University of Wollongong
Research Online

Faculty of Informatics - Papers (Archive)

Faculty of Engineering and Information Sciences

2006

Using Assumptions in Service Composition Context

Z. Lu
University of Wollongong, lu@uow.edu.au

Aditya K. Ghose
University of Wollongong, aditya@uow.edu.au

P. Hyland
University of Wollongong, phyland@uow.edu.au

Y. Guan
University of Wollongong, yguan@uow.edu.au

Publication Details

This article was originally published as: Lu, Z, Ghose, A, Hyland, P & Guan, Y, Using Assumptions in Service Composition Context, IEEE International Conference on Services Computing (SCC '06), Chicago, USA, September 2006, 289-292. Copyright IEEE 2006.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Using Assumptions in Service Composition Context

Abstract
Service composition aims to provide the efficient and accurate model of a service, based on which the global service oriented architecture (SOA) can be realized, allowing the value-added services to be generated on the fly. Because of distributed responsibilities, ownership, and control, often, it is not feasible to acquire all information needed for the service composition, thus there might be no guarantee that the service execution has an anticipated effect. In this paper, we are going to extend current Semantic Web Service Description by introducing the concept of "Service Assumption" which allows reasoning with incomplete information. Furthermore, together with the proposed service assumption, a sequence of rules is developed to describe all permitted behaviors in service composition context.

Disciplines
Physical Sciences and Mathematics

Publication Details
This article was originally published as: Lu, Z, Ghose, A, Hyland, P & Guan, Y, Using Assumptions in Service Composition Context, IEEE International Conference on Services Computing (SCC '06), Chicago, USA, September 2006, 289-292. Copyright IEEE 2006.

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/502

Using Assumptions in Service Composition Context

Zheng Lu, Aditya Ghose, Peter Hyland, Ying Guan
School of IT & Computer Science
University of Wollongong, Australia
{zl07, aditya, phyland, yg32}@uow.edu.au

Abstract

Service composition aims to provide the efficient and accurate model of a service, based on which the global service oriented architecture (SOA) can be realized, allowing the value-added services to be generated on the fly. Because of distributed responsibilities, ownership, and control, often, it is not feasible to acquire all information needed for the service composition, thus there might be no guarantee that the service execution has an anticipated effect. In this paper, we are going to extend current Semantic Web Service Description by introducing the concept of "Service Assumption" which allows reasoning with incomplete information. Furthermore, together with the proposed service assumption, a sequence of rules is developed to describe all permitted behaviors in service composition context.

1 Introduction

OWL Web Ontology Language for Services (OWL-S [1]) leverages the rich expressive power of OWL [4] together with its well-defined semantics to provide richer descriptions of Web Services. Service ontologies can be used to map service functional descriptions and domain properties into a standardized logic so that they can be machine understandable and interpretable. Recently, semantic web rules language (SWRL) [2] has been proposed to define service precondition and effect, process control conditions and their contingent relationships in OWL-S. Though OWL-S is endowed with more expressive power and reasoning options when combined with SWRL, the description provided by a combination of OWL-S and SWRL about service composition is still only a partial picture of the real world. Most of what we know about the world, when formalized, will yield an incomplete theory precisely because we cannot know everything - there are gaps in our knowledge [7]. The ontology of services, on the other hand, is finite and incomplete. Thus, a service composition specified by OWL-S has to deal with partial or incomplete knowledge. The inability to deal with incomplete knowledge may translate into an inability to deal with exceptions

In this paper, we are going to bridge the gap between the semantic service description and multiple operational domains involved by introducing the concept of "service assumptions". Currently, OWL-S has no mechanism for handling the explicit description of service assumptions and no method for reasoning about their side-effects. We will extend the current OWL-S and try to define a formal mechanism to reason about incomplete knowledge in the dynamic service composition context.

The paper is structured as follows: in Section 2, we explain the semantics of an extended atomic service and a composite service in general. In Section 3, we define the basic semantics for planning-based service composition domain. Then in Section 4, we present a framework for reasoning about incomplete knowledge in service composition context. Finally, we present our conclusions in Section 5.

2 Atomic Service and Service Selection

2.1 Atomic Service

Currently, there is no way for OWL-S to describe the various assumptions about the multiple independent application domains involved in service composition. By adopting the service assumptions into OWL-S, we can conduct reasoning about what is known in the composite service execution context against various domain assumptions. Thus the ontology for Web service becomes more complete and closer to the real world. We propose to add the assumptions as the properties of service process, which allows reasoning with incomplete information.

