A Logic-Based Redundancy Filtering Approach for Web Service Composition

SHIYANG Deng[1, 2, a] and YUYUE Du [1, b]

[1] College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590, China

[2] College of Computer Engineering, Weifang University, Weifang, 261061, China

[a]gmdsy@163.com, [b]yydu001@163.com

Keywords: Web Service Composition; Planning Graph; Redundancy Filtering; Conjunctive Normal Form; Disjunctive Normal Form

Abstract. Targeting the redundancy filtering problem of Web service automatic composition, this paper proposes a novel approach to find the solutions with logic formula. For an original service composition getting by the parallel layered planning graph, it constructs composite services with a backward method according to the source services of parameters starting from the user request outputs. The combination process of the source services is treated as a process of converting a conjunctive normal to a disjunctive normal form, which can filter all the redundant services and reduce the combination scale rapidly. Experiments with large service repository illustrate that the approach is correct and can improve the efficiency of service composition.

Introduction

Web Services are self-describing and platform-agnostic computational elements, and are advocated to support service oriented architecture, distributed computing and software reuse [1]. With the popularity of Web service application, service composition has become the hot spot of current research.

The parallel layered composition approach based on planning graph [2-6] is a frequently-used approach of Web service composition based on artificial intelligence. It separates the process of service composition into two stages of forward searching and backward combining. It constructs a parallel layered service composition at the stage of forward searching. And then, at the backward stage, it constructs composite services according to the source services of parameters starting from the user request outputs.

In large scale service repository, for the inflating of search space, efficiency has become a problem to most of the method of service composition. Some researchers [7-10] give out service discovery methods based on service clustering to reduce search space and other researchers propose QoS-aware methods to reduce the scale of composition [2-6,11,12]. These methods are helpful to improve the efficiency of service composition in large scale service repository, but there is lack of efficient method for filtering redundancy in service composition. This paper proposes a novel approach based on logic formula which treats the process of combining services as converting a conjunctive normal form to a disjunctive normal form. It can filter all the redundant services and reduce the composition scale rapidly and thus improve the efficiency of service composition.

Problem definition

Service composition is to find a group of services which can execute in a certain order with some given inputs and the final outputs of the services can meet the user request. This group of services can be called as a composite service and its formal definition is as follows.

Definition 1（Composite Service）A composite service is a triple $S=\{I,O,G\}$, where

(1) I is the input parameter set of S;

(2) O is the output parameter set of S;

(3) $G=(V, E)$ is a directed acyclic graph, where

a) $V=\{s_i, s_1, s_2, \ldots, s_n, s_o\}$ is a vertex set and each vertex represents a service in the composition, where s_i is a virtual service using $S.I$ as outputs but no inputs and s_o is a virtual service using $S.O$ as inputs but no outputs.

b) E is the set of edges of the graph. The outgoing and incoming edges of a vertex represent the output and input parameters of the service, respectively.

c) For $\forall s_k \in V$, let $V_k = \{s_{k_1}, s_{k_2}, \ldots, s_{k_m}\}$ be the source service set of s_k which contains all the vertices having directed edges to s_k and $V_k.O = \bigcup_{j=1}^{m} s_{k_j}.O$, then $s_k.I \subseteq V_k.O$; for $\forall V_k^{'} = V_k - \{s_{k_j}\}$ $(1 \le j \le m)$, $s_k.I \not\subset V_k^{'}.O$; if $s_k \neq s_o$, then $\exists s_j \in V$ and $s_k.O \cap s_j.I \neq \varnothing$.

The diagram of a composite service is shown as Fig.1.

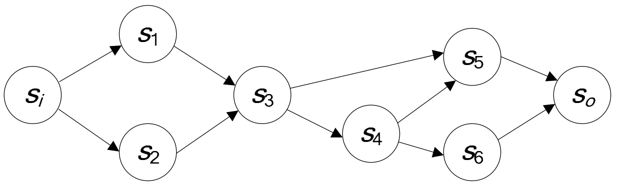

Fig.1. A composite service diagram

The parallel layered composition approach based on planning graph does service composition according to definition 1, but it can not solve the redundancy problem effectively. For example, there is an original service composition created in the forward stage of the parallel layered composition approach shown as Fig.2. There are two kinds of redundant services: (1) According to definition 1, service s_x is redundant because it is not s_o and has no post service. (2) The virtual service s_o has 3 input parameters p_9, p_{10} and p_{11} and their source service set are $\{s_3\}$, $\{s_3, s_4\}$ and $\{s_5, s_6\}$ respectively. By selecting one service from each set, we can get 4 kinds of combination – $\{s_3, s_3, s_5\}$, $\{s_3, s_3, s_6\}$, $\{s_3, s_4, s_5\}$ and $\{s_3, s_4, s_6\}$ and any of them can provide all the input parameters for S_o. Easy to know that, service s_3 is duplicated in the first two combinations and service s_4 is redundant in the last two combinations. The backward stage can remove the first kind of redundancy like s_x easily but it should find an effective method to solve the second kind of redundancy like s_3 and s_4.

