http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (6): 1150-1157, 2010
ISSN 1812-5638

A Fast Algorithm for Web Service Composition Based on Dynamic Description Logic

[1]Wei Liu, [1,2]Yu Yue Du, [1]Bao Qi Guo, [1]Chun Yan and [1]Qiang Xu
[1]College of Information Science and Engineering,
Shandong University of Science and Technology, Qingdao 266510, China
[2]The State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 10080, China

Abstract: Dynamic Description Logic (DDL) is an extension of description logic. In dynamic description logic, both services and the service composition can be expressed in formulae, but there is still much room to promote the efficiency. In this study, a fast and effective method for web service composition is proposed for the services which perform in sequence. This method takes advantage of the partial order relation among services, dividing the service composition into two phases:on the first stage, the partial order diagram is generated in the registration server; on the second stage, the fast algorithm is run based on the partial order diagram to return web service composition to meet requests of users. By analysis and verification, the fast algorithm for web service composition presented in this study can be implemented in a linear time complexity, greatly shortening response time of the system.

Key words: Dynamic description logic, service composition, fast algorithm, partial order, web ontology language, time complexity

INTRODUCTION

With the development of the semantic web (Mao, 2010) and Web Ontology Language (OWL) (Martin *et al*., 2007; Hasany *et al*., 2010), we hope to utilize the semantic web technology to integrate intelligently all kinds of web services, thus producing semantic web services. The automatic composition of semantic web services is one of the key technologies in integrating web services and it is much concerned by many researchers form the beginning. The methods for web service composition are dependent on specific framework of web services in a certain degree. The typical framework of web services is OWL-S which is based on OWL and OWL is on the basis of description logic (Horrocks *et al*. 2003). Description logic is effective in representing and reasoning static knowledge. However, description logic cannot represent and reason dynamic knowledge such as behaviors and services. So, Dynamic Description Logic (DDL) (Shi *et al*., 2004) is presented by combining description logic ALC, dynamic logic and action theory. The description ability of DDL is over logic and system based on proposition language in describing actions. The problems about reasoning are all decidable in reasoning actions. All the problems about reasoning actions can be transformed to the satisfiable problems. So, they can be reasoned based on the decision algorithm in absence of information. In sum, DDL is more applicable for describing semantic web by combining knowledge of static field based on description logic and action knowledge of dynamic field.

At present, the service composition are mainly the following several methods:

- The service composition methods based on type matching of input and output parameters

A solution based on the DAML-S (Paolucci *et al*., 2002) is proposed. The overlap relationship of the types of parameters is studied and the partial matching algorithm is proposed (Li and Horrocks, 2003). For large number of services, a synthetic testbed (Constantinescu *et al*., 2004) is proposed that can be used for simulating large deployments of services and also for generating service composition problems.

- The service composition methods based on artificial intelligence planning

The pioneer of such approaches (McIlraith and Tran, 2002) takes web services as the actions of AI planning. Therefore, web service composition process is the

Corresponding Author: Wei Liu, College of Information Science and Engineering,
Shandong University of Science and Technology, Qingdao 266510, China

process of generating planning. The adopted planning algorithm is improved based on logic program Golog. The service composition based on HTN (Hierarchical Task Network) planning algorithm (Sirin *et al*., 2004) is proposed.

As the description logic can effectively express and reason knowledge of static domain and cannot handle the dynamic knowledge such as actions and services, dynamic description logic DDL (Shi *et al*., 2004) is put forward and the service description method based on DDL (Shi and Chang, 2008) is proposed. And then the service composition algorithm based on DDL (Peng *et al*., 2008) is presented. But because the algorithm needs to enumerate all composition sequences, efficiency is difficult to be guaranteed.

