Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 403 – 407 Sept. 2007 ISSN 1003—7985

Heuristic web service composition method based on domain ontology

Ren Hongbo[1] Xing Chunxiao[2]

([1]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

([2]Research Institute of Information Technology, Tsinghua University, Beijing 100084, China)

Abstract: In order to realize automatic web service composition, a heuristic web service composition method based on domain ontology is proposed. First, this method integrates the domain ontology and the artificial intelligence (AI) planning algorithm. Then, it uses the domain ontology and its reasoning capability to infer the semantic relationship among parameters. Finally, it transforms the web service composition problem into the planning problem based on the AI planning heuristic algorithm. The preliminary experimental results show that the above method compensates for the lack of semantics in the previous AI planning method and it can satisfy the requirements of quality and efficiency of composition, thus generating composite web services according to customer requirements automatically and efficiently.

Key words: domain ontology; semantic; web service; service composition; heuristic

With the rapid development of the semantic web, web services which are considered one of the most important blocks of the semantic web, have aroused great interest in both academia and industry all around the world. Web services technology has incorporated the strengths of distributed computing, grid computing and XML, etc. Through adopting XML-based specifications such as WSDL, UDDI and SOAP, web services provide interoperability among diverse applications and platform- and language-independent interfaces for easily integrating heterogeneous systems.

There has been a great deal of research in the area of web services in the past few years. A large part of these studies has been dedicated to web services composition. It may often be the case that a web service does not provide a requested service on its own, but delegates parts of the execution to other web services and receives the results from them to perform the whole service. However, as the existing specifications and protocols can only describe services on the grammar level but not the semantic level, most composition methods are still not automatic.

Semantics use a machine-understandable way to describe the function and flow of web services in order to realize the automatic discovery, selection and composition of web services. And through the exact description of the concepts and relations between the concepts in a specific domain, domain ontology becomes the semantic foundation based on which human-to-machine and machine-to-machine can be understandable to each other. In this paper, we are interested in studying how to realize automatic web service composition based on domain ontology. We propose a new automatic composition method—HWSCO. The novelty of this method is to combine the domain ontology and the AI planning based algorithm to cover the factors such as service semantics, composition quality and composition efficiency.

The core of the method is to consider the web service composition problem as a searching problem of the state space. This paper proposes a heuristic algorithm which can fully use the semantic information of the inputs and outputs of web services provided by the domain ontology and its inference ability. Selecting subsequent web services according to the relevancy maximum principle enables us to reduce the complexity of the algorithm, and finish the searching in state space efficiently.

1 Basic Definitions

Domain ontology is the specification of the concepts and the relationship among them is specific domain.

Definition 1 Domain ontology can be described by a five tuple:

$$O = \{C, R, H^{C}, \text{rel}, A^{O}\}$$

where C is a set of concepts; R is a set of relationships; H^{C} is the level of concepts; rel is the relationship among concepts; A^{O} is the ontology axiom.

Received 2007-05-18.

Foundation items: The National Natural Science Foundation of China (No. 60473078), the National High Technology Research and Development Program of China (863 Program) (No. 2006AA010101).

Biographies: Ren Hongbo (1983—), female, graduate; Xing Chunxiao (corresponding author), male, doctor, professor, xingcx@tsinghua.edu.cn.

There are many languages to describe the ontology. We adopt the recent recommended standard OWL-S as the ontology description language. The development of OWL-S is dedicated to enabling the automatic discovery, selection, composition and execution of web services. OWL-S describes web services using three parts: service profile, service model and service grounding. In this paper, we mainly use the first part, which includes the name, description, inputs, outputs, preconditions and effects of a web service.

Based on OWL-S, the following definitions are given.

Definition 2 In domain ontology, there are two concepts: C_i and C_j. If C_i and C_j have the same semantic information, then $C_i \equiv C_j$; if the semantic information of concept C_i includes that of C_j, then $C_i \supseteq C_j$.

When solving the web service composition problem as a planning problem, the problem can be represented as a STRIPTS model of a four tuple $\prod = \langle P, W, r^{\mathrm{i}}, r^{\mathrm{o}} \rangle$, where P is a set of parameters; W is a set of web services; $r^{\mathrm{i}} \subseteq P$ is the initial input parameters; $r^{\mathrm{o}} \subseteq P$ is the desired output parameters.

