Reasoning on Context Satisfiability for Service Recommendation in Mobile Network

Li Kuang¹, Liang Chen², Yingjie Xia¹
¹Hangzhou Institute of Service Engineering
Hangzhou Normal University, China
²College of Computer Science
Zhejiang University, China
{kuangli,cliang,xiayingjie}@zju.edu.cn

ABSTRACT: The Ubiquitous Computing envisions a world of communicating mobile devices. As mobile devices usually have limited resources, it brings up the challenge to search and invoke services at any time and any place in an accurate, efficient and adaptive way. In this paper, we propose to filter services by matchmaking the precondition of service invocation and the context information of service consumer, in order to increase the accuracy and adaptivity of service discovery and recommendation in mobile environment. We construct context ontology to describe the environment when a service consumer submits service request first. And then, we propose to formalize the precondition of a successful service invocation based on a lightweight DL theory. Finally, reasoning on context satisfiability can be used for service recommendation. A case study shows our approach works well.

Categories and Subject Descriptors
C.1.4 [Parallel Architectures]: Mobile processors; E.4 [Coding and Information Theory]: Formal models of communication

General Terms
Service Oriented Computing, Mobile Network, Context Ontology

Keywords: Context Satisfiability, Precondition Formalization, Service Reasoning, Service Recommendation

Received: 18 July 2011, Revised 1 September 2011, Accepted 12 September 2011

1. Introduction

In recent years, Service Oriented Computing [1] has emerged as a highly-promising paradigm for distributed computing and software engineering, changing the way that software applications are designed, delivered and consumed. As a result, a lot of Web services have been published on the Internet, providing consumers various functionalities in a language independent, platform independent and protocol independent manner. A subsequent challenge is to locate target services in a quick, accurate and efficient way.

The challenge becomes especially pressing with the increasing popularity of smart-phones and tablet devices like the iPad, along with ubiquitous network connectivity. Nowadays people tend to search and invoke services with mobile devices, requiring helpful, accurate and real-time feedback from services at any time and any place. However, mobile devices usually have limited display, processing, storage, power and communication resources, therefore it requires more accurate and adaptive recommendation list of services for a service query on mobile devices.

A promising way to increase the accuracy and adaptivity of service discovery and recommendation, which is often neglected, is to filter services by matching the precondition of service execution and the context information of the service query request. For instance, a service consumer wants to query a bus line from location A to B, common service discovery approaches would return a bus list without considering the service time of the bus line. Since there are time of early bus and that of last bus for a bus line, suppose the last run of a bus in the returned list is 8:00pm, while the time when the consumer submits the request is 9:00pm, hence if the precondition satisfiability of the bus service is considered, the bus is unavailable for the consumer and should not be returned. Take another example, a service consumer searches a translation service to translate a text from Chinese to Spanish, common service discovery approaches would return a service list by just matching the input and output, which are both a kind of language, however a service in the list may not support the translation from the source language Chinese to the target language Spanish. In a word, it is important to take the precondition satisfiability of a service into account to increase the accuracy and adaptivity of service discovery and recommendation.

In this paper, we propose to construct context ontology first, so that we can determine what context information of service consumers will be matched against service information for service recommendation, and then we propose to formalize the precondition of successful service invocation based on a lightweight Description Language,
and finally based on the formalization of both context of service consumers and precondition of service invocation, we can reason on context satisfiability for service recommendation in mobile network.

The paper is organized as follows. Section 2 reviews the related work on the formalization and reasoning of web services. Section 3 introduces the context ontology that defines the context information of a service consumer. Section 4 introduces the formalization of precondition under which a service can be invoked successfully. Section 5 proposes two kinds of reasoning problem on context satisfiability. And finally the conclusion and future work is given in Section 6.

2. Related Work

Although research on context aware service discovery and recommendation started in early days, matchmaking between user’s context and a service description is still an important issue especially for mobile network. Our work presented in this paper is related to existing research on representation, modeling and context aware query of Web services. We therefore comment on some of these in the following.

