Reasoning about Semantic Web Services with an Approach Based on Temporal Description Logic

Juan Wang, Liang Chang, Chuangying Zhu, Rongsheng Dong

Reasoning about Semantic Web Services with an Approach Based on Temporal Description Logic

Juan Wang, Liang Chang, Chuangying Zhu, Rongsheng Dong

Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
wangjuan_980644@163.com, changl@guet.edu.cn, ccrsdong@guet.edu.cn

Abstract. Temporal description logic *ALC*-LTL not only has considerable expressive power, but also extends the description capability of description logic from the static domain to the dynamic domain. In this paper, *ALC*-LTL is applied for the composition of semantic Web services. We take the view that atomic process and composite process in the OWL-S ontology can be considered as atomic service and composited service respectively. Inputs, outputs, local variables, preconditions and results of atomic processes can all be described with *ALC*-LTL. Based on the models of services, the executability problem and the projection problem of Web services can be reasoned about effectively.

Keywords: temporal description logic; semantic Web services; OWL-S; executability problem; projection problem

1 Introduction

The target of semantic Web service is to support automation of Web Services discovery, composition and execution by means of adding sufficient semantic information in the description of Web services [1]. OWL-S is an ontology represented by the Web Ontology Language OWL for describing Web services on the semantic Web environment. From the point of view of knowledge representation, the description of semantic Web services should contain not only static information but also dynamic information.

Description logic is the logic foundation of OWL. Although description logic provides considerable expressive power for describing knowledge of static domain, it cannot be used effectively to describe and reason about dynamic knowledge. According to the characteristics and application requirements of semantic Web, researchers have proposed kinds of extensions of description logics. Temporal description logic *ALC*-LTL is proposed by Baader et al. [2] as a combination of the description logic *ALC* and the linear temporal logic LTL. It not only has considerable expressive power for describing knowledge of both the static domain and the dynamic domain, but also is decidable and EXPTIME-complete for the satisfiability problem of formulas [2]. In this paper, we will use *ALC*-LTL to model and reason about semantic Web services.

Reasoning problems investigated in this paper are the executability problem and the projection problem of semantic Web services. The executability problem will check whether a semantic Web services is executable or not w.r.t. a given initial state, and the projection problem will check whether a certain formula holds or not after the successful execution of the service [3].

In recent years, many formalisms have been proposed for these reasoning problems. McCarthy et al. [4] investigate these reasoning problems with the formalism of situation calculus. Calvanese et al. [5] investigate these reasoning problems based on LTL. Baader et al. [6] propose an action formalism based on description logic; the executability problem and the projection problem are investigated in this formalism and both of them are reduced to the consistency problem of ABoxes. As an application of that formalism, Baader et al. [3] model semantic Web services as actions and then the executability and the projection problems of semantic Web services can be investigated. Chang et al. [7] proposed a family of dynamic description logics named $DDL(X^{@})$ for representation and reasoning about actions.

In this paper, starting with the Process Model of OWL-S, we take the view that atomic process in OWL-S can be considered as atomic service and composite process in OWL-S as composited service. Therefore, the reasoning about composite process can be treated as the reasoning about composited service. Inputs, outputs, local variables, preconditions and results of the atomic processes are described with *ALC*-LTL. Based on the modeling of composited service, executability and projection problems of semantic Web services are reasoned about.

2 Temporal Description Logic *ALC*-LTL

The temporal description logic *ALC*-LTL combines description logic *ALC* with linear temporal logic LTL. Syntactically, *ALC*-LTL uses general concept inclusion axioms, concept assertions and role assertions of the description *ALC* instead of atomic propositions of LTL.

Concepts of *ALC*-LTL are constructed inductively as follows [2]:

$$C, D ::= C_i \mid \neg C \mid C \sqcup D \mid \forall R.\, C$$

where $C_i \in N_C$, $R \in N_R$. N_C and N_R respectively be disjoint sets of concept names and role names. Moreover, $C \sqcap D$ and $\exists R.C$ are introduced as abbreviations of $\neg(\neg C \sqcup \neg D)$ and $\neg(\forall R.\neg C)$.