An atomic service only performs a single function. An extended atomic service ws is described by a tuple $ws = \langle p, e, a \rangle$, where p represents service precondition which must be true for the service ws to be invoked, e represents the change of the world state, i.e. the effect after service ws completes and a represents the service assumptions. Note that p and a are different. It must be able to establish that

IEEE International Conference on Services Computing (SCC'06)
0-7695-2670-5/06 $20.00 © 2006 IEEE

p is true for ws to be invoked. On the other hand, we only need to establish that a is consistent with what is known, i.e. nothing is known that contradicts a. p is the strong condition which must be true in order to execute the service ws, while a is a weak condition. Initially we assume a to be true, unless we get additional information which is explicitly contradictory to a.

2.2 Service Selection

The process of dynamic Web Service composition over that of software component composition holds some additional critical issues, such as service matching, selection and retrieval. In this proposed framework,

- ws_i represents an atomic service.
- WS is the set of all Web Services, $ws_i \in WS$.
- all Web Service descriptions are held in their corresponding categories $\{cat_1, cat_2, \ldots, cat_n\}$. cat_i is a tangible areas split from the service registry, for example downloadable Multimedia.
- CAT is the set of all service categories $cat_i \in CAT$, $cat_i \in WS$, $cat_i = \{ws_1, \ldots, ws_m\}$.
- Service selection function $sel : CAT \rightarrow WS$ which takes a certain service category as its input and give us an atomic service based on the service matching i.e. $sel(cat_i) = ws$.

Every atomic service in the rest of this paper refers to the Web Service which is produced by the service selection defined above.

2.3 Composite Service

Intuitively, a composite Web Service which performs combined functions may include multiple atomic services. A composite service $CompWS$ is the combination of the multiple atomic services ws_i , where $0 < i < n$. $CompWS$ can be represented as:

$$CompWS = \{sel(cat_1), \ldots, sel(cat_n)\}$$

Because participants of the service composition do not necessarily share the same objectives and background, conflicts easily arise in a dynamic service composition environment.

3 Service Composition as Planning

It is often assumed that a business process or application is associated with some explicit business goal definition that can guide a planning-based composition tool to select the right service [6]. Typically, classical planners presuppose complete and correct information about the world. However, in terms of the service composition, this simplified assumption is not suitable and unrealistic. Each service node is designed, owned, or operated by different parties, thus the planning agent may not have a complete view about the world. To make more precise service description in a dynamic service composition environment, we have extended the current semantic Web Service description OWL-S by introducing the service assumption. The service assumptions together with states of knowledge, preconditions, effects, and goals are specified in Description Logic $\mathcal{L}$ [3].

Now we are prepared to define the semantics of a service composition domain. A state S is a snapshot which describes the partial state with respect to the service composition context. The state S is extensionally defined as a set of positive or negative ground atomic formulas (atoms). In addition, the initial state S_0 here is a partial description about the world, i.e. a partial state. A goal G is a set of conjunctions of atoms which need to hold in a desired state or say final state. A state transition t is represented as a tuple $t = \langle S, ws, S' \rangle$, where S, S' are states and ws is an atomic service. A service composition plan for a goal is a sequence of state transitions which lead from an initial state to a final state where all ground atomic formulas in the goal are true. In rest of the paper, we will use symbol $\models$ to represent logical entail.

In the process of service composition planning, there are three types of knowledge produced by state transitions about the current world. Let SEN_i denote a set of sentences used to change the state S_i. This set of sentences can be partitioned into three categories, namely state invariants, expansion and update, which is defined as:

$$SEN_i = \{Inv_i \mid Exp_i \mid Upd_i\}$$

1. State invariant Inv_i denotes a set of sentences which can be entailed by the knowledge in the previous state, defined as: $S_{i-1} \models Inv_i$
2. State expansion Exp_i denotes a set of sentences which cannot be entailed by the knowledge in the previous state and its negation also cannot be entailed by the knowledge in the previous state, defined as:

$$S_{i-1} \not\models Exp_i \text{ and } S_{i-1} \not\models \neg Exp_i$$

3. State update Upd_i denotes a set of sentences whose negation can be entailed by the knowledge in the previous state, defined as: $S_{i-1} \models \neg Upd_i$

Let ws_i be an atomic service, WS be the set of all Web Services, E be the set of all service effects, P be the set of all service preconditions, we define the following extraction functions:

1. Effect extraction function $f_e : WS \rightarrow E$ which takes an arbitrary atomic service ws_i as an input, and extracts the effect e_i of ws_i as its output. e_i is a set of primitive effects of ws_i and every primitive effect is a partition with the state invariant, expansion and update, i.e. $f_e(ws_i) = e_i$ and $e_i = \{eInv_i \mid eExp_i \mid eUpd_i\}$ in which $eInv_i, eExp_i, eUpd_i$ denote state invariant, expansion and update respectively.