In this study, DDL is taken as service description framework. For services which are performed in sequence, a fast and effective service composition algorithm is presented which takes full advantage of DDL describing dynamic run of services. This method utilizes the partial order relationship between services dividing the service composition into two stages: on the first stage, the partial order diagram is constructed on the registration servicer; on the second stage fast service composition is implemented based on the partial order diagram to meet the requirement of users. To achieve the goal, First inclusion operation on sets of DDL is extended, the partial order relationship between services is defined based on extended inclusion operation and the satisfiability problem of DDL formula is transformed to operations between sets. Second, the fast composition algorithm is proposed based on DDL and the correctness and time complexity is analyzed. Finally, the algorithm of generating the partial order diagram between services is proposed and the time complexity of the algorithm is analyzed. By analysis and verification, the method of service composition designed in this study is implemented in linear time complexity greatly reducing response time of the system.

DDL AND SEMANTIC WEB SERVICES

The basic idea of DDL is that the world is in the process of constant evolution and development and the occurrence of actions can push evolution. The occurrence of actions is restricted by the world today and also influences the world changing from one state to another.

The DDL contains 3 essential elements: concept, relation and actions. Base on simple concept, relation and action, complicated concepts can be constructed with construction operators which is called TBox.

Some definitions related to DDL are given as follow:

Definition 1: An atomic action with arguments is the form:

$$\alpha(V_1,V_2,\ldots,V_n) = (P, E)$$

where, α is a string representing the name of the atomic action, $(V_1,V_2,\ldots,V_n)$ is a finite sequence of all individual variables in P and E representing the interface of the atomic action, P is a set of preconditions which must be satisfied before an action are executed and E is a set of postconditions which denotes the effects of an action.

Definition 2: Formulas in DDL are defined and generated as follows:

$$\varphi, \Psi: := C(u)|R(u, v)|u = v|\neg\varphi|\varphi\vee\Psi|<\pi>\varphi$$

where, u and v are individual variables, C is a concept, R is a relation and π is an action.

Definition 3: An action of DDL is defined as follows:

$$\pi, \pi': := \alpha|\varphi\ ?|\pi\cup\pi'|\ \pi,\pi'|\ \pi^*$$

where, α is an atomic action and φ is a formula.

The service description in OWL-S and the transformation method from OWL-S description to DDL description are given in the following.

In OWL, the function of an service is described by Input, Output, Precondition and Effect (IOPE) which is denoted by S = {I, O, P, E}. Let S.I, S.O, S.P and S.E be respectively input, output, precondition formula set and effect formula set of service S. If $\alpha(V_1,\ldots, V_n) = (P, E)$ is an atomic action of DDL, S can be described as α = S, where the action name is service name. P = S.I $\cup$ S.P is the premise formula of actions which is the union of input formula set and precondition formula set of service S; E = S.O $\cup$ S.E is the result formula of actions which is the union of output formula set and effect formula set of service S. Shi and Liang (2008) proposed a detailed transformation process.

A concrete service example of booking air tickets (Peng *et al*., 2008) is given below:

Example 1: The service description of booking air ticket.

Tbox:
FlightTicketReserveConfirm $\subseteq$ ReserveConfirm,
TrainTicketReserveConfirm $\subseteq$ ReserveConfirm,
LDBusTicketReserveConfirm $\subseteq$ ReserveConfirm,

AirTicket ⊆ Ticket,
TrainTicket ⊆ Ticket,
LDBusTicket ⊆ Ticket

In example 1, the input and output of booking air ticket are as follows:

Input:
Tourism staff: one person
Departure: Beijing
Destination: Yunnan
DepartureTime: December13,2007
Transportation: Aircraft
Credit Card Number : creditCardNo
ID Number: idCard No
Output: booked air ticket

The following is the DDL description of booking air ticket service.