An expression of the form of $C \sqsubseteq D$ for two concepts C and D is called a general concept inclusion axiom(GCI).Let N_I be a set of individual names. Expressions of the forms $C(p)$ and $R(p,q)$ are called concept assertion and role assertion respectively, where $p,q \in N_I$ and $R \in N_R$.

Formulas of *ALC*-LTL are constructed inductively as follows [2]:

$$\varphi, \psi ::= C \sqsubseteq D \mid C(p) \mid R(p,q) \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathbf{X}\varphi \mid \varphi \mathbf{Y} \psi$$

where p, $q \in N_I$, $R \in N_R$ and C,D are concepts. Using these constructors, several other constructors can be defined as abbreviations: $false := \varphi \wedge \neg\varphi$, $true := \neg false$, $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$, $\boldsymbol{F}\varphi := true\mathbf{Y}\,\varphi$, $\boldsymbol{G}\varphi := \neg\boldsymbol{F}\neg\varphi$.

GCIs, concept assertions and role assertions are called *ALC*-assertion. *ALC*-assertion and its negation are called *ALC*-literals.

Semantically, the interpretation structure of *ALC*-LTL is similar with LTL, and states are organized by the progress of time. But the mapping of every state in *ALC*-LTL interpretation structure is not a set of atomic propositions, but an interpretation of *ALC*.

An *ALC-LTL interpretation structure* [2] is a pair $M = (\mathbb{N}, I)$, where,

(1) $\mathbb{N}$ is the set of natural numbers;

(2) For every natural number $n \in \mathbb{N}$, description logic *ALC* interpretation of function I is $I(n)=(\Delta, \bullet^{I(n)})$,where interpretation function $\bullet^{I(n)}$ satisfies following conditions:

① each concept name $C_i \in N_C$ is interpreted as a subset $C_i^{I(n)} \subseteq \Delta$ of Δ;

② each role name $R_i \in N_R$ is interpreted as a binary relation $R_i^{I(n)} \subseteq \Delta \times \Delta$ of Δ.

③ each individual name $p_i \in N_I$ is interpreted as a element $p_i^{I(n)} \in \Delta$ of Δ, and for all natural number $m \in \mathbb{N}$ such that $p_i^{I(n)} = p_i^{I(m)}$.

Given an *ALC*-LTL interpretation structure $M = (\mathbb{N}, I)$, the semantics of concepts and formulas are defined inductively as follows [2].

Firstly, for any natural number $n \in \mathbb{N}$, each concept C is interpreted as a set $C^{I(n)} \subseteq \Delta$. The semantics of concepts of *ALC*-LTL are defined inductively as follows:

(1) $(\neg C)^{I(n)} := \Delta \setminus C^{I(n)}$, where $\setminus$ is the operator of set difference;

(2) $(C \sqcup D)^{I(n)} := C^{I(n)} \cup D^{I(n)}$,where $\cup$ is the operator of set union;

(3) $(\forall R.C)^{I(n)} := \{x \mid \text{for all } y \in \Delta, (x, y) \in R^{I(n)} \text{ implies } y \in C^{I(n)}\}$.

Secondly, for any natural number $n \in \mathbb{N}$, $(M,n) \models \varphi$ represents formula φ is true at an instant n. The semantics of formulas of *ALC*-LTL are defined inductively as follows:

(4) $(M,n) \models C \sqsubseteq D$ iff $C^{I(n)} \subseteq D^{I(n)}$;

(5) $(M,n) \models C(p)$ iff $p^{I(n)} \in C^{I(n)}$;

(6) $(M,n) \models R(p,q)$ iff $(p^{I(n)}, q^{I(n)}) \in R^{I(n)}$;

(7) $(M,n) \models \neg\varphi$ iff $(M,n) \not\models \varphi$;

(8) $(M,n) \models \varphi \wedge \psi$ iff $(M,n) \models \varphi$ and $(M,n) \models \psi$;

(9) $(M,n) \models \boldsymbol{X}\varphi$ iff $(M,n+1) \models \varphi$;

(10) $(M,n) \models \varphi\mathbf{Y}\psi$ iff there is $k \geq 0$ such that $(M,n+k) \models \psi$ and $(M,n+i) \models \varphi$ for all i, with $0 \leq i < k$.

