**TELKOMNIKA Indonesian Journal of Electrical Engineering**
Vol.12, No.1, January 2014, pp. 852 ~ 858
DOI: http://dx.doi.org/10.11591/telkomnika.v12i1.3536

# Web Service Composition Algorithm based on Description Logic

**Wang Yuemin**
Department of Computer Science, Suqian College, Suqian, China 223800
e-mail: wym07@163.com

***Abstract***

*This paper presents a dynamic Web service composition algorithm based on description logic. Each input or output of the Web service request is regarded as a concept of description logic and according to the semantic similarity between concepts given the five types of description logic, we study on the dynamic composition algorithm based on service. Because the factors of service semantics and quality of service composition are considered, the algorithm can automatically implement dynamic composition according to the user's service request. Because of using the expression ability of semantic description logic, when Web service is combined the algorithm can determine whether the service can be combined according to the semantic relationship between concepts, thus the probability of successful combination can be improved.*

***Keywords****: Web service; dynamic composition; description logic; service composition*

## 1. Introduction

Since the Web service appears, it has always been the concern of the majority of researchers because of its characteristics of self-contained and loosely coupled. Integrating multiple Web services to complete the new more complex functions has broader application prospects. Now the researchers have a lot of research on the Web service composition.

The existing protocols and standards only describes Web service in the syntax level and can not express the semantic information, so usually Web service is combined by using the manual method [1-3]. Because the manual method requires the active participation, it is not only time-consuming but also is difficult to adapt to the dynamic change of service.

DL (Description Logic) can accurately describe the relationship between the concepts, and thus becomes the semantic basis of mutual understanding between the machines and between man and machine. Franz Baader et.al. Proposes a formal action description logic-based approach to describe the functionality of Web services and to describe Web service as the action with preconditions and effects, and uses description logic assertions to represent the preconditions and effects. Using the description logic Wang Hai and Chen Yan of the Xi'an Jiao Tong University put forward a method to match the premise and effect of web service [4-5]. This paper presents a dynamic Web service composition algorithm based on description logic, each input or output of the Web service request is regarded as a concept of description logic, and according to the semantic similarity between concepts given the five types of description logic we propose a dynamic composition algorithm based on Web service. The algorithm can automatically implement dynamic composition according to the user's service request.

## 2. Foundation of the Description Logic

Description logic is a formal tool of knowledge representation. The method that it represents an application domain knowledge is firstly to define related concepts in this field, then to use these concepts to describe properties and individuals of the objects in the field [6]. One of the most remarkable characteristics of description logic is a formal and logic-based semantics, and another notable feature is that the reasoning is emphasized as the core service [7].

The primitives of description logic are atomic concepts and atomic relations, and complex concepts derive from constructed operator [8]. In the abstract notation, the letters A

*Received June 17, 2013; Revised August 29, 2013; Accepted September 20, 2013*

and B represent atomic concepts. The letter R represents an atomic role. The letters C and D represent concept descriptions. Description language can be distinguished by operator. Language AL (Attributive language) is proposed as the minimum language with the actual value. The basic syntax of AL is shown in table 1.

Table 1. the basic Syntax of AL

| C,D | formula | explanation |
|---|---|---|
| | A | atomic concept |
| | T | global concept |
| | $\perp$ | underlying concept |
| | $\neg$ A | atomic negation |
| | C∩D | intersection |
| | $\forall$R.C | value restriction |
| | $\exists$R.T | restricted existential quantifier |

The AL concept has a formal semantics. For an interpreter I, it contains a non-empty set I (the definition domain) and an interpretation function. The function assigns a set $A^I \subseteq \Delta^I$ for each atomic concept A and assigns a binary relation $R^I \subseteq \Delta^I \times \Delta^I$ for each relation R. The interpretation function is extended to represent the semantic of the concept description by the following inductive definition.