2. Precondition extraction function $f_p : WS \rightarrow P$ which takes arbitrary atomic service ws_i as an input, and extracts the precondition p_i of ws_i as its output, i.e. $f_p(ws_i) = p_i$. Similar to the effect extraction function: $p_i = \{pInv_i \mid pExp_i \mid pUpd_i\}$.

Following the definitions we gave above, we can define the generic state transition operator as:

$$S_i = \Delta(pUpd_i, eUpd_i, S_{i-1}) + eExp_i + pExp_i$$

which means the state transition from S_{i-1} to S_i is completed by means of applying $pUpd_i$ and $eUpd_i$ to the state S_{i-1} orderly, then add the two types of expansion of knowledge $(eExp_i, pExp_i)$ to the state S_{i-1} . Note that the order of applying state update must be strictly followed.

4 Defeasible Reasoning in Service Composition

Typically, a dynamic service composition does not have a predefined boundary, based on which the problems of uncertainty and incompleteness of information could be resolved. A changing environment complicates dynamic service composition in many ways. Inspired by Non-monotonic logic [5], the following subsection will attempt to provide a formal framework for reasoning about incomplete knowledge in service composition context.

4.1 Defeasible Reasoning Framework

In this work, the reasoning about incomplete information in the process of service composition planning works with three kinds of rules, namely absolute rule, defeasible rule and defeater [5]. The absolute rule which is interpreted in the classical sense means whenever the premises are indisputable then so is the conclusion. On the other hand, the defeasible rule is that whose conclusion is normally true when its premises are, but certain conclusions may be defeated in the face of new information. Defeasible rules can be defeated by contrary evidence or by defeaters. Defeaters represent knowledge which is to prevent defeasible inference from taking place. We use the operator $\Rightarrow$ for the absolute rule, $\sim>$ for the defeasible rule and $\mapsto$ for the defeater.

Let ws_i represent an atomic service which is produced by the service selection function (see section 2.2), a_i represent the assumption of ws_i, p_i represent the precondition of ws_i, e_i represent the effect of ws_i and $isValid(ws_i)$ represent the atomic service whose preconditions can be satisfied. Inspired by Nute's defeasible reasoning [5],

Absolute rule: $p_i \Rightarrow isValid(ws_i)$ (Rule A)
which means only precondition holds, and then the service is a valid candidate service to participate in service composition.

Defeasible rule: $isValid(ws_i) \sim> e_i$ (Rule B)
which means normally e_i is the conclusion of a valid candidate service ws_i, but this e_i may be defeated in the face of new information.

Defeater: $\neg a_i \mapsto \neg e_i$ (Rule C)
which means given an assumption a_i, if the negation of the assumption is entailed by a given state of knowledge, it will prevent the Rule B from making the conclusion e_i.

4.2 Outdated Assumptions and Assumption Database

If the negation of all sentences in e_i is entailed by any state S_j, where e_i is the effect of an atomic service ws_i and $j > i$, (i.e $\forall x \in e_i, \exists j > i$ *such that* $\neg x \in Cn(S_j)$, where $Cn(S_j)$ denotes logical closure of S_j), then the assumption a_i associated with e_i is called the **outdated assumption**. The outdated assumptions are not allowed to participate in defeasible reasoning. A simple example of an outdated assumption is: a book borrowing service assumes that the borrower is in same city as the library. When the borrowed book is returned, we say this assumption is outdated.

To conduct the defeasible reasoning about the partial state of knowledge, it is necessary to describe and record various assumptions generated during the service composition planning. In this framework, we maintain an assumption database D_α to store these assumptions and their relevant effects as a pair $\langle a_i : e_i \rangle$. Same as preconditions and effects, assumptions are represented as ground literals.

