Description Logic Rule Based Semantic Web Service Composition Method

Sipei LIU¹, Jin WANG¹, Xiaojie FENG¹, Heeseon PARK², Dongjoon HYUN²

¹ Samsung Electronic China Research & Development Center, Nanjing 210008, China
² Digital Media & Communications R&D Center, Samsung Electronics, Suwon, Kyonggi 441748, South Korea

Abstract
One kind of semantic web service composition method based on description logic (DL) rule is introduced in this paper. We use DL rule to figure out the hyponymy relationship between concepts in domain ontology and model web service functional semantic of ServiceProfile, and define service DL rule (R1, R2 and R3) corresponding to sequence, split and join composition structure of GroundProfile respectively. Furthermore, we propose DL rule chain and web service community (WSC) model to describe sequential and parallel service composition, and prove that the semantics of DL rule chain and WSC are all corresponding to a new R1 rule. Furthermore, we give WSC finding algorithm and WSR matching algorithm: the former is to find WSC and DL rule chain by eliminating R2 and R3 rule, the latter is to find some DL rules matching WSR as the composition result. Finally, we analyze the related work and give some experiments to validate the rightness of our method.

Keywords: Semantic Web Service; Service Composition; DL Rule; Service Community

1. Introduction
Service-oriented computing is gaining a wide acceptance, and web service is becoming a kind of more practical and potential technology. One key of the challenging problems is how to effectively discover and composite services based on their capabilities. Now following the development of semantic web technical, and using a machine understanding description for web services capabilities, it becomes a kind of possibility to accomplish the automation, intelligence, dynamic service discovery and composition [1].

The currently adopted standards for Web services (UDDI, SOAP and WSDL) do not deal with semantics information. To overcome the consequent inaccuracy of service discovery and composition, the W3C consortium promotes OWL-S for semantically describing services. An OWL–S advertisement of a service consists of the following three parts: ServiceProfile – which provides a high level description of the service, containing information such as its inputs and outputs (i.e., the functionality of the service), other extra-functional attributes as well as a further human readable service description; ProcessProfile– which describes the service behaviors providing a view of the service in terms of process compositions. OWL–S defines three types of processes (atomic processes, composite processes, simple processes); GroundProfile–which describes how to access and interact with the service by specifying protocol and message format information.

In the past several years, semantic web service composition methods have been widely studied and proposed. To the best of our knowledge, these methods can be divided into two types: one type is to use the Petri net[2][3], finite state machine[4], situation calculus[5] and process algebra[6] etc to figure the complex interaction process in web services(i.e. the temporal relationship of web service), however, a problem of undecidability still exists, because those methods are based on either proposition logic or predicate logic; the other type is to use semantic matching[7][8][9] etc to describe the service data.
dependence relationship (the semantic of one service’s output satisfy the semantic of the other service input service), then refer to the AI planning [10][11], graph searching [12] and logic reasoning [13] etc to find the composition result. However, most of these approaches like using AI planning are restricted to sequential composition and reasoning is based on Close World Assumption. So how to use the semantic information in OWL-S sufficiently to solve service composition is still an open problem.

Description logic (DL) [14] is the basis of semantic web and the sublanguage of DL (SHOIQ (D)) has been proved to be equivalent to OWL. Recently studying has showed that the DL is limited to describe the static knowledge with a tree model property, and is lack of the capability to describe the dynamic knowledge such as semantic web services. In contrast, the rule language is a good complement to reflect the dynamic characters of web services and enables an efficient inference. Furthermore, the integration of ontology and rule has become a newly hot topic. Some new rule languages based on the DL or OWL have been proposed such as SWRL [15], ALCu [16] and description logic rule [17][18] etc (we named them as DL rule), and the condition for the decidability of DL rule has been also proposed and proved [16]. So DL rule is one of the best candidates to represent and reason about semantic web service.

In this paper, we propose a kind of semantic web service composition method based on DL rule. Our research makes the following novel contributions: 1). we use DL rule to describe hyponymy relation between concepts of domain ontology and functional semantic of atomic service and give the definitions of R1, R2 and R3 rule for sequential, split and join composition structure between atomic service. 2). for semantic service composition, we propose DL rule chain and web service community (ab. WSC) model to describe sequential and parallel service composition, and prove that their semantics are all corresponding to R1 rule. 3). we gives a kind of service composition method which includes mainly WSC finding algorithm and WSR matching algorithm.