A *ALC*-LTL formula φ is *satisfiable* iff there is an *ALC*-LTL interpretation structure $M=(\mathbb{N}, I)$ such that $(M,0) \models \varphi$.

The most basic reasoning problem in *ALC*-LTL is satisfiability problem. Baader et al. [2] has proved that this problem is EXPTIME-complete.

3 Modeling of Semantic Web Services

As a new generation of Semantic Web Services framework, OWL-S [8] has a broad perspective in application. OWL-S describes a semantic Web service by providing Service Profile, Service Model and Service Grounding. Service Profile describes what the service does; Process Model specifies how the service works; Service Grounding deals with the realization of services. Our work focuses more on Process Model.

Process Model has three kinds of processes: atomic process, composite process and simple process. Simple processes are non-invokable processes. Simple processes are not taken account of when modeling of the Web services in this paper. Flight reservation service BravoAir based on OWL-S1.2 has been described by OWL alliance. It is composed of two atomic processes, GetDesiredFlightDetails and SelectAvailableFlight, and a composite process, BookFlight. The process BookFlight is the composed of two atomic processes, LogIn and CompleteReservation. LogIn and BravoAir are described as Fig.1 and Fig.2 respectively.

(1) Modeling of atomic process.

An atomic process is a description of a service which can be executed in single step. It composes of four parts: inputs, outputs, local variables, preconditions and results.

```
define atomic process LogIn
   (inputs:(LogIn_AcctName-AcctName
            LogIn_Password-Password),
    outputs:(LogIn_Output-boolean),
    results: ((hasPassword(LogIn_AcctName,LogIn_Password)
               |->LogIn_Output(true)
                  &LoggedIn(LogIn_AcctName))
              (Correct_Password-Password
               &LogIn_Password -Password
               &hasPassword(LogIn_AcctName,Correct_Password)
               |->LogIn_Output(false)
                  &NotLoggedIn(LogIn_AcctName))
             )
   )
```

Fig. 1. Atomic process LogIn

Inputs and outputs reflect messages exchange between the service invoker and the services. Local variables are used for depicting preconditions and results. This example doesn't involve it. Every input, output or local variables is comprised of variable name and type declaration. For example, in Fig.1, LogIn_AccName is the variable name; its type declaration is AccName. Let variable names and type declaration are x and C respectively, it can be described by means of the concept assertion $C(x)$, that is, LogIn_AccName-AccName can be described with LogIn_AccName(*AccName*).

The preconditions represented by logical formulas describe the satisfied condition before invoking the services. Results represent the effects after executing the services.

Each result is composed of a condition and a set of effects and output bindings. They can be represented by logical formulas. When the Web services are executed, corresponding effects and outputs will be generated according to the condition. The preconditions, effects and the output bindings are depicted by means of Semantic Web Rule Language (SWRL). SWRL is a kind of rule language based on OWL and has considerable expressive power. SWRL-Condition is the conjunction of atoms in SWRL. Operator "∧" is used to construct the formulas. For example, LogIn_Output(true)&LoggIn(LogIn_AccName) can be represented as LogIn_Output(*true*)∧LoggIn(*LogIn_AccName*).

```
Define composite process BravoAir
   (inputs:(DepartureAirport-Airport
            ArrivalAirport-Airport
            OutboundDate-FlightDate
             …                          ),
    outputs:(FlightsFound-FlightList
             PreferredFlightItinerary-FlightItinerary
             ReservationID-ReservationNumber),
    results:(AlwaysTrue
             |->FlightsFound<=GetDesiredFlightDetails.GetDesiredFlightDetails_FlightsFound
             &PreferredFlightItinerary<=BookFlight.BookFlight_PreferredFlightItinerary
             &ReservationID<=BookFlight.BookFlight_ReservationID
             &HasFlightItinerary(TheClient,PreferredFlightItinerary))
   )
{GetDesiredFlightDetails(GetDesiredFlightDetails_DepartureAirport<=TheParentPerform.DepartureAirport,
                         GetDesiredFlightDetails_ArrivalAirport<=TheParentPerform.ArrivalAirport,
                         …                                                                    );
 SelectAvailableFlight(SelectAvailableFlight_FlightsAvailable<=GetDesiredFlightDetails.GetDesiredFlightDetails_FlightsFound);
 BookFlight(BookFlight_SelectedFlight<=SelectAvailableFlight.SelectAvailableFlight_SelectedFlight,
            BookFlight_AcctName<=TheParentPerform.BookFlight_AcctName,
            BookFlight_Password<=TheParentPerform.Password)
}
```

Fig. 2. Composite process BravoAir

(2) Modeling of composite process.