The initial state:

A_0 = {Person (person), DeparturePlace(Beijing), ArrivalPlace(Yunnan), departureDate(2007-12-13), departureTime(17:00), arrivalDate(2007-12-24), arrivalTime (8:00), IntendVehicle(flight), CreditCardNo(creditCardNo), IDCardNo(idCardNo), knows(agent,person), knows (agent,Beijing), knows(agent,Yunnan), knows(agent, 2007-12-13), knows(agent,17:00), knows(agent, 2007-12-24), knows (agent,8:00), knows(agent,flight), knowsVisacard (agent,creditCardNo), knowsIDcard (agent,idCardNo), VisaValidate(person,creditCardNo), AirTicket(y), IDCertify(person,idCardNo), SendTicketConfirm (sendTicketConfirm), holds (person, y)}.

Goal:
φ_0 = {SendTicketConfirm(sendTicketConfirm), holds(person,y)}.

The following are the atomic services used in the matching algorithm.

S_1: The service for searching for the flight information. S_1 searches for the flight information based on the departure place, the arrival place and departure date. The DDL description of S_1 is as follow:

CheckFlight(departurePlace,arrivalPlace,departureDate):
P = {DeparturePlace(departurePlace), ArrivalPlace (arrivalPlace), DepartureDate(departureDate), knows (agent,departurePlace), knows(agent,arrivalPlace), knows(agent,departureDate)}
E = {FlightNo(flightNo), knows(agent,flightNo)}.

S_2: The service for reserving air ticket. S_2 reserves the air ticket based on the flight number, departure date and ID Number. The DDL description of S_2 is as follow:

ReserveFlightTicket(flightNo,departureDate,person,idCardNo):
P = {FlightNo(flightNo), DepartureDate(departureDate), Person(person), IDCardNo(idCardNo), IDCertify (person,idCardNo), knows(agent,flightNo), knows (agent,departureDate)}
E={FlightTicketReserveConfiirm(flightTicketReserveconfirm), knows(agent,flightTicketReserveconfirm)}.

S_3: The service for bank transfer. S_3 is used for bank transfer based on the credit card and the reserve confirmation information. The DDL description of S_3 is as follow:

AccountTransForReserve(person,creditCard-No, reserveconfirm):
P = {Person(person),CreditCardNo(creditCardNo), ReserveConfiirm(reserveconfirm), VisaValidate (person,creditCardNo), knows (agent,reserveconfirm)}
E = {AccountTransConfirm (accountTransConfirm), knows(agent,accountTransConfirm)}.

S_4: The service for delivering tickets. S_4 delivers tickets based on the reserve confirmation information and bank transfer information. The DDL description of S_4 is as follow:

SendTicket(person,reserveconfirm,accountTransConfirm):
P={Person(person),FlightTicketReserveConfiirm(reserveconfirm), AccountTransConfirm(accountTransConfirm), knows (agent, person), knows(agent,reserveconfirm), knows (agent,accountTransConfirm)}
E={SendTicketConfirm(sendTicketConfirm),AirTicket(y), holds (person, y), knows (agent,sendTicketConfirm)}

In example 1, S_1, S_2, S_3 and S_4 are four atomic services. Set A_0 is the initial state composed of the input of users. Set φ_0 is a set of goals that users need to achieve. The task is to find the sequence of service composition making the expression φ is true in final state.

THE FAST WEB SERVICE COMPOSITION ALGORITHM BASED ON DDL

A service composition to meet goals correspond to an action sequence plan = $\{S_1,S_2,\ldots,S_k\}$, where $S_i = (P_i, E_i)$ $(0 = i = k)$ is a plan to achieve the goal φ_0.That is, φ_0 is satisfiable in the state followed by implementation of actions of the plan in sequence. Therefore, the following formula is not satisfiable (Peng *et al*., 2008):

$$[(S_1, S_2,\ldots,S_k)^*]\,\Pi \wedge \text{Conj}(P) \wedge <\text{plan}>\varphi_0(*)$$

where, $\wedge$ denotes the logical AND operator, Π is the formula $(\text{Conj}(P_1) \rightarrow <(P_1, E_1)> \text{true}) \wedge \ldots \wedge (\text{Conj}(P_k) \rightarrow <(P_k, E_k)> \text{true})$ and Conj (S) is the conjunction of all formulae in formula set S.