The rest of the paper is organized as follows. In the second section, we give some basic definitions of model semantic web service based on DL rule. In the second section, we analyze semantic service composition based on DL rule, and propose DL rule chain and WSC model. In the fourth section, we give WSC finding algorithm and WSR matching algorithm. In the fifth section, we generate the dataset to test these algorithms above. In the sixth section, we talk about the related work. In seventh section, we give some concluding remarks.

2. DL Rule and Semantic Web Service

2.1. DL Rule

DL rule, a decidable subset of the Semantic Web Rule Language SWRL that has been recently proposed in two independent works [17][18]. For some DL languages based on three disjoint sets of individual names Nt, concept names NC, and role names NR, let V be a countable set of first order variables. Given terms t,u ∈ Nt ∪ V, a concept (role) atom is a formula of the form Ct (Ru) with C ∈ NC (R ∈ NR). To simplify notation, we use the finite sets S of (role and concept) atoms for representing the conjunction ∧ S.

Given sets B and H of atoms, and a set x ⊆ V of all variables in B ∪ H, a description logic rule (DL rule) is a formula ∀x. B → ∧H, simply write the DL rule like B → H. The meaning of DL rule is that if the atom set B is satisfied, then the DL rule can be invoked, and atom set H can also be satisfied.

For example DL complex concepts definitions Mother ≡ Woman ∧ hasChild.child can be correspond to DL rule Woman(x) ∧ hasChild(x, y) ∧ Child(y) → Mother(x). DL Tbox axioms C ⊆ D for subclass relationship correspond to rules of the form C(x) → D(x). DL Rbox axioms R ◦ S ⊆ T express inclusions with role chains that correspond to rules of the form R(x, y) ◦ S(y, z) → T(x, z). so we can use DL rule to model the hyponymy relationship between concepts of domain ontology.
2.2. DL Rule and Semantic Web Service

We acquire the atomic service functional semantic (atomic process) and the service temporal relation (composite process) from ServiceProfile and ProcessProfile of OWL-S each other. Then we use DL rule to model them. Here in order to keep the decidability for DL rule, we suppose that all predict symbols occurring in DL rule come from the concept or role atoms, and all entities in OWL-S are all defined in domain ontology [20].

Domain ontology can be described by the form of \( Q = (C, R, I) \), where \( C \) represents Concepts, \( R \) represents Roles, and \( I \) represents instance. The hyponymy relation between concepts is corresponding to two basic relation “equivalentClass” and “subClassOf", in another word, this means two concepts satisfy \( C_x \equiv C_y \) or \( C_x \subseteq C_y \), using DL rule it can be represented as \( C_x \rightarrow C_y \).

WS\(_i\) that represents an atomic web service corresponding to one atomic process in OWL-S, can be described by the IOPE model, \( I_i \) and \( O_i \) represent input and output parameters respectively, which are mapping to the concept set in domain ontology, \( P_i \) and \( E_i \) represent precondition and effect respectively, which are the constraints on input and output parameter and can be mapped into the role set in domain ontology. So we can suppose for atomic web service \( W_i = \{I_i, O_i, P_i, E_i\} \), where \( I_i \subseteq C \), \( O_i, P_i \subseteq R \). The semantic of the input and output semantic of web service can be represented as complex concepts like (1) and (2), and web service can be seen as an atomic role involved with these two complex concepts in ontology.

\[
P_i \land I_i = \Theta p_{i1} \land \Theta p_{i2} \land \cdots \land \Theta p_{im} \land \Theta = \{\exists, \forall\}, (1)
\]

\[
E_i \land O_i = \Theta e_{i1} \land \Theta e_{i2} \land \cdots \land \Theta e_{in} \land \Theta = \{\exists, \forall\}, (2)
\]

Definitions 1. (Atomic web service).\( W_i \) that represent an atomic web service is corresponding to one atomic process in OWL-S, the atomic web service can be seen as a role in ontology, and the semantic can be described as the form: \( P_i \land I_i \rightarrow E_i \land O_i \), where \( P_i \land I_i \) and \( E_i \land O_i \), which is corresponding to complex concept in ontology, represent input semantic and output semantic of atomic web service each other.