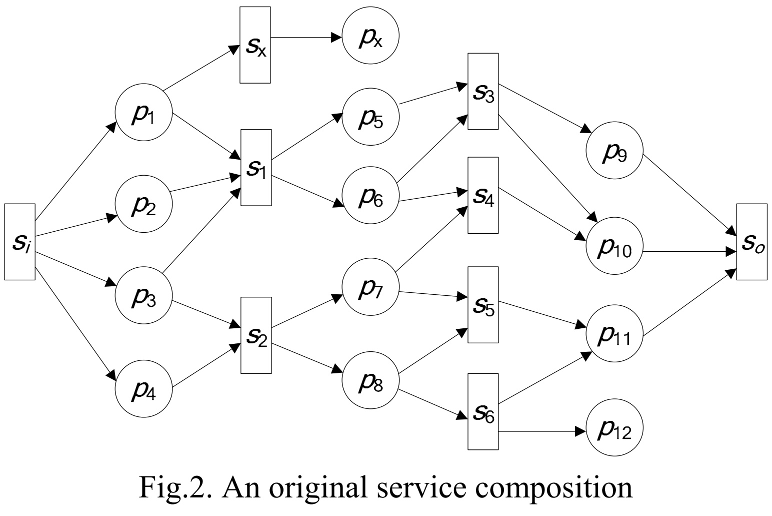

Fig.2. An original service composition

Annotation: Fig.2 is a detailed form of directed acyclic graph which can easily denote the parameter dependency among services. Actually, it is the famous Petri net [13]. In Fig.2, a rectangle denotes a service, a circle denotes a parameter and directed arcs denote the input or output relations between service and parameter.

Analyzing the combination process, we can know that, the source service set of a parameter can be denoted as a logic disjunctive form because any service of it can output this parameter, and simultaneously, all the source service sets of a service's input parameters can be denoted as a conjunctive normal form because a combination must select one service from each source set to output all the inputs for the service. Analogously, a combination can be denoted as a logic conjunctive form and all the combinations can be denoted as a disjunctive normal form. So the process of combing the source services of a service can be regard as converting a conjunctive normal form to a disjunctive normal form and we can construct service compositions by filtering redundant service with logic formula.

Construct Service Compositions with Logic Formula

The logic formulas used in this paper are shown as follows.

Absorptive law: $A\wedge(A\vee B)=A \quad A\vee(A\wedge B)=A$ (1)

Commutative law: $A\wedge B=B\wedge A \quad A\vee B=B\vee A$ (2)

Associative law: $(A\wedge B)\wedge C=A\wedge(B\wedge C) \quad (A\vee B)\vee C=A\vee(B\vee C)$ (3)

Definition 2（Minimal Cover）[14] For a given family of sets $F=\{F_1, F_2,\ldots, F_n\}$, H is called a minimal cover set of F, and set $M_c(F)=\{H \mid H$ is a minimal cover set of $F\}$ is called a family of minimal cover sets of F, if

(1) $\forall F_i \in F, H\cap F_i \neq \varnothing$

(2) $\forall H^{'} \subset H, \exists F_i \in F, H^{'} \cap F_i = \varnothing$

According to definition 2, combing the source services of a service is to find the family of minimal cover sets of its parameters' source sets, and if treating a family of sets as a conjunctive normal form, the family of minimal cover sets can be treat as the equivalent disjunctive normal form of it.

Theorem 1 For a given family of sets $F=\{F_1, F_2,\ldots, F_n\}$, if $\exists F_i \subseteq F_j (i\neq j)$, then $M_c(F)= M_c(F-\{F_j\})$.