In the current state, Conj (P)can achieve the goal φ_0 through the implementation of the planning <plan>. The formula Π ensures that an arbitrary atomic action is certain to execute when its premise is satisfiable. And the formula $[(S_1, S_2,\ldots,S_k)^*]\Pi$ ensures that the sequence composed of atomic actions is certain to implement when the premise is satisfiable.

Therefore, as long as an action sequence that makes the formula (*) unsatifiable can be found, the action sequence is the target service composition sequence that meets the goal. it is inefficient by the enumeration of all possible service sequences to verify the satisfiability. In fact, DDL description for services is simple and effective and there exists partial order relation between services. Therefore, the service composition problem can be transformed to operations on sets to gain more efficient algorithms. To do that,first the extension of inclusion operation on sets is given and then the fast service composition algorithm based on partial order diagram between services is put forward.

The extension to inclusion operation on sets and the definition of partial order between services: Concepts in DDL has the nature of heritage and thus the extension to inclusion operation on sets is necessary.

Definition 4: In DDl, let M and N be two sets, $\sqsubseteq$ is the generalized inclusion operation on sets, then $M \sqsubseteq N$ iff the following two conditions are satisfied simultaneously:

(1) $\forall C(x) \in M, \exists D(y) \in N, C = D \vee C \subseteq D$ is satisfiable
(2) For a given map f: variable name $\rightarrow$ the concept name that the variable belongs to, $\forall R(x, y) \in M, \exists R(z, w) \in N, (f(x) = f(z) \vee f(x) \subseteq f(z)) \wedge (f(y) = f(w) \vee f(y) \subseteq f(w))$ is satisfiable

In the definition 4, the conditions 1 is related to concepts in sets, while the condition 2 is related to the relation in sets. $C \subseteq D$ means that C is the sub-concept of D. $f(x) \subseteq f(z)$ means that f (x) is the sub-concept of f (z). In example 1, $S1.P \sqsubseteq A_0$ is because the element knows (agent, departurePlace) in S1.P can find a relation with the same name knows(agent, Beijing) in A_0 and departurePlace and Beijing belong to the same concept name DeparturePlace.

In addition, it is not difficult to find the partial order relation between services. For example, in the service for reserving air tickets, due to the need for information such as flight number, the service for searching for the flight information must first be performed. And the bank transfer needs service reserve confirmation information, so the service for reserving air tickets must be first performed.

The following are the strict formal definition of the partial order between services.

Definition 5: Given two services S_a and S_b, S_a executes before S_b denoted by $S_a < S_b$, if and only if $S_a.E \sqsubseteq S_b.P$.

In example 1, $S_1 < S_2$, $S_2 < S_3$, $S_3 < S_4$ and $S_2 < S_3$. $S_2 < S_3$ is because FlightTicketReserveConfirm is the sub-concept of ReserveConfirm.

After determining the partial order between services, the partial order diagram between the services will be formed. A web service composition corresponds to a top-down chain of the partial order. Therefore, the service composition problem is transformed to the process of finding top-down chain in the partial order diagram. In example 1, $S_1 < S_2 < S_3 < S_4$ is one target chain to find.

The basic idea of fast web service composition algorithm: The world is a collection of concepts and relations and changes with occurance of services. As a result, service composition in DDL is turned into the following: the input of users is taken as the initial state of the world A_0; select a service sequence in which the occurrence of each service will add some new elements for the world; if this sequence occurs and the arrival state set of the world contains all goal elements that users need in the set φ_0, the sequence is the service composition sequence meeting goals of users.

Let A be current state of the world and φ be the goal elements which is not achieved in the current state, according to the above analysis, there will be the following conclusions:

- The service S in the current state A can be executed if and only if $SP \sqsubseteq A$
- After execution of service S, $A = A \cup \{S.E\}$
- After execution of service S, $\varphi = \varphi - S.E$

where, $\cup$ and - are common operations on sets, A and φ have initial value A_0 and φ_0.