This meaning of DL rule for web service is that if the input semantic of web service can be satisfied, then the DL rule can be invoke and the output semantic can be generated. In the same way, the semantic of web service request (ab. WSR) can be described by the DL rule as \( P_r \land I_r \rightarrow E_r \land O_r \). For the composite process of OWL-S in parallel service composition, we can give the corresponding definition of DL rule.

Definitions 2. (Service DL Rule). For the basic process structure (Sequence, Split and Join) in parallel service composition, we give three types of service DL rules:

\( R_1 \) : if the functional semantic between two atomic web services satisfies DL rule \( E_i \land O_i \rightarrow P_j \land I_j \), which represents sequence composition process, i.e. one composition service is to invoke \( W_{S_1} \) firstly, then invoke \( W_{S_2} \).

\( R_2 \) : if the functional semantic between these atomic web services satisfies DL rule \( (E_i \land O_i) \land (E_j \land O_j) \ldots \land (E_m \land O_m) \rightarrow P_j \land I_j \), which represents join composition process, i.e. before invoking \( W_{S_i} \), we need invoke parallel \( W_{S_1}, W_{S_2}, \ldots, W_{S_m} \) firstly.

\( R_3 \) : if the functional semantic between these atomic web services satisfies DL rule \( E_i \land O_i \rightarrow (P_1 \land I_1) \land (P_2 \land I_2) \ldots \land (P_n \land I_n) \), which represents split composition process, i.e. after invoking \( W_{S_i} \), we can invoke parallel \( W_{S_1}, W_{S_2}, \ldots, W_{S_n} \).

In the same way, we can define the DL rule between WSR and Web Service.

\( R_1 \) : \( WSR \rightarrow W_{S_1} \iff P_i \land I_r \rightarrow P_i \land I_j \), \( WSR \rightarrow W_{S_2} \iff E_i \land O_r \rightarrow E_i \land O_j \)

\( R_2 \) : \( WSR \rightarrow W_{S_1} \land W_{S_2} \ldots \land W_{S_n} \iff P_i \land I_r \rightarrow (P_i \land I_1) \land (P_i \land I_2) \ldots \land (P_i \land I_n) \)
According to the definition of service DL rule, it is obvious we can conclude theorem 1 and 2.

**Theorem 1.** DL rule of R3 type can be converted equivalently two multiple DL rule of R1 type.

**Proof.** R3 rule may be supposed as well

\[
WS_i \rightarrow WS_k \land WS_{k+1} \land \cdots \land WS_{k+m}
\]

\[
E_i \land O_i \rightarrow (P_k \land I_k) \land (P_{k+1} \land I_{k+1}) \land \cdots \land (P_{k+m} \land I_{k+m})
\]

\[
\neg (E_i \land O_i) \lor ((P_k \land I_k) \land (P_{k+1} \land I_{k+1}) \land \cdots \land (P_{k+m} \land I_{k+m}))
\]

\[
\neg (E_i \land O_i) \lor (P_k \land I_k) \land \neg (E_i \land O_i) \lor (P_{k+1} \land I_{k+1}) \land \cdots
\]

\[
\land (\neg (E_i \land O_i) \lor (P_k \land I_k)) \land (E_i \land O_i) \lor (P_{k+1} \land I_{k+1}) \land \cdots
\]

\[
\land (\neg (E_i \land O_i) \lor (P_k \land I_k)) \land (E_i \land O_i) \lor (P_{k+1} \land I_{k+1})
\]

\[
WS_i \rightarrow WS_k \land WS_{k+1} \land \cdots \land WS_{k+m} .
\]

**Theorem 2.** R1 rule is transitive.

**Proof.** supposing there exists two R1 rules \(WS_i \rightarrow WS_{i+1}\) and \(WS_{i+1} \rightarrow WS_{i+2}\), and we need to prove \(WS_i \rightarrow WS_{i+2}\). For ①, we can know if \(WS_i\) is invoked, then \(WS_{i+1}\) can be invoked, so for ②, we can also invoke \(WS_{i+2}\), it is obvious that ③ is satisfied.

### 3. Semantic Web Service Composition Based on DL Rule

Given a repository of semantic web services and a web service requirement (WSR), the composition problem involves automatically finding a group of services that can be composed to obtain the desired service. According to the different WSR, the service composition can be divided into two sequentially service composition and parallel service composition. The composition result can be a service chain or a directly acyclic graph [12].