A composite process is composed of sub composite or atomic processes by control constructs. It mainly contains inputs, outputs, results and the body of the composite process. The body of a composite process is composed of control constructs and binding relationships. For example, in Fig.2, input DepartureAirport of GetDesiredFlightDetails binds with input GetDesiredFlightDetails_DepartureAirport of GetDesiredFlightDetails when composite process BravoAir invokes atomic process GetDesiredFlightDetails. The representations of inputs and outputs are similar with atomic processes. Results are the binding of inputs and outputs of the subprocesses. For example, output GetDesiredFlightDetails_FlightsFound of subporcess GetDesiredFlightDetails binds with output FlightsFound of composite process BravoAir.

We take the view that atomic process in OWL-S can be considered as atomic service and composite process in OWL-S as composited service. In the composited service, if the inputs or outputs has binding relationship with the inputs of composite process, the inputs that have binding relationship are replaced by the inputs of the composite processes; if the inputs or outputs has binding relationship with the outputs

of the composite process, the outputs that exist binding relationship, are replaced by the outputs of the composite processes.

For each service s_i of the composited service $s_0,s_1,...s_k$ ($0 \le i < k,\ i,k \in \mathrm{N}$), let $\boldsymbol{P}_i$ be a set of precondition formulas, representing the preconditions that should be satisfied before executing the services s_i. Let $\boldsymbol{E}_i$ be a set of effect formulas, representing the effects that should be satisfied after executing the services. $\boldsymbol{P}_i$ and $\boldsymbol{E}_i$ can be described as follows:

① For each input or local variable of the atomic process, adding the form of formulas $C(x)$ in $\boldsymbol{P}_i$;
② For each precondition of the atomic process, adding the form of formulas $C(x)$ in $\boldsymbol{P}_i$;
③ For each output of the atomic process, adding the form of formulas $C(x)$ in $\boldsymbol{E}_i$;
④ For each result r_{ij} ($j \in \mathbb{N}$) of the atomic process, it combines a precondition and a group of effects and output constraints, let the precondition be φ_j, and the effects and output constraints transform to the formula $\psi_{j1},\psi_{j2},...,\psi_{jt}$ ($t \in \mathbb{N}$), then adding φ_j in $\boldsymbol{P}_i$, adding $\psi_{j1},\psi_{j2},...,\psi_{jt}$ in $\boldsymbol{E}_i$.

4 Reasoning about Semantic Web Services

Before trying to apply the service, we want to know whether it is indeed executable, this is an executability problem. If the service is executable, we may want to know whether applying it achieves the desired effect, this is a projection problem. These problems are basic inference problems considered in the reasoning about semantic Web services [3].

In order to describe reasoning tasks, composited service system should be described first of all. *ALC*-LTL literals are divided into two mutually disjoint sets, L_F and L_S, where L_F represents the set of *ALC*-LTL literals which are true on the current situation; each *ALC*-LTL literal in L_S corresponds to a service name s_i, and represents that the service s_i has just been performed. For any set $\boldsymbol{Q}$ composed of formulas, we use $Conj(\boldsymbol{Q})$ to denote the conjunction of all the elements of $\boldsymbol{Q}$. The specification of dynamic system is as follows:

(1) For each service s_i in L_S, the effects of the services are specified by means of formulas of the form (1)

$$\boldsymbol{G}(Conj(\boldsymbol{P}_i) \rightarrow \boldsymbol{X}(s_i \rightarrow Conj(\boldsymbol{E}_i))) \tag{1}$$

Formula (1) shows that service s_i is executed under the conditions denoted by $Conj(\boldsymbol{P}_i)$ brings about the conditions denoted by $Conj(\boldsymbol{E}_i)$.