In order to strike a balance between the execution time of the algorithm and expression ability. This study does not use all syntax elements in DDL because it is not acceptable for the user if the response time of the system is too long. The following is the specific realization process.

First of all, given the initial state set of the world A_0 and the goal set φ_0, find the first service that can be

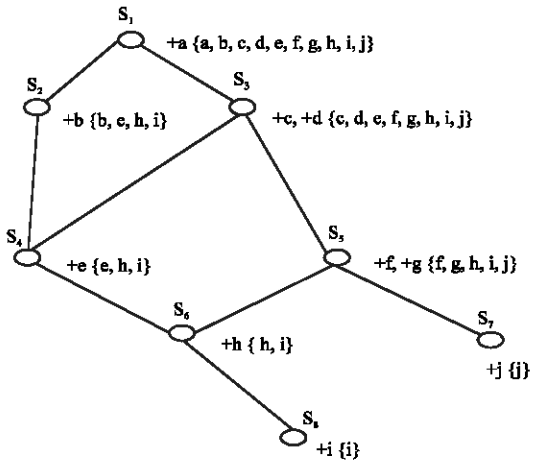

Fig. 1: The partial order diagram between services with instruction information set

executed where P ⊆ A_0 must be satisfied. After the service is selected and executed, the state of the world is changed by E of the service. And then in the service partial order diagram, find all the successor service nodes and select the service that can be executed in the new state of the world. However, there are many services that meet the needs. Which service node should be selected? In order to avoid blind search, set up instruction information set and the instruction information set points out all elements which can be gotten after executing all successor services. Only elements of the goal set φ are in instruction information set can the service be selected.

Below is a general example to illustrate the above realization process

In Fig. 1, there are eight services S_1~S_8. Each edge connects with the services with the partial order relation, the alphabets next to the services refer to the specific elements in the set. For example, the alphabets letters +c and +d next to S_3 show S_3E = {c, d}. That is, the implementation of the service S_3 will add two elements c and d to the current state. The set {c, d, e, f, g, h, i, j} is the instruction information set of S_3 indicating all elements which can be gotten after executing all successor services of S_3. As the partial order diagram has hierarchy, instruction information sets of the upper level can be obtained by merging instruction information sets of the lower level.

The fast service composition algorithm : After obtaining the partial order diagram between services, the fast service composition based on DDL can be executed.

Algorithm 1: the fast service composition algorithm based on DDL .

Input: The initial state set A_0, the goal set φ_0
Output: The service composition sequences meeting requirements of users
A = A_0; φ = φ_0;
For each S ∈Services
If (S.P ⊆ A) Then < S, A∪ {S.E}, φ- S.E > is pushed into the stack;
Endfor
Exist = false;
While (the stack is not empty) Do
Pop the top element off the stack and assign the first component to S, the second component to A and the third component to φ
If (φ is empty) Then
A service composition sequence meeting requirements of users is found
Exist = True;
Else
For each each t ∈ the successor service node set of s
If (t. P ⊆ A and φ ⊆ t. C)
Then <t, A∪ {t. E}, φ-t. E> is push into the stack;
End for
End if
End while
If (!Exist) Then there is no service composition sequence meeting requirements of users

In algorithm 1, t. C is a set of instruction information; the first component of the vector pushed into the stack represents the selected service, the second component indicates the state of the world after the service S is executed and the third component is the goal elements that are not achieved after executing the service S.

The run and time complexity analysis of the fast composition algorithm: The goal set of users can contain one or more elements. And the number of the service composition (the partial order path) which can achieve the goal can be more than one.

In the study, based on the number of the goal elements and the number of paths, service composition is divided into five classes: single-goal and single-path, single-goal and multi-path, multi-goal and single-path,multi-goal and multi-path and no path. For each class an example is selected to show the operation. Assume the constraint condition t. P ⊆ A is always true in order to highlight the effect of the instruction information in seeking paths. That is, t can always be executed in the current state. In addition, assume that only the service S_1 can be executed in the initial state A_0 in order to simplify the analysis process. The following illustrates the run of each class of service composition.