Proof: Treat F as a conjunctive normal form of $F_1\wedge F_2\wedge\ldots\wedge F_n$, because of $F_i\subseteq F_j$, there must exist a set G and $F_j=F_i\cup G$. According to absorptive law (formula 1), $F_i\wedge F_j=F_i\wedge(F_i\vee G)= F_i$. That means the two conjunctive normal forms of F and $F-\{F_j\}$ are equivalent, then the disjunctive normal forms of then are equivalent. According to definition 2, $M_c(F)= M_c(F-\{F_j\})$. ■

According to theorem 1, before combing the source services of a service, we can check the source service sets. If one set contains another, it can be removed and the combing result is the same as the original one. For example, service s has 3 input parameters p_1, p_2 and p_3 and their source service set are $\{s_1\}$, $\{s_1, s_2\}$ and $\{s_3, s_4\}$ respectively. Because $\{s_1\}\subseteq\{s_1,s_2\}$, set $\{s_1,s_2\}$ can be removed and the result can be got by combing $\{s_1\}$ and $\{s_3,s_4\}$. This method can filter redundancy and improve the efficiency by reducing the scale of combinations.

Nevertheless, it can not remove all the redundant services only using theorem 1. There is a service s with 4 input parameters and the source sets are $\{s_1, s_2\}$, $\{s_3\}$, $\{s_1, s_4\}$ and $\{s_3, s_4\}$. According theorem 1, the set $\{s_3, s_4\}$ can be removed because it contains $\{s_3\}$. Then, we can get four combinations - $\{s_1, s_3, s_1\}$, $\{s_1, s_3, s_4\}$, $\{s_2, s_3, s_1\}$ and $\{s_2, s_3, s_4\}$. Because service s_1 is duplicated, the first combination will become $\{s_1, s_3\}$. Further, service s_4 is redundant in the second combination and service s_2 is redundant in the third one. Thus, there are only two combinations - $\{s_1, s_3\}$ and $\{s_2, s_3, s_4\}$ on the source services of s. To this problem, we can also use logic formula to filter redundant service in the combing process.

Theorem 2 Let $F=\{F_1, F_2,\ldots, F_{n-1}\}$ be a given family of sets, the family of minimal cover sets of F be $M_c(F)=\{H_1, H_2,\ldots, H_m\}$, F_{n+1} be a set and $G=F\cup\{F_{n+1}\}$, if $\exists H_i \in H$ and $U=H_i\cap F_{n+1}\neq\varnothing$, then $H_i\in M_c(G)$ and for $\forall f_k\in U$, if $\exists H_j\in H$ and $H_i-\{f_k\}\subseteq H_j$, then $(H_j\cup\{f_k\})\notin M_c(G)$.

Proof: First, because the minimal cover set H_i can be treat as a conjunctive form and the set F_{n+1} can be treat as a disjunctive form, all the minimal cover sets of G extended from H_i can be denote as $H_i\wedge F_{n+1}$. Suppose $f_k\in U$, $A= H_i-\{f_k\}$, $B= F_{n+1}-\{f_k\}$, then, H_i can be denoted as $H_i=A\wedge f_k$ and

F_{n+1} can be denoted as $F_{n+1} = B \vee f_k$. According to formula 3 and formula 1, $H_i \wedge F_{n+1} = (A \wedge f_k) \wedge (B \wedge f_k) = A \wedge (f_k \wedge (B \wedge f_k)) = A \wedge f_k = H_i$, so that, H_i is a minimal cover of G, that is $H_i \in M_c(G)$. Second, because of $H_i - \{f_k\} \subseteq H_j$, there will be $H_i \subseteq H_j \cup \{f_k\}$, according to definition 2, $H_j \cup \{f_k\}$ is not a minimal cover set of G, that is $(H_j \cup \{f_k\}) \notin M_c(G)$. ■

Using theorem 1 and 2, this paper gives out an incremental algorithm to filter the redundant service in the process of service combination. The general steps are as follows.

Step 1: Reduce the family of source service sets $H=\{H_1, H_2,\ldots, H_m\}$ according theorem 1;

Step 2: Choose the first source set H_1, make each service of H_1 as a combination and put it into a list *rstL*;

Step 3: Choose next source set H_j, for each combination F_i in *rstL*, if $F_i \cap H_j = setU \neq \varnothing$, then move F_i into a temporary list *rstLT* from *rstL*; For each service *s* in *setU*, put $X_{is}=F_i - \{s\}$ into a hash map *mapX* (the key is *s*, the value is a family of sets,，X_{is} is an element of it).