2.1 Formalization and Reasoning of Web services

The most popular formal theories used for the formalization of Web services include Description Language [2], automata [3], Petri Net [4] and process algebra [5]. The latter three theories are appropriate to express the internal behavior of a web service, while DL is more appropriate to express the interface semantic of a web service, especially the properties, constraints and relations of the ontology referenced by service interface. Hence we concentrate on the related work on DL-based service formalization and reasoning.

DL is a formalized language for knowledge representation. Based on atomic concepts and relations, it can construct more complex concepts and relations by constructs such as number restriction, inverse relation and composite relation. There are two main components in DL: TBox and ABox. TBox contain a set of axioms for concepts and relations, while ABox contain a set of instances of concepts and relations. Researchers [6-16] proposed to model the interface semantics of services by DL and its extension such as DDL [17]. DL and DDL provides powerful reasoning mechanism such as the consistency, satisfiability, inclusion relation between axioms and instances, therefore it can support automatic and intelligent service discovery and composition. However, in most existing work, they simply model the precondition of a successful service invocation as an ABox assertion, in the form of a function name with some parameters. As the authors in [18] analyzed, the precondition satisfiability plays an important role in service discovery and composition, but current work does not provide solutions on how precondition will be modeled. Hence in this paper we detailed on how precondition is modeled based on a lightweight DL. The application of simplified DL alleviates the complex of reasoning on precondition as well, which is very suitable for service recommendation in mobile network.

2.2 Context aware service discovery

Presently there is much work on service discovery based on the matchmaking of interface, behavior and QoS. Recently, context matching has been applied to service discovery. Generally speaking, context is all the information that can describe the environment a service consumer is in [19].

Feng et al [20] constructed user’s context ontology and service context ontology first, proposed three kinds of inference rules between user and service context, which are rules for filtering, priority selection and user’s preference, and implemented corresponding context inference algorithms. Yang et al [21] proposed to infer the implied context of users based on directly perceptive context information first, and then find appropriate services based on the matchmaking between service descriptions and the implied user’s context. [22] proposed a new context query language to describe complex constrains on context. Doulkerides et al [23] proposed a context aware service discovery system in which context information is represented as key-value pairs and matched against service interfaces. Bormann et al [24] proposed to find the most appropriate network operator before making phone calls based on user’s location and call time. There are also some approaches on extending service discovery protocols in mobile computing with support of context awareness [25-26].

Compared with the work above, in this paper, we propose a relatively complete ontology to describe user’s context when he submits his service request. The context ontology explicitly determines what information can fully describe the environment a user is in and what information will be matched against service description. Second, existing work neglected the fulfillment of precondition of a successful service invocation, and here we concentrate on the matchmaking between context of service consumers and precondition of service invocation, so that quick, accurate and efficient service recommendation can be achieved.

3. Context Ontology

In order to recommend services based on context reasoning in mobile network, we need to identify what context means first. In this section, we construct context ontology so that we can determine what context information of service consumers will be matched against service information for service recommendation in the following.

In general, context is all the information that can represent the activities of a user. As Figure 1 shows, we specialize the concept of context into an ontology consisting of two parts: user profile context, describing the intrinsic information of a user, which is irrelevant to current service invocation; and service invocation context, describing all the instant information relevant to current service invocation.
User profile context can be further divided into basic information and personal preferences. Basic information characterizes the static features of a user, such as gender, date of birth, specialty, and social role. Personal preferences characterizes implicit priority rules when the user chooses services, for instance, one wants a service with short response time although it takes more price.

Service invocation context can be further divided into three parts: (1) purpose of service invocation, which is the potential service invocation instances for current user. For example, a user wants to find a translation service that can translate Italian into Korean, such request will be translated into finding a service whose input and output signatures are both of a language, while ignoring the real use of the service for current user, that is, to translate from Italian to Korean, and we have analyzed how the ignorance of such information affects the result of service discovery in Section 1, hence we emphasize such context information as the purpose of service invocation; (2) Natural environment, which describes the local environment of the user when he submits the service request, such as the location where he is, the time when he requests a service, the date (whether it is a workday or a holiday); (3) Computing environment, which describes the computing environment of the user that is closely related to the software and hardware requirements to invoke a service. For example, the hardware device the user uses for service invocation, including the model, the operating system, the memory, CPU of the device and so on, the network where the user invokes a service, such as network delay, bandwidth, security level, and access pattern.

