}, b_n > \in f_n^I$, and $< b_n, x > \in g^I)\}$
$\exists U.P_q$	$\{a \in \triangle^I \mid \exists x \in \triangle^I_{\mathcal{D}}.(< a, x > \in U^I$, and $x \in P_q^{\mathcal{D}})\}$
$\exists U_1, U_2.P_b$	$\{a \in \triangle^I \mid \exists x_1, x_2 \in \triangle^I_{\mathcal{D}}.(< a, x_1 > \in U_1^I, < a, x_2 > \in U_2^I$, and $< x_1, x_2 > \in P_b^{\mathcal{D}})\}$
$\exists U_1, U_2, U_3.P_t$	$\{a \in \triangle^I \mid \exists x_1, x_2, x_3 \in \triangle^I_{\mathcal{D}}.(< a, x_1 > \in U_1^I, < a, x_2 > \in U_2^I, < a, x_3 > \in U_3^I$, and $< x_1, x_2, x_3 > \in P_t^{\mathcal{D}})\}$
$g\uparrow$	$(g\uparrow)^I = \{a \in \triangle^I \mid \neg\exists x \in \triangle^I_{\mathcal{D}}.(< a, x > \in g^I)\}$

among these services can be specified as follows:

$$\begin{aligned} CP\_Require \doteq\ & (\exists deposit \circ atTime, c\_Deli \circ lBound. <) \sqcap \\ & (\exists c\_Pay \circ atTime, c\_Deli \circ rBound. \geqslant) \sqcap \\ & (\exists interval, c\_Deli \circ rBound, c\_Pay \circ atTime.+) \\ & \sqcap(\exists interval. \leqslant_{12}) \sqcap (\exists lBound, rBound. \leqslant) \end{aligned}$$

3. ACTION THEORY

In this section, we start with providing some very practical scenarios to illustrate the significance of actions.

A2. The so-called "state of the world" (SoW) is generally assumed to be altered through invoking a service operation, viz. the SoW for a services contract is not permanent. If the underlying logic can entail the current state of a contract, the effect produced by an action should also be entailed by the "updated" SoW.

Example 2. Consider again the CP service. If the computer delivery is executed successfully, it will cause some effect to the other involved parties, e.g., the computer ownership will be changed. Therefore, the preconditions of next action should be satisfied, otherwise the entire services contract may be blocked at certain state. Meanwhile, it also needs to ensure that the contract can actually enter into some expected state eventually.

Our action theory is based on [1]. Considering the main features of semantic Web services, an action is defined as follows:

Definition 2. (Action): An action is a quadruple (*input*, *pre*, *output*, *eff*), where:

(1). *input* and *output* consist of a finite set of concepts, and each concept should have at least one instance in some particular SWSc state, so that the legality of *input* and *output* data can be preserved.

(2). *pre* consists of a finite set of ABox assertions, representing preconditions for the action;

(3). *eff* consists of a finite set of conditional effect descriptions in the form of φ/χ, where φ is a set of ABox assertions and χ is a set of assertions in the form of C(a), $\neg C(a)$, $R(a, b)$, $\neg R(a, b)$, g(a,x), $\neg g(a, x)$, $P_q(x)$, $P_b(x, y)$, and $P_t(x, y, z)$ with C being a primitive concept, R a role, and g a concrete feature w.r.t. $\mathcal{T}$.

Example 3. Consider an ABox as {user(peter), hasDeposit(peter,A1), amount(A1,amt), $=_{1000}$ (*amt*), computer(L3YG), vendor(CP), own(CP,L3YG), bank(Deutsche), affiliated(A1,Deutsche)}, and action computer delivery can be described as follows (*input* and *output* are omitted for simplicity):

$$\begin{aligned} pre &= \{\exists hasDeposit.(\forall affiliated.\{Deutsche\} \sqcap \\ &\quad \exists amount. \geqslant_{800})(peter), user(peter)\} \\ eff &= \{\top(peter)/(own(peter, L3YG), \neg own(CP, L3YG)))\} \end{aligned}$$

Definition 3. A Semantic Web Services contract (SWSc) is represented as a triple $\mathcal{SC} =< \mathcal{T}, \mathcal{A}, \mathcal{ACT} >$ such that:

(1). $\mathcal{T}$ corresponds to a terminology box (TBox) in DLs, which describes the skeleton of the contract.

(2). $\mathcal{A}$ stands for an axiom box (ABox).

(3). $\mathcal{ACT}$ describes all the legal actions to be executed in $\mathcal{SC}$. These actions can "drive" a contract's state to be transitioned.

Intuitively, $\mathcal{T}$ plays the role of orchestrating a set of specified actions, $\mathcal{A}$ represents the current state of the entire contract, and $\mathcal{ACT}$ provides the descriptions of a set of actions. The interplay among them is thus obvious and natural, since TBoxes coordinate the action set, and ABoxes can be updated through the execution of an action. The formal semantics of our $\mathcal{SC}$ is defined as follows:

Definition 4. (State) A state of a $\mathcal{SC}$ is a set of assertions in the form of $C(a)$, $\neg C(a)$, $R(a, b)$, $\neg R(a, b)$, g(a,x), $\neg g(a, x)$, $P_q(x)$, $P_b(x, y)$, and $P_t(x, y, z)$ with C being a (complex) concept name.

Definition 5. (Model). Let α be an action, a transition T_α is a set of pairs $< s, s' >$, where s and s' are states. A model $\mathfrak{M}$ of a services contract is defined as: $\mathfrak{M} =< \mathfrak{S}, \mathfrak{T} >$, where $\mathfrak{S}$ is a set of states, and $\mathfrak{T}$ is a set of transitions of states in $\mathfrak{S}$.

In our framework, we consider state transition as ABox updates, i.e. if a $\mathcal{SC}$ enforces an action successfully, it is regarded as updating the ABox from the current state to a new state. Based on the above definitions, we can obtain the following lemma.

LEMMA 1. *(Indeterminism) The update result of an ABox after an action is applied is indeterministic, i.e. there may exist more than one possible state to be the consequence of an update.*

LEMMA 2. *The amount of possible consequent states through applying an action to the current state is finite.*

With the decidability of $\mathcal{ALCQO}(\mathcal{Q}^*)$ itself and the finite consequent states due to ABoxes updating strategy, we can obtain:

THEOREM 2. *(Well-defined) Given a services contract $\mathcal{SC}$, checking its consistency, executability, and projection is decidable.*

4. CONCLUSION AND FUTURE WORK

We restrict the underpinning of our framework to a comparatively simple yet quite expressive Description Logic, viz., $\mathcal{ALCQO}$. Meanwhile, we have incorporated concrete domain and action theory into $\mathcal{ALCQO}$, so as to equip it with the abilities to describe, coordinate, and trace the dynamic behaviors of involved parties.

In future, we will try to work out the tight complexity bounds of action reasoning, and implement some practical algorithms.

5. REFERENCES

[1] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. AAAI 2005*, pages 572–577, July 2005.

[2] H. Davulcu, M. Kifer, and I.V.Ramakrishnan. Ctr-s: A logic for specifying contracts in semantic web services. In *Proc. WWW 2004*, pages 144–153. ACM, May 2004.

[3] C. Lutz. Description logics with concrete domains - a survey. *Advances in Modal Logic*, 4:265–296, 2003.

[4] C. Lutz. Combining interval-based temporal reasoning with general tboxes. *Artificial Intelligence*, 152(2):235–274, 2004.