For a service set \(\{WS_i\}\) and WSR, sequential service composition can be seen as finding a semantic web service chain to satisfy WSR, and can be described in a form \(WS_1, WS_2, \ldots, WS_n\), where the joint service is a semantic matching between input semantic and output semantic, and the first and the last one match with input and output semantic of WSR respectively. So based on service DL rule, we can get R1 rule such as \(WS_i \rightarrow WS_{i+1}\) (1 ≤ i ≤ n), WSR → WS, and \(WS_n \rightarrow WSR\) as shown in Figure 1. So sequential service composition is corresponding to a chain style structure composed of R1 rules, we name it DL Rule Chain.

**Definitions 3.(DL Rule Chain)** we name \(WS_1, WS_2, \ldots, WS_n\) as DL rule chain, if there is a DL rule of R1 type \(WS_i \rightarrow WS_{i+1}\) (1 ≤ i ≤ n) between any two joint services \(WS_i, WS_{i+1}\), then \(WS_i, WS_n\) is called head and tail of DL chain respectively.

So using DL rule to model semantic web service, sequentially service composition can be seen as a process to find one DL rule chain which satisfy the given WSR. Based on definition 3 and theorem 2, we can get theorem 3 easily.

**Theorem 3.** If there is a DL chain between web service \(WS_i\) and \(WS_n\), then there also exists a R1 rule \(WS_i \rightarrow WS_n\).

As you see like Figure 2, it is an example of parallel composition with sequence, split and join composition structure. For every Split+Join composition process structure, we name it web Service community (WSC) Model, which can be seen as one independent unit composed of service sets like black boxes, and for other service that are not belong to this structure. WSC can be thought of one complex service having special input and output parameters. Based on the model, we can use sequential composition method to represent the parallel composition question relying on the sequential relation of WSC, as show in
Figure 2, every ellipse represents a WSC.

![Fig.2 The Sequential Composition of WSC](image)

In fact, WSC can be thought as an atomic service set which is composed of sequence, branch and split composition structure and can be seen as an independence unit with special functional semantic. Based on DL rule, WSC is correspond to a DL rule set composed of R1, R2 and R3, and all atomic services in the premise of R2 DL rule also occur in the conclusion of some R1 DL rule, consequently, these DL rule can represent one Split+Join composition structure. We can give the formal definition of WSC model based on DL rule.

**Definitions 4.** (WSC). For atomic web service \( WS_i \), if there are R2 DL rule \( WS_k \land WS_{k+1} \land \cdots \land WS_{k+s} \rightarrow WS_m \), R3 DL rule \( WS_1 \rightarrow WS_p \land WS_{p+1} \land \cdots \land WS_{p+s} \) and R1 DL rule sets \( \{ WS_p \land WS_{k+i} / i = 0,1,\ldots,s \} \), then there is a WSC between \( WS_i \) and \( WS_m \), and \( WS_i \) and \( WS_m \) is called as the head and tail of this WSC respectively.

So using WSC, we can give the formal definition of parallel service composition:

**Definitions 5.** (Parallel Service Composition). For web service sets \( \{ WS_i \} \) and \( WSR \), parallel composition of semantic web services can be thought as discovering a chain style structure of WSC that can satisfy \( WSR \), the formal representation is \( WSC_1, WSC_2, \ldots, WSC_n \), where \( WSC_i \rightarrow WSC_{i+1} (1 \leq i \leq n-1) \), \( WSR \rightarrow WSC_1 \land WSC_n \rightarrow WSR \).

Here we will hope that parallel service composition is correspond to a WSC chain, we need to ensure that there are always two WSC to matching input semantic and output semantic of WSR, but there are two special cases that the DL rule related with WSR is R2 or R3, it can not find WSC matching WSR, as show in Figure 3.A, we give a solution for these two cases by importing two agent service based on the functional semantic of WSR. As show in Figure 3.B, we import two agent services WSRO and WSRI which use the output semantic and input semantic as the functional semantic each other.