$T^I = \Delta^I$

$\perp^I = \varphi$

$(\neg A)^I = \Delta^I \backslash A^I$

$(C \cap D)^I = C^I \cap D^I$

$(\forall R.C)^I = \{a \in \Delta^I \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$

$(\exists R.T)^I = \{a \in \Delta^I \mid \exists b.(a,b) \in R^I\}$

When $C^I = D^I$, two concepts of C and D are equivalent, writing C ≡ D.

Adding more operators to AL as follows, the language of stronger expression ability can be obtained, which is known as the family of AL languages.

The joint of concept can be written as CUD, and can be interpreted as: $(CUD)^I = C^I U D^I$

Completely existential quantifier can be written as existing R.C, and can be interpreted as: $(\exists R.C)^I = \{ a \in \Delta^I \mid \exists b.(a, b) \in R^I \wedge b \in C^I\}$

Note that it is different from $\exists$R.T. Because you can use any concept, the constraint is writing as: $(\geq nR)^I = \{ a \in \Delta^I \mid \{b \mid (a, b) \in R^I\}\mid \geq n \text{和} (\leq nR)^I = \{ a \in \Delta^I \mid \{b \mid (a, b) \in R^I\}\mid \leq n$

Given a description logic knowledge base and two description logic concepts, description logic reasoner provides two standard reasoning services:

(1) the implication: Check whether S is more specific than D.

(2) the satisfiability: Check whether S is satisfiable.

Given a request D(demand) and a resource S(supply), reasoning services based on these standards five kinds of semantic matching type between concepts can be given as follows:

(1) S and D exactly match. S and D are semantically equivalent, and the characteristics that D requires can be provided by S. For the D, S has no any additional features.

(2) S and D fully match. S is more specific than D, so all features D requires are provided by S. There are other features in S which are neither required by D nor conflict with the characteristics of D.

(3) S and D plug-in match. D is more specific than S. All the features provided by S are required by D. D also asks other features, and these features are neither provided by S nor conflict with the characteristics of S.

(4) S and D potentially match. D is compatible with S. The characteristics that D requires do not conflict with the characteristics that S provides in logic.

(5) S and D partially match. D and S are not consistent. The characteristics that D requires logically conflict with the characteristics that S provides.

We describe semantic information of Web services by using description logic. Through the reasoning function of description logic we can achieve the dynamic discovery and composition of web services.

## 3. Web Service Composition Based on Description Logic

The Web service follows certain technical specifications, which provide a unified service registration, discovery, binding and integration mechanism for internet application. Description logic can precisely define the interface semantics of web service, and with the help of description logic a Web service can be abstracted as an entity which consists of input and output [9-11]. A Web service can be described by the expression that $WS_i=(I_i, O_i)$ ,where $WS_i$ is the name of web service, $I_i$ is the input set of web service and $O_i$ is the output set of web service. The user's request can be deemed as a special class of Web service. A Web service request can be described by the expression that ReqWS=(RI, RO), where ReqWS is the request name of Web service, RI is a input set of service requests and RO is the output set of service requests.

In order to obtain better Web service composition, we need to calculate the semantic similarity between concepts [12]. On the calculation of semantic similarity between concepts there are a lot of achievements. The study adopts some ideas and gives the calculation formula of semantic similarity between concepts based on description logic.

**Definition 1: Semantic Similarity between Concepts.** The semantic similarity between concepts refers to the degree that the two concepts can match, and it is a quantitative definition given on the basis of the five match types. The calculation formula is as follows:

$$M(S, D) = \frac{a}{d + a} \tag{1}$$

Where the concepts of S and D are expressed by description logic, the a is an adjustable parameter, a>0, d is a nonnegative integer, M (S, D) is in the range of [0, 1], the value of D is determined by the strategies as follows:

(1) When S and D exactly match, let d=0, then M (S, D) =1;

(2) When S and D perfectly match, let d=1, then 0<M (S, D) <1;

(3) When S and D plug-in match, potentially match or partially match, let M (S, D) =0.

Semantic similarity between the sets of concepts can be defined on the basis of semantic similarity between concepts.