(2) For all *ALC*-LTL literal F in L_F, the frame axioms as formula (2) shows:

$$\boldsymbol{G}(\boldsymbol{X}F \leftrightarrow \bigvee_a (\boldsymbol{P}_a \wedge \boldsymbol{X}a) \vee (F \wedge \bigwedge_b (\neg \boldsymbol{P}_b \vee \boldsymbol{X}\neg b)) \tag{2}$$

where $F \in L_F$. The atomic services a are those services that under the circumstances described by P_a make F become true, and b are those services that under the circumstance described by P_b make F become false. All *ALC*-LTL literal F can be depicted by using of frame axioms.

(3) The initial situation can be described with the expression, $Conj(\boldsymbol{P}_{init})$. $\boldsymbol{P}_{init}$ is a set of concept assertions such that $\boldsymbol{P}_{init} \subseteq L_F$.

Taking the process BookFlight as an example, it can be described as follows:

If AcctName is *Tom*, Password is *123456*, Confirm information is *confirmation* and SelectedFlight is *FlightItineraryList*.

First of all, atomic process LogIn is modeled. If the AcctName and the Password are correct, user logs in the system successfully. This service can be described as formula (3).

$$\boldsymbol{G}(Conj(\boldsymbol{P}_{\text{LogIn}}) \rightarrow \boldsymbol{X}(\text{LogIn} \rightarrow Conj(\boldsymbol{E}_{\text{LogIn}})) \tag{3}$$

where $Conj(\boldsymbol{P}_{\text{LogIn}})$=AcctName(*Tom*)∧Password(*123456*)∧hasPassword(*Tom,123456*), $Conj(\boldsymbol{E}_{\text{LogIn}})$=Output(*true*)∧LoggedIn(*Tom*).

When the output of atomic process LogIn is true, the user chooses the flight from *FlightItineraryList* and confirms it. The description of ConfirmReservation is shown as formula (4).

$$\boldsymbol{G}(Conj(\boldsymbol{P}_{\text{ConfirmReservation}}) \rightarrow \boldsymbol{X}(\text{ConfirmReservation} \rightarrow Conj(\boldsymbol{E}_{\text{ConfirmReservation}})) \tag{4}$$

where $Conj(\boldsymbol{P}_{\text{ConfirmReservation}})$=SelectedFlight(*FlightItineraryList*)∧LoggedIn(T*om*)∧Output(*true*)∧ Confirm(*comfirmation*),$Conj(\boldsymbol{E}_{\text{ConfirmReservation}})$=preferredFlightItinerary(*BeiJing-Paris*)∧ReservationID(*20123039*).

Composite process (LogIn,ConfirmReservation) can be described by formula (3) and formula (4).

Next, frame axioms are used to describe the *ALC*-LTL literal in L_F, for example, LoggedIn(*Tom*) can be described as formula (5).

$$\begin{aligned}\boldsymbol{X}\text{LoggedIn}(Tom) \leftrightarrow ((Conj(P_{\text{LogIn}}) \wedge \boldsymbol{X}\text{LogIn}) \vee (\text{LoggedIn}(Tom) \wedge \\ (\neg Conj(P_{\text{LogIn}}) \vee \boldsymbol{X} \neg \text{LogIn})\end{aligned} \tag{5}$$

Finally, the initial situation can be described as formula (6).

$$Conj(\boldsymbol{P}_{init})=\text{AcctName}(Tom) \wedge \text{Password}(1253456) \wedge \text{hasPassword}(Tom,123456) \tag{6}$$

A service is executable if performing it does not contradict such a truth assignment, i.e., in current situation, if F is true, then the service s_i can be executed, else s_i can't be executed.

Let Γ be the formula descriptions of the services system, the expression $Occurs(s_0,s_1,\ldots,s_k,rs)$ be the formula (7).

$$(s_0 \wedge \boldsymbol{X}(s_1 \wedge \boldsymbol{X}(\ldots \boldsymbol{X}(s_k \wedge rs)\ldots))) \wedge \boldsymbol{G}(rs \rightarrow \boldsymbol{X}\boldsymbol{G} \neg rs) \tag{7}$$

Formula (7) expresses that the sequence of services $s_0,s_1,\ldots,s_k$ occurs, resulting in a situation denoted by the new *ALC*-LTL literal *rs*, and *rs* is true only once. *rs* acts a marker for the situation resulting by excuting $s_0,s_1,\ldots,s_k$ [5]. Projection problem can be reduced to the validity problem of formula (8):

$$\Gamma\rightarrow(Occurs(s_0,s_1,...,s_k,rs)\rightarrow\boldsymbol{G}(rs\rightarrow\varphi)) \tag{8}$$

The reasoning tasks above can be transformed to the satisfiability problem. Tableau decision algorithm for *ALC*-LTL can be used to verify the satisfiability problem. Consequently, it makes the verification of executability and projection problems of the composited service become true.