A STRIPTS model defines a state space $\psi = \langle S, s_0, S_{\mathrm{G}}, \Omega(.), f, c \rangle$, where the states $s \in S$ are a collection of parameters in P; the initial state $s_0 \in S$ is such that $s_0 = r^{\mathrm{i}}$; the goal state $S_{\mathrm{G}} \in S$, is such that $r^{\mathrm{o}} \subseteq S_{\mathrm{G}}$; $\Omega(s)$ the set of web services $w \in W$ such that $w^{\mathrm{i}} \subseteq s$. That is, in the state s, w can be invoked; the transition function $f(w, s) = s'$ that maps a state s into another state s'' such that $s' = s \cup w^{\mathrm{o}}$ for $w \in \Omega(s)$; $c(w)$ is the invocation cost of w. A solution of the state model is a finite sequence of web services: $w_1, w_2, \ldots, w_n$.

Definition 3 (web service composition) Suppose that a request r has initial input parameters r^{i} and desired output parameters r^{o}, then the web service composition problem is to find a finite sequence of web services, $w_1, w_2, \ldots, w_n$ such that① w_i can be invoked sequentially from 1 to n; ② $(r^{\mathrm{i}} \cup w_1^{\mathrm{o}} \cup \ldots \cup w_n^{\mathrm{o}}) \supseteq r^{\mathrm{o}}$; ③ The total cost $\sum_{i=1}^{n} c(w_i)$ is minimized.

Definition 4 (full matching) Suppose that a state $s \in S$ is given, web service $w_i \in \Omega(s)$. If for $w_2 \in W$, $w_1^{\mathrm{o}} \supseteq w_2^{\mathrm{i}}$, then w_1 can fully match w_2.

Definition 5 (partial matching) Suppose that a state $s \in S$ is given, let a web service $w_1 \in \Omega(s)$. If for $w_2 \in W$, w_1 cannot fully match w_2 but $w_1^{\mathrm{o}} \cap w_2^{\mathrm{i}} \neq \varnothing$, then w_1 can partially match w_2.

2 HWSCO Method

The ontology library is defined by specific domain specialists. The semantic descriptions of web services are defined through the ontology library. We introduce the ontology concepts into the STRIPS model, so P, r^{i} and r^{o} in the model are all ontology concept collections.

The interface of the web service is to pass the control information and data information among services. Academia has propounded a web service composition approach based on interfaces[1-3], which seeks matches between input and output parameters to realize the dynamic composition. However, parameters with different spellings may have the same semantic meaning. So solely relying on the syntactic (text-based) similarity of parameters for determining if two web services can be composed together is insufficient. Therefore, it is vital to find semantic relationships among parameters for interface matching.

The first step of the HWSCO method is to use domain ontology to describe the semantic meaning of the parameters of web services and to find all relevant entailments such as the class inheritance relation between two classes that may not be directly encoded in the subclass relationships using OWL inferencing rules. According to definition 2, there are equal and subsumption relationships between ontology concepts. When a service input subsumes the output, the output type is just a specialized version of the input type, so these services can still chain together. Suppose a service accepts an input of type address, which is defined in certain ontologies with the concept hierarchy(see Fig. 1). An output of a service would be compatible if it were of a type of address of another concept that is subsumed by address, such as hotel address.

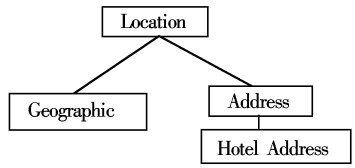

Fig. 1 Ontology hierarchy example

The match between the services whose output type is a subclass of the other service's input type is called a generic match. According to the domain ontology and inferencing rules, the inference engine[4] also orders the generic matches such that the priority of the matches are lowered when the distance between the two types in the ontology tree increases.

Section 1 presents the definitions of state space and other parameters. The next step is to consider the web service component problem as a search problem in state space, which finally satisfies the three requirements of definition 3.

The algorithm is divided into two parts: the for-

ward search and the regression search. First, we use the forward search to calculate the cost of achieving individual parameters starting from r^{i}; secondly, based on the first step, it approximates the optimal sequence of web services that connects r^{i} to r^{o} using regression search. The core of the algorithm is to use a novel heuristic method to reduce the number of web services during the solving process.

In the forward search, we define $k_{r^{\mathrm{i}}}(p)$ as the cost of achieving $p \in P$ from a state r^{i}. This cost can be characterized by the solution of the recursive equation as follows:

$$k_{r^{\mathrm{i}}}(p) = \min_{w \in O_w(p)} [c(w) + \max_{p' \in w^{\mathrm{i}}} k_{r^{\mathrm{i}}}(p')]$$

Suppose that the cost of executing a web service $c(w)$ is 1. $O_w(p)$ is a set of web services that $w \in W$ and $p \in w^{\mathrm{o}}$. First, if $p \in r^{\mathrm{i}}$, $k_{r^{\mathrm{i}}}(p)$ are initiated to 0 and to ∞ otherwise. $\forall w \in \Omega(s)$, each parameter $p \in w^{\mathrm{o}}$ is added to s and $k_{r^{\mathrm{i}}}(p)$ is updated until for $\forall p \in r^{\mathrm{o}}$, $k_{r^{\mathrm{i}}}(p)$ are obtained. If w is the first web service to generate p, we name w as a predecessor web service of $p \in P$, denoted as $\mathrm{PD}_{\mathrm{ws}}(p)$.