![Fig.3 Import Agent Services](image)

4. Service Composition Method and Example Illustration

In the last section, we have talked about semantic web services composition based on DL rule chain and WSC model. So based on DL rule to model atomic web service and basic composition structure, sequential web service composition is to find DL rule chain, and parallel service composition is regarded as a kind of sequential composition based on WSC. At the same time, DL rule chain can also be seen as a simple and special WSC which just includes R1 and R3 rule. So service composition method should include two basic algorithms: one is to find WSC and DL rule chain, the other is to find one R1 rule matching WSR.
4.1. WSC Finding Algorithm

In this following section, we will study two kind of expanding form of WSC definition, and prove that the semantic of WSC is equivalent with one R1 rule. Then we give a WSC finding algorithm based on eliminating R2 and R3 rule.

According to theorem 1, we give some deduction for R3 rule and R1 rule set in WSC such as:

$$WS_t \rightarrow WS_{t+1} \land \cdots \land WS_{t+s} \Rightarrow \{WS_t \rightarrow WS_{t+l} \mid l = 0, 1, \ldots, s\} \quad (1)$$

For the R1 DL rule set in definition of WSC $$\{WS_{t+l} \rightarrow WS_{k+l} \mid l = 0, 1, \ldots, s\}$$

$$\begin{align*}
(1) \quad & \Rightarrow \{WS_t \rightarrow WS_{k+l} \mid l = 0, 1, \ldots, s\} \\
& \Rightarrow WS_k \land WS_{k+1} \land \cdots \land WS_{k+s}
\end{align*} \quad (2)$$

Based on the above deduction, we give two kind of expand form for the definition of WSC:

**Expanding 1.** If there are R3 rule $$WS_t \rightarrow WS_{k+l} \land WS_{k+1} \land \cdots \land WS_{k+s}$$ and R2 rule $$WS_k \land WS_{k+1} \land \cdots \land WS_{k+s} \rightarrow WS_m$$, Then there is a WSC between $$WS_t$$ and $$WS_m$$.

**Expanding 2.** If there is a R1 rule set $$\{WS_t \rightarrow WS_{k+l} \mid l = 0, 1, \ldots, s\}$$ and a R2 rule $$WS_k \land WS_{k+1} \land \cdots \land WS_{k+s} \rightarrow WS_m$$, Then there is a WSC between $$WS_t$$ and $$WS_m$$.

WSC represent a Split+Join composition process, so from the aspect of functional semantic, WSC is semantic related DL rule set, and can be thought as one R1 rule between the head and tail of WSC. We can give the theorem 4 and corresponding proving process.

**Theorem 4.** If there is a WSC between $$WS_t$$ and $$WS_m$$, then there exists also one R1 rule $$WS_t \rightarrow WS_m$$.

**Proof.** According to Expanding 1 of WSC definition, supposing there exist R2 and R3 rule to make one WSC between $$WS_t$$ and $$WS_m$$, we give some transforms for R2 and R3 rule:

$$\begin{align*}
(1) \quad & \Rightarrow E_i \land O_i \rightarrow (P_{i,1} \land I_{i,1}) \land (P_{i,2} \land I_{i,2}) \land \cdots \land (P_{i,m} \land I_{i,m}) \\
(2) \quad & \Rightarrow WS_t \rightarrow WS_{t+i} \land WS_{t+i+1} \land \cdots \land WS_{t+i+m} \Rightarrow W_{t+i} \land WS_{t+i+1} \land \cdots \land WS_{t+i+m} \\
& \Rightarrow (E_{i,1} \land O_{i,1}) \land (E_{i,2} \land O_{i,2}) \land \cdots \land (E_{i,m} \land O_{i,m}) \Rightarrow P_i \land I_i
\end{align*} \quad (3)$$

Based on DL rule, $$WS_t$$ can be represented as:

$$\begin{align*}
(3) \quad & \Rightarrow E_i \land O_i \rightarrow (E_{i,1} \land O_{i,1}) \land (E_{i,2} \land O_{i,2}) \land \cdots \land (E_{i,m} \land O_{i,m}) \\
& \Rightarrow WS_t \rightarrow WS_{t+i} \land WS_{t+i+1} \land \cdots \land WS_{t+i+m} \\
& \Rightarrow WS_t \rightarrow WS_{t+i} \land WS_{t+i+1} \land \cdots \land WS_{t+i+m}
\end{align*} \quad (4)$$

So on the basis of WSC and characters of DL rule above, we can talk about the discovery problem of WSC in DL rule set. Firstly, according to theorem 1, every R3 rule can be transformed into R1 rule. Secondly, semantic related R1 rules can form a DL rule chain, which is correspond to a new R1 rule according to theorem 3. Thirdly, according to expanding 2 of WSC definition and theorem 4, if one R2 rule can find some DL rule chain to match, then there will exist one WSC and can be expressed in a R1 rule between head and tail of WSC. So the problem of discovering WSC is to eliminate the R2 and R3 rule in the corresponding DL rule set and get a new R1 rule which is correspond to each WSC.