Example 3-1. On 1st Aug 2011, a user wants to translate some texts in Chinese into Spanish by using an online translating service. He submits the service request in Hangzhou by his mobile phone. The model of the mobile phone is HTC G7 with operating system android 2.2, CPU QSD8250 1GHz, 576MB RAM and 512MB ROM. The following segment is an example of the description of user's context information.

```xml
<context>
  <user profile>
    ...
  </user profile>
  <service invocation>
    <purpose>
      <input>Chinese</input>
      <output>Spanish</output>
    </purpose>
    <Natural Environment>
      <location>Hangzhou</location>
    </Natural Environment>
  </service invocation>
  <date>1st Aug 2011</date>
  ...
  <Network>
    ...
  </Network>
  <Computing Environment>
    <Hardware Device>
      <Model>HTC G7</Model>
      <Operating System>Android 2.2</Operating System>
      <CPU>QSD8250 1GHz</CPU>
      <Memory>576MB RAM and 512MB ROM</Memory>
    </Hardware Device>
    ...
  </Computing Environment>
</context>
```

Figure 1. Context Ontology
4. Modelling Precondition Expressions

Service description standards such as WSDL-S and OWL-S do not support reasoning on service precondition satisfiability itself, although they may provide the framework to describe the preconditions that a service can be invoked successfully. Hence we need to formalize preconditions.

Definition 1. (Basic Symbol) The basic symbols of DL can be divided into several parts:

1. a set of concept names \( N_c \);
2. a set of role names \( N_r \);
3. a set of individual names \( N_i \);
4. a concrete domain \( D \) consisting of a set of data, where each element is associated with a data type and a value;
5. concept constructs \( \{ \} \) (enumeration), \( \leq \) (max number restriction), \( \geq \) (min number restriction), \( \forall \) (any value restriction), \( \exists \) (value existence restriction), \( \cup \) (union), \( \wedge \) (intersection) and \( \neg \) (negative);
6. role construct \( \neg \) (inverse), \( \cup \) (union), \( \wedge \) (intersection);
7. formula constructs \( \neg \) (negative), \( \land \) (and), \( \lor \) (or).

Definition 2. (Role) \( R,T \) are roles in the following form:
\[
R,T ::= \forall r \mid r^\neg \mid R \cup T \mid R \cap T
\]
where \( r \in N_r \). Roles can be divided into abstract roles and concrete roles. Abstract roles can be constructed by concrete roles and other abstract roles using \( \cup \) and \( \cap \). \( r^\neg \) represents the inverse of the role \( r \). The inclusion relations between two roles and the inverse of a role are defined in the RBox.

Definition 3. (Concept) \( C,S \) are concepts in the following form:
\[
C,S ::= c \mid \{ u \} \mid \leq n S \mid \geq n S \mid \forall S.C \mid \forall S.C \mid \exists S.C \mid \neg S \mid S \cup C \mid S \cap C
\]
where \( c \in N_c \), \( u \in N_i \), and \( n \) can be any number. Hence a concept can be a concept name \( c \), an individual instance \( \{ u \} \), a number restriction on another concept \( S \) denoted by \( \leq n S \) or \( \geq n S \) which means the defined concept is composed of/has at least \( n \) copies of \( S \) or at most \( n \) copies of \( S \). A type restriction denoted by \( \forall S.C \) which means any \( S \) must be of the type \( C \), or \( \exists S.C \) which means there exists some \( S \) of the type \( C \), the inverse of another concept \( S \) denoted by \( \neg S \). And concepts can be connected by union symbol \( \cup \) or intersection symbol \( \cap \).

