

An Approach to Semantic Matching of Web Services

Dingjian Chen¹, Jian Wu¹, Shuyu Li², Manfu Ma¹, Zhengguo Hu¹

¹ Computer school, Northwestern Polytechnical University,
Xi'an, Shaanxi, 710072, China

² Computer school, Shaanxi Normal University,
Xi'an, Shaanxi, 710062, China

Abstract. *One of the challenging problems that Web service technology faces is the ability to effectively discover services based on their capabilities. An approach is proposed to tackle this problem in the context of description logics. The semantic information of a Web service can be described with three factors: functional capabilities, action and non-functional attributes. We formalize these three factors matchmaking and propose a novel matchmaking algorithm that takes as input a service request Q and an ontology T of services and finds a set of services whose functional capabilities, action and non-functional attributes matching ranks are all highest. We then present a simple example to illustrate our approach to semantic matching of Web service.*

Keywords: Web services Discovery, Semantic matchmaking, Description logics.

1.0 Introduction

The Semantic Web services vision is to describe Web services' capabilities and content in an unambiguous, computer-interpretable language and improve the quality and robustness of existing tasks, such as Web service discovery and invocation. Semantic Web services will also enable a broad range of new automation tasks that humans previously performed, including automated composition, interoperation, execution monitoring, and recovery. To support this vision, Semantic Web services will provide more powerful Web service development tools that enable, among other things, automated simulation and verification of Web service properties and consistency-checking and debugging features. Examples of such efforts include DAML-S [1], WSMF [2], and METEOR-S. Work in this area is still in its infancy. Many of the objectives of the Semantic Web services paradigm, such as description of service capabilities, dynamic service discovery, and goal driven composition of Web services, have yet to be reached.

Our work focuses on the issue of dynamic discovery of Web services based on their capabilities. Dynamic service discovery is usually based on the rationale that services are selected, at run-time, based on their functional capabilities, actions and non-functional attributes. The key problem of Dynamic service discovery is the

matchmaking of Web services. Our aim is to ground the discovery process on matchmaking between a requester query and available Web service descriptions. We formalize the service matching approach based on semantic in the context of description logics (DLs) [3]. A key aspect of DLs is their formal semantics and reasoning support. DLs provide an effective reasoning paradigm for defining and understanding the structure and semantics of concept description ontologies. This is essential for providing formal foundations for the envisioned Semantic Web paradigm [4]. Indeed, DLs have heavily influenced the development of some Semantic Web ontology languages (e.g., DAML+OIL or OWL [5]). We adopt OWL-S to describe the Semantic Web services.

2.0 Principles for matchmaking

We suppose that services and requests are expressed in a DL L , equipped with a model-theoretic semantics. We suppose also that a common ontology for services and requests is established, as a TBox T in L . Now a match between a services S and a requests R could be evaluated according to T .

According to OWL-S, a Web service is constructed with service profile, process model and grounding [6]. So we give now a formal definition of Web service as definition 1. Requests have the same forms with services.

Definition 1: (*Web service*) A Web service is a tuple $S(x_1, \dots, x_n) \equiv (F_S, P_S, N_S)$ where:

- S is the name of the service,
- x_1, \dots, x_n are individual variables,
- F_S describes the functional capabilities of the service. In more detail, F_S is a tuple $F_S = (I_S, O_S, PC_S, EF_S)$ where I_S and O_S are the inputs and outputs of the service, PC_S and EF_S are the preconditions and effects of the service. All the four elements are expressed with DL formula.
- P_S describes the action of the service. In more detail, P_S is a tuple $P_S = (Pr_S, Ef_S)$ where Pr_S is

the set of preconditions and Ef_S is the set of postconditions. (Referring to DDL [7])

$-N_S$ describes the non-functional attributes of the service, and is expressed with DL formula.

Consequently, in order to check whether the service S match with the request R , at first we should check matching relations between F_S and F_R , P_S and P_R , N_S and N_R respectively, then colligate these three matching relations to get the matching relation between S with R . This framework ensures the first rule that we would like to hold for matchmaking, namely, an open-world assumption.

Rule 1 (Open-world descriptions): *The absence of a characteristic in the description of a proposal should not be interpreted as a constraint of absence. Instead, it should be considered as a characteristic that could be either refined later, or left open if it is irrelevant for the issuer of the proposal.*

Moreover, if all constraints of a request R were fulfilled by a service S , but not vice versa, then S should be among the top ranked supplies in the list of potential partners of the requester, while R should not appear at the top in the list of potential partners of the supplier.