Furthermore, the elements which compose WSC can either atomic service or a WSC, as show in Figure 2, the left big WSC is composed of two little WSC. So the process to find WSC is a recursion. Firstly, just finding the basic WSC, then adding new R1 rule corresponding to basic WSC and going on find new WSC until sum of WSC is not changing. So we can give the WSC finding algorithms, as show in Figure 4.

The process of WSC finding algorithm is composed of three parts: Step 1, R3 rule is transformed into R1 rule to eliminate R3 rule. Step 2 is to find the DL rule chain in R1 rule set, and adding the corresponding new R1 rule. Step 3, for every R2 rule, trying to find the R1 rules to match it, and adding the corresponding R1 rule. And for finding DL rule chain, we just use the step1 and step 2 of this algorithm. So if we suppose that the quantity of R1, R2 and R3 rule is $$n_1$$, $$n_2$$ and $$n_3$$, R2 and R3, we can analyze to know that Time Complexity of the WSC finding algorithm is $$O((n_1+n_2)^2 \times n_3)$$. 


Algorithm WSCFinding ({DL-Rules}, {WSC})
Input: {DL-Rules}: DL rule set
Output: {WSC}: WSC set  {RuleChain}: DL rule chain set
1 [initialization]
 {RuleChain} ← null, {WSC} ← null
2 [R3 rule is transformed in to R1 rule]
For rule in {DL-Rules} DO
IF rule=WS0→WS1∩WS2∩…∩WSn Then
{DL-Rules} = {DL-Rules} ∪ {WS0→WSi|i=1,2,…,n}.
3 [Finding DL rule chain and adding the corresponding R1 rule]
For rule1, rule2 in {DL-Rules} DO
IF rule1=(WS1→WS2) and rule2=(WS2→WS3) Then
{DL-Rules} = {DL-Rules} ∪ {WS1→WS3}.
4 [for every R2 rule try to find R1 rule set to match, then generate WSC, adding corresponding new R1 rule]
Done ← False.//sign whether there is new WSC to be found
For rule in {DL-Rules} DO
IF rule=WS1∩WS2∩…∩WSn→WSs Then
IF {WS1→WSs} ⊆ {DL-Rules} && WSs ∈ {R3.header} Then
Done ← True.
WSC = rule ∪ {WS1→WSs}.
{WSC} = {WSC} ∪ WSC.//generate a WSC
{DL-Rules} = {DL-Rules} ∪ {WS1→WSs}.//adding the new R1 rule corresponding to WSC
{DL-Rules} ← delete(rule).//delete the R2 rule that have matched
5 [recursion or output]
IF (Done)
GOTO 3.//if finding a new WSC, then recursion
ELSE
Return {WSC}.
-------------------------------------------------------------------
Fig.4 WSC Finding Algorithm

4.2. WSR Matching Algorithm

After the execution of WSC finding algorithm, R2 and R3 rule can be removed, and the corresponding R1 rules to DL rule chain and WSC have been imported. So in this condition, the problem of web service composition can be seen as finding one R1 rule to match WSR. Here we give one definition of matching between WSR and R1 rule, and then we proposed one kind of service composition algorithm.

Definitions 4. (WSR Matching R1 rule): For WSR with its functional semantic is like \( P_i \land I_i \rightarrow E_r \land O_r \), and R1 rule \( WS_j \rightarrow WS_j \), if there are two DL rules between WSR and the head or tail of R1 rule such as \( P_i \land I_i \rightarrow P_i \land I_i \) and \( E_j \land O_j \rightarrow E_j \land O_j \), then we called this R1 rule matching WSR.
Based on definition 4, for \( P_i \land I_i \rightarrow P_j \land I_j \land (E_j \land O_j \rightarrow E_i \land O_i) \), we can conclude that there is a R1 rule between WSR and head of R1 rule (tail of R1 rule and WSR). and we name it as input (output) semantic matching rule of WSR. So based on this ideas, we give one algorithm to find the R1 rule which is match WSR. The process includes three steps: 1). Finding all R1 rules which belong to the input semantic matching rules of WSR. 2). Finding all R1 rules which belong to the output semantic matching rules of WSR. 3). Finding one R1 rule which belongs to the rule set of 1) and 2). If the R1 rule found is DL rule chain or WSC, there will return all related DL rule. The algorithm is showed in figure 5.