**Definition 2: Semantic Similarity between the Sets of Concepts**. The semantic similarity between the sets of concepts refers to the degree that the two sets of concepts can match. The calculation formula is as follows:

$$MM(SS, DD) = \frac{1}{m}\left[\sum_{i=1}^{m} \max_{j=1}^{n}(M(S_i, D_j))\right] \tag{2}$$

Where SS = $\{S_1, S_2, \ldots, S_m\}$ and DD = $\{D_1, D_2, \ldots, D_n\}$ indicate the concept sets, the elements of which are expressed by description logic , and MM (SS, DD) is in the range [0,1].

**Definition 3: Output Decomposition Scheme**. That the output collection ($O(WS_1)$, $O(WS_2)$, …, $O(WS_n)$) of the service ($WS_1$, $WS_2$, …, $WS_n$) is the output decomposition scheme of the output collection $O(WS_x)$ of service $WS_x$ , if it meets:

(1) For each output X of $O(ws_x)$, there is one output set $O(ws_i)$ of the O(WS1), O(WS2), …, O(WSn), and x can precisely match or plug-in match one certain output Y of $O(wS_i)$ ;

(2) If the collection ($O(ws_1)$, $O(ws_2)$, …, $O(ws_n)$) lacks any one, the collection is no longer a output decomposition scheme of $O(ws_x)$.

If we decompose $O(WS_x)$ we may get more output decomposition schemes.

**Definition 4: Matching Degree of the Output Decomposition.** The matching degree of the output decomposition refers to the degree that the O(ws$_1$), O(Ws$_2$), …, O(WS$_n$) matches with O(WS$_x$). The degree of matching is calculated as follows:

$$MatchingDegree = \frac{1}{n}\sum_{i=1}^{n} MM\ (O(WS_i), O(WS_x)) \quad (3)$$

Where the collection (O(ws$_1$), O(ws$_2$), …, O(ws$_n$)) is one output decomposition scheme of O(Ws$_x$).

For some output decomposition scheme (O(ws$_1$), O(ws$_2$), …, O(ws$_n$)) of O(Ws$_x$), we need to sort the services (WS$_1$, WS$_2$, …, WS$_n$) corresponding to the O(ws$_1$), O(ws$_2$), …, O(ws$_n$), for the inputs and outputs of the services (WS$_1$, WS$_2$, …, WS$_n$) may have some association. We can sort the services (WS$_1$, WS$_2$, …, WS$_n$) by means of the Service association graph defined below.

**Definition 5. Service Association Graph**. The service association graph G= (V, E) is a directed graph, where V is the finite nonempty set of the service, and E is the set of service association between the input and output. Element WS$_i$ in V represents the name of the Web service, and ordered pair <WS$_i$, Ws$_j$> in E represents that one output X of WS$_i$ precisely or completely matches one input Y of Ws$_j$.

Sort the services (WS$_1$, WS$_2$, …, WS$_n$), where the WS$_1$, WS$_2$, …, WS$_n$ are the corresponding service names of some output decomposition scheme O(WS$_1$), O(WS$_2$), …, O(WS$_n$). Here we only discuss that the service relationship diagram is a directed acyclic graph. Figure 1 is an example for a service relationship graph.

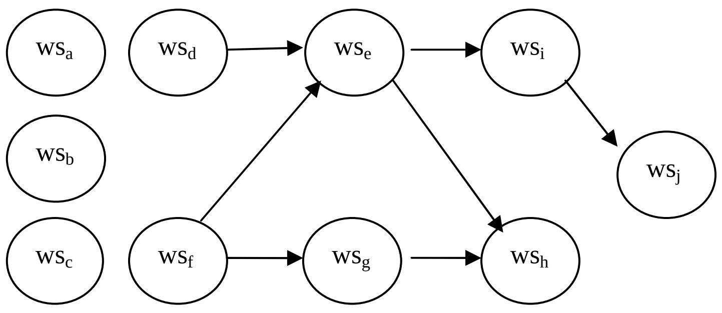

Figure 1. An Example for A Service Relationship Graph

For a given service correlation graph, each service of the graph is sorted as follows:

(1) Output isolated service in the figure in an arbitrary orde.