A finite set of concept definitions with unique left-hand sides constitute TBox \( T \). Concepts are called defined in \( T \) if concept names occur in the left-hand side of a definition in \( T \), otherwise concepts are called primitive in \( T \). And we set TBox \( T \) acyclic, i.e. no cyclic dependencies between concepts are not allowed.

Definition 4. (Formula) \( \psi, \varphi \) are formulae in the following form:
\[
\psi, \varphi ::= \psi C(u) \mid \psi C(v) \mid C(v) \in (v_1, \ldots, v_m) \mid \neg \psi \mid \neg C(u) \mid \psi \mid \varphi \mid \psi \lor \varphi
\]
where \( u,v,v_1,\ldots,v_m \in N_r \). \( C(u) \) and \( C(v) \) are concepts instantiated by individual \( u \) and \( v \) respectively, \( \psi \) can be \( =, \neq, \leq, \geq, <, >, \) and \( R \) is a relation function between \( u \) and \( v \). Hence, an atomic formula can be an arithmetic comparison between concepts denoted by \( C(u) \mid C(v) \); a test for set membership denoted by \( C(u) \mid C(v) \in \{ v_1, \ldots, v_m \} \) where \( v_i \) is a continuity interval and \( \{ v_1, \ldots, v_m \} \) is a discrete set; a set comparison denoted by \( C(u) \mid C(v) \mid \{ v_1, \ldots, v_m \} \), where \( C(u) \mid C(v) \) some \( \{ v_1, \ldots, v_m \} \) is equal to true when there exists some \( v \in \{ v_1, \ldots, v_m \} \) that \( C(u) \mid C(v) \) is true, while \( C(u) \mid C(v) \) all \( \{ v_1, \ldots, v_m \} \) is equal to true when for all \( v \in \{ v_1, \ldots, v_m \} \), \( C(u) \mid C(v) \) is true; and a user defined role relation \( R \) between individual \( u \) and \( v \). Formula can be composed of atomic formulae in the above form and formula constructs. \( \psi \lor \varphi \) is true when \( \psi \) is not true. \( \psi \varphi \) is true when \( \psi \) and \( \varphi \) are both true. \( \psi \lor \varphi \) is true when either \( \psi \) or \( \varphi \) is true.

Definition 5. (Precondition) \( P = \{ \varphi_1, \ldots, \varphi_m \} \) represents the preconditions that must be fulfilled before a service can be executed successfully, where \( \varphi_i | 1 \leq i \leq n \) is a formula.

Example 4-1. The online BabelFish translator services can convert text from one language to another language. This service has three inputs: the text to be translated, its language and the desired output language. There are total of nine languages supported by the translator. The precondition of the service requires that the input language and output language should be different.

\[
LanguagePair = \{ (\geq 1 \text{ source}) \cap (\forall \text{ source.Language}) \cap (\geq 1 \text{ target}) \cap (\forall \text{ target.Language}) \}
\]

\[
SupportedLanguagePair = \{ (\text{English, Chinese}), (\text{Chinese, English}), (\text{Korean, Chinese}), (\text{Chinese, Korean}), (\text{Spanish, English}), (\text{English, Spanish}), (\text{French, Italian}), (\text{Italian, French}) \}
\]

\[
LanguagePair(source,target) \text{ in SupportedLanguagePair}
\]

5. Reasoning On Precondition Satisfiability

Based on the formalization of both context of service consumers and precondition of service execution, we can reason on the precondition satisfiability for service recommendation.

The reasoning problem can be divided into two kinds: the executability of a service instance and the realizability of a web service.

For a list of candidate web services, we need to identify whether the services can be executed successfully under the context of current consumer. The reasoning problem is defined as the executability of a service instance, which can be implemented as Definition 6 states.