Algorithm WSRMatching ([DL-Rules], WSR)

Input: {DL-Rules}: R1 rule set
WSR: web service request
Output: {result}: one R1 rule matching WSR

1) [initialization]
{result} ← null

2) [Using WSCFinding algorithms to find WSC and DL rule chain]
{DL-Rules} = {DL-Rules} ∪ WSCFinding({DL-Rules}).

3) [Finding input and output semantic matching of WSR]
For rule in {DL-Rules} DO
If WSR→rule.head and rule is R1 Then
Var1 ← add(rule).
IF rule.tail→WSR THEN
Var2 ← add(rule).

4) [Finding r1 rule matching WSR]
FOR rule1 ∈ Var1, rule2 ∈ Var1 DO
IF rule1 = rule2 Then
IF rule1 is WSC or DL rule chain THEN
{Result} ← add(WSC or DL rule chain).
ELSE {Result} ← add(rule1).

6) [return result]
Return {Result}.

Fig.5 WSR Matching Algorithm

4.3. Example Illustration

Here we use the parallel service composition example (as show in Figure 2) to illustrate the execution process of service composition method. To be easier to illustrate, we just give the input and output parameters of web service, and suppose these parameters have been mapping to concepts of ontology. The input of WSC finding algorithm is as show in Figure 6.

The process and result of WSC finding algorithm and WSR matching algorithm is as follows:

1) Construct the DL rule to describe functional semantic of atomic web service, as show in figure 7.A.

2) The result of Step 1 to Eliminate R3 rule is as show in Figure7.B. The step of step 2 to find DL rule chain and add new R1 rule is as show in figure 7.C.
(3) Invoke recursively Step 3 to eliminate R2 rule which can form a WSC and corresponding to a R1 rule, the result of this WSC finding algorithm as show in figure 8.

(4) For WSR, invoke WSR matching algorithm, the composition result is as show in Figure 9.
5. Simulation Experiment

Due to lack of testing dataset for semantic web services based on OWL-S, and most of web service is described by WSDL, so we can not test our method directly, we construct a test dataset to have a simulation experiment and validate our WSC finding algorithm and WSR matching algorithm. The hardware and software surrounding of dataset construction and simulation experiment is such: 3.0 GHz Pentium 4 CPU, 1G RAM, Windows XP OS, MyEclipse 6.0 platform.

The ideas to construct dataset are such as: select any natural number from 1~100000 as the object to model atomic semantic web service. Select randomly several objects as premise and conclusion to construct DL rule. For example, 101—>232, 50&&320&&67—>897, 562—>90&&150&&80 is correspond to R1, R2 and R3 rule. The construction process mainly utilizes the converse direction of theorem 3 and theorem 4; firstly construct some R1 rules, then adding some R1, R2 and R3 rule which is related with R1 rule selected and make the original R1 is correspond to WSC or DL rule chain, at last, delete the original R1 rule. Repeating above steps until the quantity of dataset is to a fixed sum. In this constructing process, there are two key parameters: one is the quantity of DL rule in dataset SUM, another is the probability P of that original R1 rule selected corresponding to WSC. In order to avoid randomicity, we generate the dataset with every fixed parameter for 15 times and get the average result.

The simulation experiment includes three parts:

1. Let SUM={100,500,1000,2500,3000,5000} and P={1/3,1/2,2/3}, generate the dataset each other to test the quantity of WSC that WSC finding algorithms can find. Obviously, while the value of sum is fixed, if the value of P is more, then the dataset will include more WSC; while the value of P is fixed, if the value of SUM is more, then the dataset will generate more WSC. As show in figure 10, with the rise of value for P and SUM, WSC finding algorithm find more WSC. So this experiment validates the finding capability of WSC finding algorithm.