- Single-goal and single-path A = {m, n} and φ = {j}

In this case, the changes in the stack are as follows:

(1) (S_1, {m, n, a}, {j})
(2) (S_3, {m, n, a, c, d}, {j})
(3) (S_5, {m, n, a, c, d, f, g}, {j})
(4) (S_7, {m, n, a, c, d, f, g, j}, {})
(5) *Φ (note:* indicates finding a path)

The changes in the stack show the access path in the partial order diagram. In step 1, only S_1 is selected and pushed into the stack. In step 2, when accessing S_1, according to instruction information goals cannot be achieved along S_2($\varphi \not\subset S_2.C$). And goals can be achieved along S_3($\varphi \not\subset S_3.C$). Therefore, S_2 is abandoned and S_3 is pushed into the stack. In step 3, when accessing S_3, according to instruction information goals cannot be achieved along S_4($\varphi \not\subset S_4.C$). And goals can be achieved along S_5($\varphi \not\subset S_5.C$). Therefore, S_4 is abandoned and S_5 is pushed into the stack. In step 4, when accessing S_5, abandon S_6 and S_7 is pushed into the stack. In step 5, when accessing S_7, φ is empty and the goals are achieved. So, the target chain is $S_1<S_3<S_5<S_7$.

- Single-goal and multi-path A = {m, n} and φ = {i}

In this case,the changes in the stack are as follows:

(1) (S_1, {m, n, a}, {i})
(2) (S_2,{m, n, a, b}, {i}),(S_3, {m, n, a, c, d}, {i})
(3) (S_2,{m, n, a, b}, {i}), (S_4, {m, n, a, c, d, e}, {i}), (S_5, {m, n, a, c, d, f, g}, {i})
(4) (S_2,{m, n, a, b}, {i}), (S_4, {m, n, a, c, d, e}, {i}), (S_6, {m, n, a, c, d, f, g, h}, {i})
(5) (S_2, {m, n, a, b}, {i}), (S_4, {m, n, a, c, d, e}, {i}), (S_8, {m, n, a, c, d, f, g, h, i}, { })
(6) *(S_2, {m, n, a, b}, {i}), (S_4, {m, n, a, c, d, e}, {i})
(7) (S_2, {m, n, a, b}, {i}), (S_6, {m, n, a, c, d, e, h}, {i})
(8) (S_2, {m, n, a, b}, {i}), (S_8, {m, n, a, c, d, e, h, i}, {})
(9) *(S_2, {m, n, a, b}, {i})
(10) (S_4,{m, n, a, b, e}, {i})
(11) (S_6,{m, n, a, b, e, h}, {i})
(12) (S_8,{m,n,a,b,e,h,i},{})
(13) *Φ

In step 3, when accessing S_3, goals can be achieved along both S_4 and S_5 and so both S_4 and S_5 are pusehed into the stack.; In step 6, when accessing S_8, the first target chain $S_1<S_3<S_5<S_6<S_8$ is found; In step 9, the second target chain $S_1<S_3<S_5<S_6<S_8$ is found; in step 9, the third target chain $S_1<S_2<S_4<S_6<S_8$ is found.

It is shown that all target sequences that can satisfy goals of users can be found using the algorithm and there is no redundant information in the stack.

- Multi-goal and single-path A = {m, n} and φ = {b, i}

In this case, changes in stacks are as follows:

(1) (S_1, {m, n, a},{b, i})
(2) (S_2,{m, n, a, b},{i})
(3) (S_4,{m, n, a, b, e}, {i})
(4) (S_6,{m, n, a, b, e, h}, {i})
(5) (S_8, {m, n, a, b, e, h, i},{ })
(6) *Φ

In step 2, when accessing S_1, according to instruction information {b, i} can be achieved along S_2, while only {i} can be achieved along S_3. Therefore, S_3 is abandoned and S_2 is pushed into the stack. S_2 can achieve {b}. The goal set φ is reduced to {i}. In step 6, when accessing S_8, the target chain $S_1<S_2<S_4<S_6<S_8$ is found.