Step 4: For each combination F_i in *rstL*, for each service *s* in H_j, if there is no set X_{is} in *mapX* where $X_{is} \subseteq F_i$, then combine F_i with *s* and put F_i into *rstLT*;

Step 5: Let *rstL*=*rstLT* and *rstLT*=$\varnothing$, then redo the steps of 3, 4 and 5, till all the sets in *H* are checked.

To the above service *s* and the family of source service sets $H=\{\{s_1, s_2\}, \{s_3\}, \{s_1, s_4\}, \{s_3, s_4\}\}$, do step 1, then $H=\{\{s_1, s_2\},\{s_3\},\{s_1, s_4\}\}$; do step 2, then $rstL=\{\{s_1\},\{s_2\}\}$; do step 3, 4 and 5, then $rstL=\{\{s_1,s_3\},\{s_2,s_3\}\}$; redo step 3, because of $\{s_1, s_3\}\cap\{s_1, s_4\}=\{s_1\}$， move $\{s_1, s_3\}$ to *rstLT* and put $\{s_3\}$ into *mapX*; do step 4, for the rest combination $\{s_2,s_3\}$ in *rstL*, check the last source service set $\{s_1,s_4\}$, because s_1 is in *mapX* and $\{s_3\}\subseteq\{s_2,s_3\}$, then $\{s_2,s_3\}$ will not combine with s_1; because s_4 is not in *mapX*, add $\{s_2,s_3,s_4\}$ into *rstLT*; do step 5, get the final combination list $rstL=\{\{s_1, s_3\}, \{s_2, s_3, s_4\}\}$.

Combing source services in the process of constructing composite services is a geometric series problem. Too many compositions take much execution time and it is difficult to select a suitable composite service for users. Suppose a service has *m* parameters and each parameter has *k* source services, there will be k^m combinations. Using the method of this paper, in the best situation, all the source service sets are same and there are only *k* combinations, then the execution time is only $\frac{1}{k^{m-1}}$ to the simple method; in the worst situation, every set has no intersection with others, the number of combinations is the same as the simple method; in other situations, because a service usually has more than one parameter and there are many similar services in mass service repository, it is probable that two source service has intersection or one includes another, thus the method of this paper can reduce the scale of combinations drastically.

Experiments

Experiments in this paper use the parallel layered composition approach based on planning graph. In the forward stage, we use web service cluster to improve the search efficiency. In the backward stage, three methods are used to compare the average execution time and service filtration efficiency. The first one is from literature [5] which uses Qos-based method (QM) to filter redundant services, the second one uses the logic-based method (LM) of this paper to filter redundant services, and the third method uses the first two methods (QLM). The programs are coded by Java language and the hardware environment is ThinkPad R61 (CPU dual core 2.10GHz, 2.97GB RAM).

Analog data is used in the experiments. For each method, five tests are done with the number of services from 50,000 to 250,000. With randomly created 100 user requests in each test, the average execution time is recorded and shown in Figure 3. The average execution time of LM is more than that of QLM and but much less than QM. This illustrates that the runtime efficiency of

LM is higher than QM and using both QM and LM can have higher runtime efficiency than any of them alone.

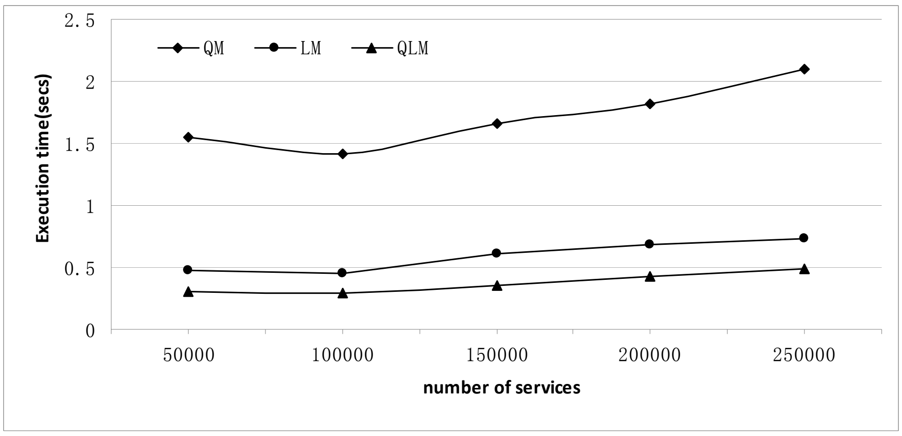

Fig.3. Service composition efficiency

With the repository of 250,000 services, the filtration efficiency of the three methods is shown in Table 1. With the same 100 user requests, the three methods construct composite services with the original compositions getting by the same forward method. In Table 1, the column of *Services* denotes the average total number of the services in the parameter source sets of the end virtual service before filtration. The column of *Service sets* denotes the average number of input parameter source sets of the end virtual service before filtration. The column of *combinations* is the number of source service combinations of the end virtual service. From the service number after filtration it seems that QM has higher filtration efficiency than LM, but the execution time of LM is much less than QM. That is because LM can filter redundant service in the combination process. The LM method can not get the composite service with optimal QoS, so we use the QLM method of both QM and LM to filter redundancy and it can have higher efficiency than those two.