4.3 Defeasible Reasoning Process

Service composition planning can be viewed as a process of resolving conflicts and gradually refining a partially specified plan, until it is transformed into a complete plan that satisfies the goal. Service composition planning is similar to the classical planning in that each state is represented by a conjunction of literals and each Web Service is related to a transition between those states. However, unlike classical AI planning techniques, in this proposed framework, the planner is the rule based system which allows making tentative conclusions and revising them in the face of additional information. In other words, the planner is endowed

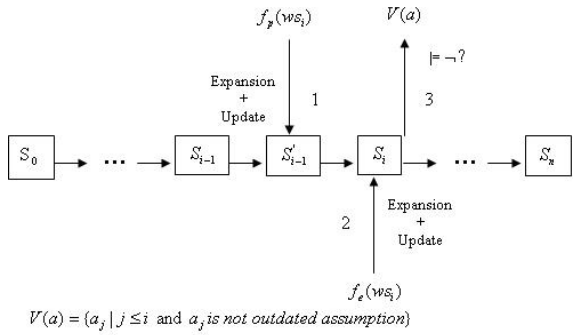

Figure 1. Defeasible Reasoning Process

with the ability to reason about incomplete information in the service composition context.

For any state S_{i-1}, Web Service ws_i is not applicable to the state until certain minimal criteria are met. ws_i is specified in terms of the precondition p_i, effect e_i and assumption a_i, where p_i must be satisfied for ws_i to be valid (Rule A), the effect may be concluded (Rule B) and the negation of a_i plays the role of being the defeaters(Rule C).

A state in our framework is not a complete view of the world. Usually, an agent is forced to perform sensing operations which is aiming at finding out the information which could satisfy the precondition p_i. Like "1" showed in the fig 1, the sensing operation may lead to knowledge expansion and update of the state S_{i-1}. When the sensing operations complete, if p_i is satisfied, we can conclude that ws_i is applicable to the current state (Rule A). Due to the expansion and update of knowledge to state S_{i-1}, before the transition to state S_i, we get an intermediate state S'_{i-1} which holds the current knowledge of the state after the agent's sensing operation. Following the sensing operations, effect e_i is applied to the current state to simulate the action (Rule B). Again, the effect e_i may expand and update the knowledge of the current state. This process can be presented as the generic state transition operation as we defined in Section 3.

One of the main features in this proposed framework is the ability to describe various service assumptions and support defeasible reasoning with these assumptions. The assumptions generated from the service composition planning are represented as a set of ground literals stored in the assumption database D_α. After the effect is applied to the current state, the knowledge in the state may be expanded or updated. For the new state of knowledge, the planner needs to carefully perform the checking to see whether any outdated assumption is in D_α. Because the outdated assumptions are not allowed to participate the defeasible reasoning, all outdated assumptions are deleted from D_α. Next, it is to find the defeaters by the mean of checking whether any negation of assumptions can be entailed by the current state. The negation of service assumption which is not outdated plays the role of being a defeater, which prevents the effect associated with this service assumption being applied to the state (Rule C). Up to now, the process of state transition from S_{i-1} to S_i is completed. We have illustrated that how the new state is reached in the presence of possibly conflicting rules.

5 Conclusions

The goal of dealing with incomplete information in the service composition context is certainly a challenging task. The proposed framework is an attempt at tackling the problem of how to achieve consistent service composition when information available is insufficient.

In this work, we have extended the OWL-S to a richer service description representation schema by introducing the concept of service assumption. We also adopted defeasible rules for reasoning with various assumptions. We illustrated how knowledge based planning could reason about incomplete knowledge in service composition context and construct the service composition plan. During the process of the service composition, we showed that absolute rules could be used for service precondition satisfaction, especially defeasible rules and defeaters could be employed to make tentative conclusions based on the information at hand, and to detect potential conflicts when further information about the problem is available.

References

[1] OWL Services Coalition. OWL-S: Semantic markup for Web Services, OWL-S White Paper. 2005. http://www.daml.org/services/owl-s/1.1/overview/#1

[2] "Semantic Web Rule Language", May 21, 2004 http://www.w3.org/Submission/2004/03/.

[3] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F., Eds. The Description Logic Handbook:Theory, Implementation and Applications. Cambridge University Press. 2003.

[4] Dean, M. and Schreiber G. "OWL Web Ontology Language". Reference W3C Recommendation, http://www.w3.org/tr/owl-ref/. Feb 2004.

[5] Donald Nute. "Defeasible logic." In D. Gabbay and C. Hogger (eds.), Handbook of Logic for Artificial Intelligence and Logic Programming, Oxford University Press, 1994. Vol. III:353-395.

[6] S. McIlraith and T. C. Son. Adapting Golog for composition of Semantic Web Services. In Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (KR2002), Toulouse, France, April 2002.

[7] Reiter R. "ON REASONING BY DEFAULT", Proceedings of the theoretical issues in natural language processing-2 July 1978