(2) Select arbitrarily one service whose incoming degree is zero and output it.

(3) Delete this service and its all outgoing edges from the figure.

(4) Repeat the step (2) and (3) until all the services have been output.

Service sequence obtained after the output operation may not be unique. For the service association graph shown in Figure 1, output each service, then a possible sorting result is : WS$_a$, WS$_b$, WS$_c$, WS$_d$, WS$_f$, WS$_e$, WS$_g$, WS$_h$, WS$_i$, WS$_j$.

**Definition 6: Sorted Output Decomposition Scheme**

O(WS$_{p1}$), O(WS$_{p2}$), …, O(Ws$_{pn}$) is the sorted output decomposition scheme of O(WS$_x$), if it meets:For any two service WS$_{pi}$ and Ws$_{pj}$, if i<j or WS$_{pi}$ and Ws$_{pj}$ have no any input and output association, or one output X of WS$_{pi}$ accurately or completely matches one input Y of Ws$_{Pj}$, where the WS$_{p1}$, Ws$_{p2}$, …, WS$_{pn}$ are the corresponding services of some output O(WS$_{p1}$), O(WS$_{p2}$), …, O(Ws$_{pn}$).

To the corresponding service WS$_1$, WS$_2$, …, WS$_n$ of some output decomposition scheme O(WS$_1$), O(WS$_2$), …, O(WS$_n$) of O(WS$_x$), we construct service association graph, and output the service in the graph, then we can get service sequence WS$_{p1}$, Ws$_{P2}$, …, WS$_{pn,}$ The output set O($WS_{p1}$), O($WS_{p2}$), …, O($Ws_{pn}$) corresponding to the $WS_{p1}$, $Ws_{P2}$, …, $WS_{pn}$ is a sorted output decomposition scheme of the O($ws_x$).

## 4. Web Service Composition Algorithm based on Description Logic

### 4.1. Basic Idea of the Algorithm

We regard the output collection RO(ReqWS) of the user's web service request as the root of the tree, and achieve output decomposition of the RO(ReqWS), thus we can get all possible output decomposition schemes. Based on the definition 4 we can calculate the matching degree of output decomposition, according to which output decomposition scheme is sorted from large to small. The maximum matching degree program has a top priority to being done further output decomposition. For the selected output decomposition scheme, we can construct service association graph accordingly, then gain the sorted output decomposition scheme. This process is repeated until the service composition that meets customer's service requests is achieved. In this process sometimes we need backtrack. With the implementation of algorithm we can get the dynamic variation of the tree structure.

### 4.2. Algorithm Description

Under the support of Web service dynamic combination algorithm ,based on description logic and knowledge base we use the semantic similarity between concepts as the foundation to calculate semantic similarity between the collections of concepts, and further to calculate the matching degree of output decomposition. Maximum matching degree scheme has a top priority to being done further output decomposition. For the selected output decomposition scheme, we should sort it, and then gain the sorted output decomposition scheme.

Input: the Web services database, the Web service request, the description logic knowledge base K, the depth limit (bound)

Output: the value of bool type. If it is true, it indicates that exists service combination of meeting user's service request and the corresponding service to output set of the serviceOutPutStack is what we want. Otherwise, it does not exist service combination of meeting user's service request.