Definition 6. (Executability of a service instance) Let \( CT_u = \{ T_1 = v_1, T_2 = v_2, \ldots, T_m = v_m \} \) be the context description of a service consumer, in which \( T_i \mid i = 1,\ldots,n \) are properties belonged to the context ontology and \( v_i \mid i = 1,\ldots,n \) are instance values, let \( P_s = \{ \varphi_i, \ldots, \varphi_m \} \) be the precondition of...
successful execution of service \( s \). For all \( \phi_j \mid j=1...m \) with arguments \( \alpha_1, ..., \alpha_k \), i.e., \( \phi_j(v_i/\alpha_{j1}, ..., v_i/\alpha_{jk}) \), and appropriate mappings between \( T_i \mid i =1...n \) and \( \alpha_j \mid j=1...k \), if the substitution of corresponding values into \( \alpha_j \mid j=1...k \), i.e., can be inferred to be true, then the service \( s \) is said to be executable under the context \( CT_u \) of current user \( u \).

Example 5-1. Identify whether the service \( s \) described in Example 4-1 can be invoked successfully by user under the context \( u \) stated in Example 3-1, i.e., the executability of service instance in Example 4-1 under context in Example 3-1.

\[
\begin{align*}
\text{u.context/service invocation/purpose/input} & \iff \text{s.Language Pair.source} \\
\text{u.context/service invocation/purpose/output} & \iff \text{s.Language Pair.target} \\
\text{LanguagePair(Chinese/source, Spanish/target) in Supported LanguagePair} & = (\text{Chinese, Spanish}) \\
& \in \{\text{(English, Chinese)}, \text{(Chinese, English)}, \\
& \text{(Korean, Chinese)}, \text{(Chinese, Korean)}, \text{(Spanish, English)}, \\
& \text{(English, Spanish)}, \text{(French, Italian)}, \text{(Italian, French)}\} \\
\Rightarrow & \text{False}.
\end{align*}
\]

Hence we can conclude that the service \( s \) in Example 4-1 cannot be invoked successfully by user under context \( u \) in Example 3-1.

The second kind of reasoning is the realizability of a web service. For any web service, we need to identify whether there exists some state under which the web service can be executed successfully. If the service cannot be executed under any state, such service is meaningless. It can be implemented as Definition 7 states.

Definition 7. (Realizability of a web service) Let \( P \) be the precondition of successful execution of service \( s \), let \( \{CT_{w1}^m \cdots CT_{wm}^m \} \) be the context of service consumers who have tried to invoke \( s \). If there exists some \( CT_{w}^m \) that \( CT_{w}^m \models P \) can be inferred to be true, then the service \( s \) is said to be realizable, otherwise, \( s \) is not.

Example 5-2. Identify whether the service \( s \) described in Example 4-1 is realizable.

\[
\begin{align*}
\text{u.context/service invocation/purpose/input} & \iff \text{s.Language Pair.source} \\
\text{u.context/service invocation/purpose/output} & \iff \text{s.Language Pair.target} \\
\text{If u.context/service invocation/purpose/input} & = \text{English and} \\
\text{u.context/service invocation/purpose/output} & = \text{Chinese, then} \\
\text{LanguagePair(English/source, Chinese/target) in Supported LanguagePair} & \Rightarrow \text{True}.
\end{align*}
\]

And it is true for all the source and target pairs that belong to SupportedLanguagePair.

Hence we can conclude that there exists some context under which the precondition of \( s \) is true and \( s \) is realizable.

6. Conclusion And Future Work

In this paper, we propose to reason between service precondition and user’s context information based on a lightweight DL in order to achieve accurate and efficient service recommendation in mobile network.

Our main contributions include: First, we construct context ontology and clearly define what properties should be described to express the environment when a user submits service request. Second, we detailed the formalization of service precondition based on a simplified DL. And third, we reason about the executability of a service instance and the realizability of a web service by inferring the satisfiability of service precondition under user’s context.

In the future, we plan to develop tools that can formalize and reason on user’s context and service precondition automatically, as well as conduct large scale of experiments for application of context aware service recommendation in mobile network.

7. Acknowledgement

The work is supported in part by the following funds: Zhejiang Provincial Natural Science Foundation of China under grant number Y1110591, Hangzhou Normal University under grant number 2010HSKQ0006, National Natural Science Foundation of China under grant number 61002009, Science and Technology Planning Project of Zhejiang Province under grant number 2010C31018.

References