In the regression search, we adopt a heuristic-based greedy algorithm. The sub-goal in the algorithm is denoted as subGoal. We define W as a set of web services, $w \in W$ and $w_i \in \mathrm{PD}_{\mathrm{ws}}(p)$, $p \in \mathrm{subGoal}$. In each step of the regression search, we adopt the heuristic algorithm to select web services from W. The heuristic based on a hypothesis is as follows:

Hypothesis 1 Choosing a web service with a greater contribution to match the sub-goal earlier in the search helps reaching the initial state faster:

$$h_{\mathrm{sg}}(w) = |w^{\mathrm{o}} \cap \mathrm{subGoal}|$$

In other words, our heuristic attempts to avoid a partial matching case by reducing the size of partial matching web services as much as possible. The algorithm is shown as follows:

```
Input: r^i, r^o
Ourput: w_1, w_2, ..., w_n
    For all (p ∈ P)
      if p ∈ r^i then k_{r^i}(p) = 0
          else k_{r^i}(p) = ∞
    s = (r^i \ r^o); C = φ; t = 1
    while ¬ (s ⊇ r^o)
        δ = {w | w ∈ Ω(s), w ∉ C}
        For all p in w^o (w ∈ δ)
          if k_{r^i}(p) = ∞
              k_{r^i}(p) = t; PD_ws(p) = w; s = s ∪ {p}
        C = C ∪ δ; t + +
    s = (r^o \ r^i); subGoal = s
    while ¬ (subGoal = φ)
        W = ∪_{p ∈ subGoal} PD_ws(p)
        χ = arg max_{w ∈ W} h_sg(w)
        s = s ∪ (χ^o \ r^i)
        sol = sol ∪ {χ}
        subGoal = (subGoal ∪ χ^i) \ s
    s = r^i
    while ¬ (sol = φ)
        if w ∈ Ω(s) and w ∈ sol
          Print w; s = s ∪ w^o; sol = sol \ {w}
```

3 Algorithm Analysis

To validate the efficiency of our algorithm, we conducted a comparison with another heuristic web services composition algorithm—BF*[5] and a semantic-based best-result service composition method[2-3]. All the experiments were performed on a PC platform with Pentium 4 (1.7 GHz), Windows XP SP2, and 1 024 MB RAM. All the algorithms were implemented in Java. We adopted stochastic generating web services and stochastic generating user goals as test sets. The experiment selects nine data sets — the total number of services are 300, 600, 900, 1 200, 1 500, 1 800, 2 100, 2 400, 2 700 plus 100 service composition requests to test the service composition and calculate the average consuming time and the successful composition rate as the result. The experimental results are shown in Fig. 2 and Fig. 3.

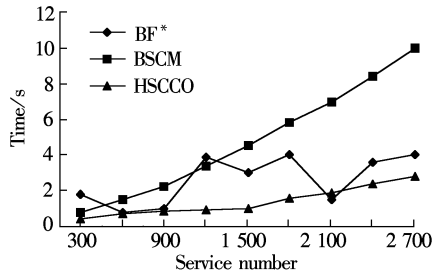

Fig. 2 Composition efficiency comparison

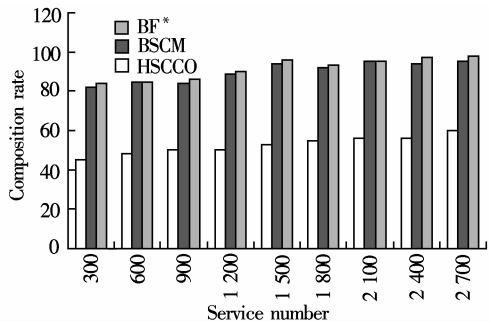

Fig. 3 Composition rate comparison

From Fig. 2, we can see that BSCM has the lowest efficiency. With the increase in the number of web services, the composition time increases rapidly, which means lower expansibility. Since this method is to extend the semantics of the services first, then use the best-result method to get the optimal set of web services. It simply considers the service quality and at the same time sacrifices the efficiency. The performance of the BF* algorithm has greater fluctuation with the

changing of the number of services, which is related to its heuristic rules. In the BF* algorithm, if the output set of a web service has the greatest intersection with the goal set, this service will be given the priority to be selected. According to different test sets, the performance of this algorithm will vary greatly. The HWSCO method has the best performance. And we can see in Fig. 3 that the gap of the rate of successful composition between HWSCO and BSCM is very small, but their rates are much higher than BF*, which illustrates the necessity of the service composition based on semantics.