There are two problems need to be verified. First, composited service (LogIn, ConfirmReservation) is indeed executable; second, the formula LoggedIn(*Tom*) is indeed true after executing the services (LogIn,ConfirmReservation).

Let us introduce the formula *Occurs*(LogIn,ConfirmReservation,$_{rs}$), which is formula (9).

$$(\text{LogIn}\wedge\boldsymbol{X}(\text{ConfirmReservation}\wedge rs))\wedge\boldsymbol{G}(rs\rightarrow\boldsymbol{XG}\neg rs) \tag{9}$$

Because in current situation, AcctName(*Tom*), Password(*123456*) and hasPassword(*Tom,123456*) in $\boldsymbol{P}_{init}$ are true, LogIn can be executed. In the same way, all elements in $\boldsymbol{P}_{\text{ConfirmReservation}}$ are true, ConfirmReservation can be executed. Composited service (LogIn, ConfirmReservation) is executable.

Let Γ be the description of BookFlight, the projection problem can be described as formula (10).

$$\Gamma\rightarrow(Occurs(\text{LogIn,ConfirmReservation},_{rs})\rightarrow\boldsymbol{G}(rs\rightarrow\text{LoggedIn}(Tom))) \tag{10}$$

Formula (10) is satisfiable, so after executing the services (LogIn, ConfirmReservation), the formula LoggedIn(*Tom*) is true.

5 Conclusion

In this paper, we take the temporal description logic *ALC*-LTL proposed by Baader et al. as a tool for reasoning about semantic Web services. By treating each atomic process represented in OWL-S as an atomic service, the inputs, outputs, local variables, preconditions and results of atomic processes are all described by *ALC*-LTL. Based on modeling of the composited service, the reasoning problems of semantic Web services, such as executability and projection problems can be carried out.

One of our future works is to study the verification of composed Web services based on the reasoning mechanisms investigated in this paper. Another work is to design decision algorithm for *ALC*-LTL and develop a reasoning tool for it.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Nos. 60903079, 60963010, 61163041), the Natural Science Foundation of Guangxi Province (No.2012GXNSFBA053169), and the Innovation Project of Guangxi Graduate Education (No. 2011105950812M21).

References

1. MeIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems 16(2),46-53 (2001)
2. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. In: Brewka, G., Lang, J. (eds.) Proceeding of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08), pp.684-694. AAAI Press, Cambridge (2008)
3. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: A description logic based approach to reasoning about Web services. In: Vasiliu, L. (eds.) Proceeding of the WWW 2005 Workshop on Web Service Semantics: Towards Dynamic Business Integration, pp.636-647. ACM Press, Chiba (2005)
4. McCarthy, J., Hayes, P.: Some philosophical problems form the standpoint of artificial intelligence. Machine Intelligence 4, 463-502 (1969)
5. Calvanese, D., De Giacomo, G., Vardi, M.: Reasoning about actions and planning in LTL action theories. In: Fensel, D., Giunchiglia, F., McGuinness, D., Willians, M. (eds.) Proceeding of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR'02), pp. 593-602. Morgan Kaufmann, San Francisco (2002)
6. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: first results. In: Veloso, M., Kambhampati, S. (eds.) Proceeding of the 12th National Conference on Artificial Intelligence (AAAI'05), pp. 572-577. AAAI Press, Menlo Park (2005)
7. Chang, L., Shi, Z., Gu, T., Zhao, L.: A family of dynamic description logics for representing and reasoning about actions. Journal of Automatic Reasoning 49(1),1-52 (2012)
8. The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Technical report, http://www.daml.org/services/(2002)