- Multi-goal and multi-path: A = {m, n} and φ = {d, i}

In this case, changes in stack are as follows:

(1) (S_1, {m, n, a}, {d, i})
(2) (S_3, {m, n, a, c, d}, {i})
(3) (S_4, {m, n, a, c, d, e},{i}), (S_5, {m, n, a, c, d, f, g}, {i})
(4) (S_4,{m, n, a, c, d, e}, {i }), (S_6, {m, n, a, c, d, f, g, h}, {i})
(5) (S_4,{m,n,a,c,d,e},{i }), (S_8, {m, n, a, c, d, f, g, h, i},{})
(6) *(S_4, {m, n, a, c, d, e}, {i})
(7) (S_6, {m, n, a, c, d, e, h}, {i})
(8) (S_8, {m, n, a, c, d, e, h, i}, {})
(9) *Φ

In step 3, when accessing S_3, {d} is achieved, {i} can be achieved along either S_4 or S_5 and so S_4 and S_5 are pushed into the stack. In step 6, when accessing S_8, the first target chain $S_1<S_3<S_5<S_6<S_8$ is found. In step 9, when accessing S_8, the second target chain $S_1<S_3<S_4<S_6<S_8$ is found.

- No path: A = {m, n} and φ = {i, j}

In this case, changes in stack are as follows:

(1) (S_1, {m, n, a}, {i, j})
(2) (S_3, {m, n, a, c, d}, {i, j})
(3) (S_5, {m, n, a, c, d, f, g}, {i, j})
(4) Φ

When accessing S_1 and S_3, {i, j} can be achieved according to instruction information set. In step 4, when accessing S_5, {i, j} cannot be achieved simultaneously according to instruction information set of S_6 and S_7. The stack is empty and no target chain can be found. In fact, there exists no service composition meeting requirements of users.

From the above examples it can be concluded that the algorithm can deal with all cases correctly and the validity and correctness can be guaranteed.

Here are the time complexity of the fast service composition. For the case of the single path, let h be the number of the level of the partial order diagram, N be the number of services and r be the average number of branches of nodes of the partial order diagram, $h = Log_rN$ is the time complexity of the fast service composition. For the case of the multiple paths, let N be the number of services,m be the number of target chains and r be the average number of branches of nodes of the partial order diagram, $O(N+mLog_rN)$ is the time complexity of the fast service composition.Therefore,the algorithm in this study is more efficient than the algorithm of enumerating all possible service sequences.

In the above analysis of time complexity, it is assumed that the constraint condition t. P ⊑ A is always true.In practical run,it is a condition for pruning. So, the practical time complexity is lower.

Algorithm for generating partial order diagram of services: Algorithm is built on the basis of the partial order diagram between services. Therefore, how to quickly creating the partial order diagram between services is also a very important issue. The diagram describes the static relation between services and it can be created in the registration and management server of web services. Because the service registration is dispersed, the partial order diagram can be generated in a incremental way.

Let G be a partial order diagram and r be a new arrival service. Service r needs to be added to diagram G and the relevant data need to be modified. The algorithm for generating partial order diagram between services is described below.

Algorithm 2: The algorithm for generating partial order diagram between services

For each s ∈ Services

If (r is the successor service of s) then r is added to the successor service set of s and s is added to the precursor set of r

If (r is the precursor service of s) then r is added to the precursor service set of s and s is added to the successor service set of r

Endfor

r is pushed into the stack

the updated service node set D={}

While (the stack is not empty) do

Pop the top element off the stack and assign it to s

If (the precursor service set of s is not empty) then

For each t∈ the precursor service set of s

If (t∉D) Then

t.C = t.C ∪ r.E;

D = D ∪ {t};

t is pushed into the stack;

End If

End For

End If

End while

In Fig. 1, let S_8 be the new arrival service. Two branches $S_8 \rightarrow S_6 \rightarrow S_5 \rightarrow S_3 \rightarrow S_1$ and $S_4 \rightarrow S_2$ are passed: the element i is added to the instruction information set of nodes of every branch.