4 Related Work

Web service composition is a current research hotspot. Now there are several web service composition languages such as BPEL4WS[6], WSCI, etc., all of which are used to describe the web service composition model. These works are the foundation of our research work.

There are two ways based on ontology to describe the semantic information between web services. One is to use the domain metadata to differentiate similar semantics. The other direction is based on methods to combine services whose annotations match based on some notion of similarity. In Ref. [7], matching of web services from a directory is formalized based on various inexactness measures. SWORD[1] was one of the initial attempts to use planning to compose web services. It does not model service capabilities in ontology but uses rule chaining to compose web services. Sirin et al.[8] used contextual information to find matching services at each step of service composition. They further filter the set of matching services by using ontological reasoning on the semantic description of the services as well as by using user input. Web services modeling ontology (WSMO)[9] is a recent effort for modeling semantic web services in a markup language (WSML) and also defining a web service execution environment (WSMX) for it.

5 Conclusion

Existing web service specifications and protocols are limited to the syntactic level, which cannot express semantic information, thus limiting the dynamic composition ability of web services. This paper proposes a heuristic web service composition method based on domain ontology, which uses domain ontology and its inference rules to filter web services according to semantic matching, and then uses an AI planning based heuristic algorithm to design an efficient service composition algorithm. The experiment shows that this method not only can generate service composition automatically, but also can guarantee the quality and efficiency of the composition.

This paper discusses the common situation of service composition (the semantic of the input parameters of service contain the semantic of the output parameters of subsequent services). The next step is to expand the algorithm proposed by this paper to more service composition situations (For example: several service outputs combined to semantically contain the input of the subsequent services).

References

[1] Ponneskanti S R, Fox A. SWORD: a developer toolkit for web service composition[C]//*Proceedings of International World Wide Web Conference*. Honolulu, Hawaii, USA, 2002: 83 – 107.

[2] Mao Z M, Katz R H, Brewer E R. Fault tolerant, scalable, wide-area internet service composition. Technical Report No. UCB//CSD-01-1129[R]. Berkeley: University of California, 2001.

[3] Zhang R, Arpinar B, Aleman-Meza B. Automatic composition of semantic web services[C]//*Proceedings of International Conference on Web Services*. Las Vegas, Nevada, USA, 2003: 38 – 41.

[4] Sirin E, Hendler J, Parsia B. Semi-automatic composition of web services using semantic descriptions[C]//*Web Services*: *Modeling*, *Architecture and Infrastructure*, *Workshop in Conjunction with ICEIS*. Angers, France, 2003: 17 – 24.

[5] Oh S C, On B W, Larson E J, et al. BF*: web services discovery and composition as graph search problem[C]//*Proc of the* 2005 *IEEE Intl Conf on e-Technology*, *e-Commerce and e-Service*. Las Vegas, Nevada, USA, 2005: 784-786.

[6] BPEL4WS Consortium. Business process execution language for web services[EB/OL]. (2003)[2007-04-30]. http://www.ibm.com/Developerworks/library/ws-bpel.

[7] Paolucci M, Kawmura T, Payne T, et al. Semantic matching of web services capabilities[C]//*Proceedings of the First International Semantic Web Conference* (*ISWC*). Sardinia, 2002: 333-347.

[8] Sirin E, Parsia B, Hendler J. Composition-driven filtering and selection of semantic web services[C]//*Proceedings of the First International Semantic Web Services Symposium*. AAAI Press, 2004: 129 – 136.

[9] Erol K, Hendler J, Nau D S. HTN planning: complexity and expressivity[C]//*Proceedings of the* 12*th National Conference on Artificial Intelligence* (*AAAI*-94). Seattle, 1994: 1123 – 1128.

一种基于领域本体的启发式 web 服务组合方法

任红博[1] 邢春晓[2]

([1] 清华大学计算机科学与技术系,北京 100084)
([2] 清华大学信息技术研究院,北京 100084)

摘要:为了实现 web 服务的自动组合,提出了一种基于领域本体的启发式算法. 该方法将领域本体与人工智能规划方法相结合,利用领域本体及其推理能力,推理出参数间的语义关系,在此基础上运用人工智能规划的启发式算法将 web 服务组合问题转化为规划问题加以解决. 实验结果表明,该方法弥补了以往人工智能规划方法中缺乏语义的不足,综合考虑了服务语义、服务组合质量和服务组合效率等因素,能高效地自动生成满足用户需求的组合 web 服务.

关键词:领域本体;语义;web 服务;服务组合;启发式
中图分类号:TP311