2. Let P=1/3. Firstly, constructing the dataset that includes some fixed quantity WSC, then add some DL rules that are not related with these WSC to assure SUM={500, 1000, 1500, 2000, 2500, 3000, 4000, 5000} each other, then using such dataset to test the execution efficiency of WSC finding algorithm, as show in Figure 11. This experiment shows that, with the rise of WSC quantity and SUM, the execution time of WSC finding algorithm is longer, but the uptrend is acceptability.

3. Let P=1/3, and set SUM={500, 1000, 1500, 2000, 2500, 3000, 4000, 5000} respectively to generate different dataset, and set the R1 rule as the different WSR which is used as the initial rule and deleted.
finally, then using such datasets to test the execution efficiency of WSR Matching algorithm, as show in Figure 12. This experiment shows that, with the rise of SUM, the execution time of WSR matching algorithm becomes a little longer, but the execution is much more efficiency.

6. Related Work

Web services composition has been active area of research recently. Most of current approaches are all using graph searching, AI plan, semantic matching, logic reasoning or theory proof etc to study sequentially service composition, and the result of which is just linear chain of services. For parallel service composition, most methods use Petri-net, situation calculus or introducing action theory to describe the parallel composition structure.

Li etc[7] introduce a kind of web service dynamic composition method based on domain ontology, use the ontology and its reasoning capability to produce a optimization web service dynamic composition graph, then based on the graph, give a efficient composition method. Fu etc[8] propose a kind of finding service chain based on the semantic matching to solve sequential service composition. In the service chain, the adjoining service is to match in the semantic, except the first service and the last service. Rama etc [10] propose a kind of method combining semantic matching and planning, using the semantic similarity as the basis to choose the service in the algorithm of planning. These three kinds of methods are all using different strategies to scale the service matching degrees and restrict to sequential service composition.

Kona etc [12] introduces a kind of automation service composition method based on directed acyclic graph and universal semantic description language (USDL), this method can find the sequential and parallel service composition, but they don’t build the connection of ontology and service. Le etc [3] propose to use product rule to model the dependence relation between atomic service, let service composition refer to reasoning Petri net. These methods enlarge service composition study to parallel service composition, but they neither consider the underlying semantic information of web service and related domain ontology, nor referring to complex composition structure in ProcessProfile of OWL-S.

Narayanan etc[5] etc use situation calculus to model atomic service, and use Petri net to figure out the composition process in DAML-S, then let service composition problem convert into the reachable of Petri net. But the reasoning is based on the level of proposition logic. Mcllrait etc [19] use situation calculus to model atomic service and composition process completely then let the service composition refer to using AI plan tool to do. These models have pay attention to model the composition process in OWL-S and regard web service as a kind of knowledge to reason, but they can not model Split+Join or Split structure, and the reasoning based on situation calculus is based on first order logic with an undecidability problem.

Shi etc [13] point that the deficiency of DL to model and reason on web service, and propose to use the dynamic description logic to model atomic service and composition process, and take service as a kind of dynamic knowledge to reason, but this method don’t also deal with Split+Join or Split structure, and this implement relies on special reasoning tools. Domenico etc [20] use SWRL to model atomic service, and covert service composition to search some rule chain to satisfy WSR, but their method is just in an initial stage, and mainly consider sequential service composition.
7. Conclusion and Future Work

We use DL rule to model several aspects that including the hyponymy relationship between concepts of domain ontology, functional semantic of atomic service in ServiceProfile, and the composition process structure in GroundProfile. Thus we can describe both the static semantics and dynamic characters of web services.

For the parallel service composition, we introduce WSC model to describe Split+Join and Split composition structure. As a result, parallel service composition can be seen as sequential composition based WSC. For the sequential service composition, we propose DL rule chain to model. At the same time, we have proposed a kind of web service composition method based on DL rule and give a group of experiments to validate our method.

Our future work includes extending DL rule to support the other composition business in OWL-S, for example repeat, choice and iterative etc. and extending the dataset constructing method to generate more truly OWL-S description by using Wordnet or other domain ontology.

References

[1] OWL-S [EB/OL], www.daml.org/services/owl-s/1.0/owl-s.html
[18] Gasse, F.; Sattler, U, and Haarslev, V. Rewriting rules into SROIQ axioms. In Poster at 21st Int. Workshop on Description Logics (DL-08)