Table 1. Filtration efficiency of the three algorithms

filter method	Services	service sets	Services after filtration	Service sets after filtration	Combinations	execution time
QM	1609.08	9.24	23.58	9.24	4.98	2.22
LM	1609.08	9.24	509.36	3.73	36.18	0.689
QLM	1609.08	9.24	5.73	2.46	3.21	0.445

Conclusions

This paper proposes a novel approach for filtering redundancy in the process of Web service automatic composition. It constructs composite services with a backward method according to the source services of parameters starting from the user request outputs. The combination process of the source services is treated as a process of converting a conjunctive normal to a disjunctive normal form, which can filter all the redundant services and reduce the combination scale rapidly. Experiments with large service repository illustrate that the approach is correct and can improve the efficiency of service composition.

Acknowledgements

This work is supported by the National Basic Research Program of China under grant 2010CB328101; the National Natural Science Foundation of China under grants 61170078 and 61173042; the doctoral program of higher education of the specialized research fund of China under grant 20113718110004; and the Science and Technology Development Program of Weifang City of China under grant 2010rkx011.

References

[1] D. Berardi, F. Cheikh, et al. Automatic service composition via simulation [J]. Int. J. Found. Comput. S. 2008, 19(2): 429–452.

[2] M. Naseri and A. Tomhidi. QoS-Aware Automatic Composition of Web Services using AI planners[C]. Proceedings of Internet and Web Applications and Service(ICIW'07): IEEE Computer Society, 2007: 29-35

[3] Z.Q. Huang, J. Wei, et al. Effective pruning algorithm for QoS-aware service composition [C]. Proceedings of IEEE Conference on Commerce and Enterprise Computing: IEEE Computer Society, 2009:519-522

[4] J. Wei, C. Zhang, et al. QSynth: A Tool for QoS-Aware Automatic Service Composition [C]. Proceedings of International Conference of Web Services(ICWS2010): IEEE Computer Society, 2010: 42-49

[5] S.G. Deng, B. Wu, et al. QoS Optimal Automatic Composition of Semantic Web Services [J]. Chinese Journal of Computers, 2013, 36(5): 1015-1030.

[6] H. Liu, Z.B. Zheng, et al. A global graph-based approach for transaction and QoS-aware service composition [J]. KSII Transactions on Internet and Information Systems, 2011, 5(7): 1252-1273

[7] P. Sun and C.J. Jiang, Using service clustering to facilitate process-oriented semantic web service discovery. Chinese Journal of Computers, 31 (2008) 1340-1353.

[8] X.Z. Liu, G. Huang, H. Mei. Consumer-Centric Service Aggregation: Method and Its Supporting Framework [J], Chinese Journal of Software, 2007, 18(8): 1883-1895.

[9] S. Han, H.Y. Wang, L.Z. Cui. A user experience-oriented service discovery method with clustering technology[C]. Second International Symposium on Computational Intelligence and Design, 2009: 64-67

[10] R. Sudha and S. S. Thamarai. Semantic grid service discovery approach using clustering of service ontologies [C]. In: Proceedings of IEEE TENCON 2006, Nov. 14-17, pp. 1-4.

[11] EI Haddad J, M. Manouvrier, M. Rukoz. TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition. IEEE Transactions on Services Computing, 2010, 39(1), 73-85

[12] L. Wang, Y.X. He. A Web service composition algorithm based on global QoS optimizing with MOCACO [C]//Proceedings of International Conference on Informatics, Cybernetics, and Computer Engineering. Melbourne, Australia, 2012:79-86

[13] T. Murata. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, 1989, 77(4), 541-580.

[14] H. Zhang, L.F. Wu, et al. An Algorithm of r-Adjustable Negative Selection Algorithm and Its Simulation Analysis [J]. Chinese Journal of Computers, 2005, 28(10): 1614-1619