CONCLUSION

In this study, a web service description method based on DDL is given and an improved service composition algorithm based on DDL is proposed. The algorithm firstly extends the inclusion operation on sets, on this basis the partial order relation between services is defined and then the partial order diagram is formed. According to the partial order diagram the fast service composition algorithm is implemented. At last an example is given in order to verify the correctness and effectiveness of the proposed algorithm. The results show that this algorithm is correct and has good adaptability and its time complexity is greatly improved.

Future work is to fully excavate the potential of dynamic description logic, continue to expand operations on sets to incorporate more syntax elements of DDL and grammatical components and seek the algorithms to support rapid web service discovery and automatic service composition.

ACKNOWLEDGMENTS

This study is supported in part by the National Natural Science Foundation of China under Grants 60773034, 90818023, 90718012 and 60803032; the Scientific and Technological Developing Program of Shandong Province of China under Grant 2008GG30001024; the National Basic Research Program of China (973 Program) under Grant 2010CB328101; the Taishan Scholar Construction Project of Shandong Province, China; the Open Project of the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of

Sciences under Grant SYSKF0804; the Research Project of SUST Spring Bud under Grant 2009AZZ165 and the Graduate Innovation Fund of Shandong University of Science and Technology under Grant YCA090320.

REFERENCES

Constantinescu, I., B. Faltings and W. Binder, 2004. Large scale, type-compatible service composition. Proceedings of the IEEE International Conference on Web Services, June 6-9, San Diego, California, pp: 506-513.

Hasany, N., A.B. Jantan, M.H.B. Selamat and M.I. Saripan, 2010. Querying ontology using keywords and quantitative restriction phrases. Inform. Technol. J., 9: 67-78.

Horrocks, I., P.F. Patel-Schneider and F.V. Harmelen, 2003. From SHIQ and RDF to OWL: The making of a web ontology language. J. Web Semantics, 1: 7-26.

Li, L. and I. Horrocks, 2003. A software framework for matchmaking based on semantic web technology. Proceedings of the 12th International Conference on World Wide Web, May 20-24, Budapest, Hungary, pp: 331-339.

Mao, Y., 2010. A semantic-based genetic algorithm for sub-ontology evolution. Inform. Technol. J., 9: 609-620.

Martin, D., M. Burstein, D. McDermott, S. McIlraith and M. Paolucci *et al*., 2007. Bringing semantics to web services with OWL-S. World Wide Web, 10: 243-277.

McIlraith, S.A. and S.C. Tran, 2002. Adapting golog for composition of semantic web services. Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning, April 22-25, Morgan Kaufmann, Toulouse, France, pp: 482-496.

Paolucci, M., T. Kawamura, T.R. Payne and K. Sycara, 2002. Semantic matching of web services capabilities. Proceedings of International Semantic Web Conference, June 9-12, Sardinia, Italy, pp: 333-347.

Peng, Y., L.M. Chen, L. Chang and Z.Z. Shi, 2008. Semantic web service matching based on dynamic description logic. J. Comput. Res. Dev., 45: 2102-2109.

Shi, Z.Z. and L. Chang, 2008. Reasoning about semantic web services with an approach based on dynamic description logics. Chinese J. Comput., 31: 1599-1611.

Shi, Z.Z., M.K. Dong, Y.C. Jiang and H.J. Zhang, 2004. A logical foundation for the semantic web. Sciencein China, 34: 1123-1138.

Sirin, E., P. Bijan, W. Dan, H.A. James and N.S. Dana, 2004. HTN planning for web service composition using SHOP2. J. Web Semantics, 1: 377-396.