Before the start of algorithm, the linear list SingleSearchList=(RO(ReqWS)). The root node RO (ReqWS) of the SearchTree is created. The stack ServiceOutputStack is empty. The available input set UsableInput store each input of the set RI(ReqWS) of the user's service request .

```
       The beginning of the algorithm:
01. SearchNode=GetFirst(SingleSearchList)
02. if IsMember(SearchNode, GetTail(SingleSearchList)), goto FAIL
03. if(SatisfyTerminateCondition(SearchNode)andSearchNode!=RO(ReqWS))gotoSUCCESS
04. if GetLength(SingleSearchList)>bound, goto FAIL
05. PossibleSchemaList=CalculatePossibleDecomPositions(SearchNode)
06. SortSchemaList(PossibleSchemaList)
07. LOOP: if Null(PossibleSchemaList), goto FAIL
08. CertainSchema=GetFirst(PossibleSchemaList)
09. PossibleSchemaList=GetTail(PossibleSchemaList)
10. SortSchema(CertainSchema)
11. for each RecursivcNode in CertainSchema do
12. {  InsertForward(RecursiveNode, SingleSearchList)
13.   Result=ServiceComPoseBacktrack(SingleSearchList)
14.   SingleSearchList=GetTail(SingleSearchList)
15.   if Result=false
16.   {  NodeList=Traverse(SearchNode)
17.      For each Node in NodeList do
18.      {  if lsLeafNode(Node)
19.         {  ServiceOutputStack.Pop()
20.            ModifyUsablelnPut()  }
21.        if Node!=SearchNode SearchTree.DeleteNode(Node)  }
22.      goto LOOP  }
23.   else SearchTree.AddChildNode(SearchNode, RecursiveNode)  }
24. return true
25. SUCCESS: ServiceOutPutstack.Push(SearchNode)
26. ModifyUsablelnPut()
27. return true
28. FAIL: return false
```

```
The end of the algorithm.
```

## 4.3. Running Process of the Algorithm

Figure 2 shows an example of SearchTree and ServiceOutPutstack when the algorithm returns true. In the SearchTree, the root node RO(ReqWS) says output set of the user's service requests. Other than the root node, the remaining nodes represent the output set of some service in the Web services database. The downward dashed arrow shows the failing path. We use dashed lines because the failing path has been removed.The downward solid arrow indicates a successful path. In this case, after the execution of the algorithm we get service composition {$WS_h$, $WS_b$, $WS_q$, $WS_n$, $WS_d$, $WS_n$}. In the execution of the service composition, we perform firstly the corresponding service of output set which is firstly pushed.

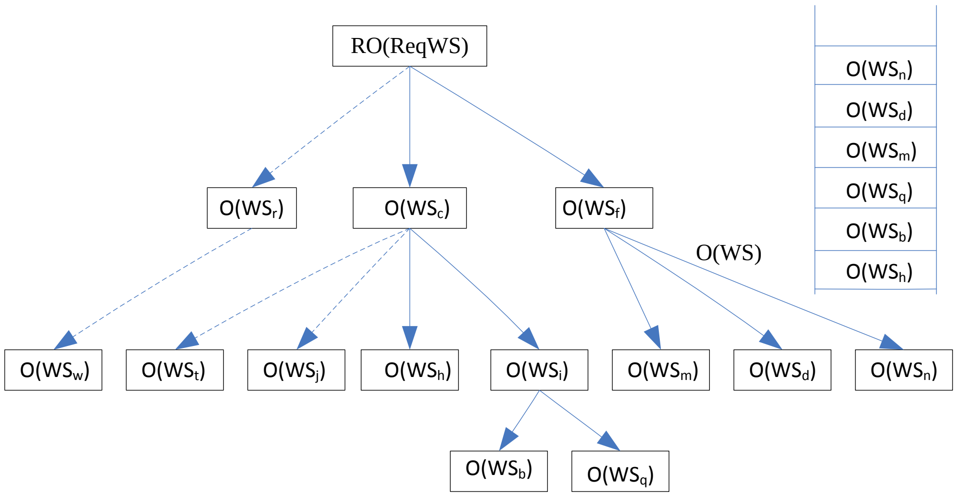

Figure 2 The Instance Graph of Search Tree and Services Output Stack

## 4.4. Algorithm Running Instance

Suppose there are 4 Web services in Web service library. They are respectively:
(1) WSI (SourceCity, DestinationCity, StartDate →Airline)
(2) WS2 (DestinationCity, InDate→ Hotel)
(3) WS3 (StartDate, EndDate, City→CarRent)
(4) WS4 (MeetingName→StartDate, DestinationCity, EndDate)
Suppose there are the following knowledge in the ontology library:

The Meeting can be decomposed into three sub-goals: Airline, Hotel and CarRent. When the user requires the Meeting as output, he should provide MeetingName and SourceCity as input. Firstly the Meeting is decomposed to sub-targets set {Airline, Hotel, CarRent}, of which each sub-goal can not achieve further output decomposition. In the second stage of the decomposition process, the above three sub-goals are decomposed respectively from the output to the input. Take sub-target Airline as an example, through the web service ws1(SourceCity, DestinationCity, StartDate→Airline) the node Airline can be decomposed into SourceCity, DestinationCity and startDate. The sourceCity can meet the termination condition. While through the Web service ws4 ws4(MeetingName→startDate, DestinationCity, EndDate) the DestinationCity is decomposed to the MeetingName, so a termination condition is satisfied. StartDate can be obtained from the Web service ws4(MeetingName→startDate, DestinationCity, EndDate), and other decomposition method of the Hotel and CarRent is as the same as that of Airline.

Finally, the service composition sequence is WS4, WS1, WS2, WS3.The output is not unique.

## 5. Conclusion

Automatic combination of the Web service reflects the user as the center of the business model, so the user's various complicated demand can be implemented through the Web service combination rather than through rewriting the code. The existing service description standards and protocols are limited to syntactic level, so they can not express the semantic information and limit the ability of dynamic composition of the service. While the description logic can accurately describe the semantic relation between concepts and can provide a semantic foundation for mutual understanding between human and machine and between machines. Therefore, based on five kinds of semantic matching between the concepts of description logic, this paper gives the similarity the input and output set, and accordingly gives output decomposition matching degree and service correlation graph and some related concepts. The paper proposes a dynamic composition algorithm of the Web service based on description logic, and the algorithm considers the factors such as service semantics and quality of service composition.

## References

[1] WANG Jie-Sheng, LI Zhou-Jun, LI Meng-Ju. Solutions Towards Automated Composition of Semantic Web Services: A Survey. *Computer Science*. 2007; (134): 19-23.

[2] LIU Hua-wen, SHEN Chun, YANG Dong, LIU Lei.Survey of Semantic Web Service Techniques. *Journal of Jilin University (Information Science Edition)*. 2010; 28(1): 47-54.

[3] LI Shun-xin, LING Hai-yang, JIANG Nan. Research on Web Services Composition Model Based on Workflow Template. *Computer and Modernization*. 2009; (7): 44-47.

[4] Liu Sipei, Liu Dayou, Qi Hong, and Guan Jinghua. Composing Semantic Web Service with Description Logic Rules. *Journal of Computer Research and Development*. 2011; (05): 831-840

[5] WANG Hai1, CHEN Yan, FAN Lin, LI Zengzhi. A Description-Logic-Based Preconditions/Effects Matchmaking Method for Semantic Web Services. *Journal of Xi'an Jiaotong University*. 2010; 44(4): 39-42.

[6] Naji Hasan AH, Gao Shu, AL-Gabri Malek, Jiang Zi-Long. An Optimal Semantic Network-Based Approach for Web Service Composition with QoS. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(8): 4505-4511.

[7] ZHOU Xiang-bing. Semantics topic map Web service composition approach using description logic. *Journal of Computer Applications*. 2010; 30(10): 2763-2767.

[8] YANG Qing, ZHU Li, CHEN Wei. Framework of Ontology Evolution Based on Description Logic. *Computer Engineering*. 2010; 36(13): 79-81

[9] Tari A, Elgedawy I, Dahmani A. A dual-layered model for web services representation and composition. *Intelligient Information System*. 2009; (32): 237-265.

[10] WANG Qing-ming. Dynamic Web service composition model based on domain ontology. *Journal of Computer Applications*. 2009; 29(7): 1957-1959.

[11] Cao Wei, Fan Xi-quan, Bie Xiao-wu, Liu Gang-feng, Wu Cheng-hai. Architecture Description Approach of Information Systems to System Optimization. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(3): 1322-1327.

[12] LIU Si-pei, LIU Da-you, QI Hong, WANG Jia-qiang. Service community chain based approach for web service composition. *Journal of Jilin University (Engineering and Technology Edition)*. 2010; 40(l): 148-154.