Rule 2 (Non-symmetric evaluation): *A matchmaking system may give different evaluations to the match between a service S and a request R , depending on whether it is trying to match S with R , or R with S — i.e., depending on who is going to use this evaluation.*

Based on these two rules, we give some algorithms for matchmaking of the three factors, i.e. functional capabilities, action and non-functional attributes. From high to low we also define four class matching ranks for every factor, which are *Exact*, *Strong*, *Weak* and *Not-Match*. Obviously, the service which has highest matching rank of all factors is the best. But in generic cases, consumer would like more interesting of functional capabilities than the others, or they would so much as have no interesting in action or non-functional attributes, so we adopt the matchmaking principle like this:

(Matchmaking principle): *Let CS be the set of all candidate services. First evaluate the matching ranks of functional capabilities of all candidate service in CS . Then remark all services which have highest rank, and remove the rest which are not marked from CS . If consumer has interest in matching action or non-functional attributes, then evaluate the matching ranks of action or non-functional attributes of all service in CS . At last, choose the service which has highest rank as the most appropriate matchmaking.*

The remainder of this paper is organized as follows. Section 3 describes the formalization of semantic matching of Web service in the context of DL-based ontologies. Section 4 presents an example of the proposed service matchmaking technique. We review related work in Sect. 5 and provide concluding remarks in Sect. 6.

3.0 Matching approach

3.1 Functional capabilities matchmaking

In this section, we'll first provide the approach to the functional capabilities matchmaking of Web service. As mentioned in definition 1, the semantic of functional capabilities of Web service is described by IOPE, i.e. $F_S = (I_S, O_S, PC_S, EF_S)$. So the functional capabilities matchmaking can be reduce to respectively check the subsumption relations of IOPE between service and request. First, some definitions are presented as follows. **Definition 2:** *Let $f_a : A \rightarrow \{\Delta\}$ be a mapping from assertions to the set of concepts and roles.*

(1)

$$f_a(\varphi) = \begin{cases} \{C\}, & \text{if } \varphi = C(a) \\ \{R\}, & \text{if } \varphi = R(a, b) \\ f_a(\pi) \cup f_a(\psi), & \text{if } \varphi \text{ is an assertion which is} \\ & \text{composed with assertion } \pi \text{ and } \psi \end{cases}$$

(2) $|f_a(\varphi)|_C$ is the number of concepts in $f_a(\varphi)$, $|f_a(\varphi)|_R$ is the number of roles in $f_a(\varphi)$, and $|f_a(\varphi)| = |f_a(\varphi)|_C + |f_a(\varphi)|_R$.

Definition 3: (Reduced Set of concept) *Let \mathbb{L} be a DL, \mathbb{T} be a set of axioms in \mathbb{L} .*

(1) $A = \{A_1, \dots, A_n\}$ is called a reduced clause set if either $n=1$ or no clause subsumes the conjunction of the other clauses: $\forall 1 \leq i \leq n: A_i \not\sqsubseteq A \setminus A_i$. The set A is then called a reduced set form (RSF) of every description $C \equiv A_1 \sqcap \dots \sqcap A_n$.

(2) The RSF of description $C \equiv A_1 \sqcap \dots \sqcap A_n$ can be constructed as follows: Let $C^\Gamma = \{A_1, \dots, A_n\}$ be the set of all elementary concepts of C . For any $A_i \sqsubseteq A_j, 1 \leq i, j \leq n$, if let $C^\Gamma = C^\Gamma - A_i$ until C^Γ is a RSF, the set C^Γ is then called a general reduced set form (GRSF) of C . If contrarily let $C^\Gamma = C^\Gamma - A_j$ until C^Γ is a RSF, the set C^Γ is then called a specific reduced set form (SRSF) of C .

Definition 4: (Structural subsumption and Structural equivalent) *Let $\{A_1, \dots, A_n\}$ and $\{B_1, \dots, B_m\}$ be the RSFs of description $A \equiv A_1 \sqcap \dots \sqcap A_n$ and description $B \equiv B_1 \sqcap \dots \sqcap B_m$.*

(1) A and B are structural equivalent (denoted by $A \equiv B$) iff: $(n = m) \wedge (\forall 1 \leq i \leq n, \exists 1 \leq j, k \leq m: A_i \equiv B_j \wedge B_i = A_k)$.

(2) A and B are structural subsumption (denoted by $A \sqsubseteq B$) iff: $(n = m) \wedge (\forall 1 \leq i \leq n, \exists 1 \leq j \leq m: A_i \sqsubseteq B_j)$.

Definition 5: (the semantic of functional capabilities of Web service): *Let $S \equiv (F_S, P_S, N_S)$ be a Web service described with OWL-S, $F_S = (I_S, O_S, PC_S, EF_S)$ describes the semantic of functional capabilities of S :*

- (1) $I_S \equiv I_1 \sqcap \dots \sqcap I_n$ and $O_S \equiv O_1 \sqcap \dots \sqcap O_m$ are called input and output expression of S . $I_S^\Gamma = \{I_i, I_j, \dots\}, 1 \leq i, j \leq n$ and $O_S^\Gamma = \{O_i, O_j, \dots\}, 1 \leq i, j \leq m$ are the GRSF of input and output.
- (2) $PC_S \equiv \varphi_1 \wedge \dots \wedge \varphi_n$ and $EF_S \equiv \psi_1 \wedge \dots \wedge \psi_m$ are called precondition and effect expression of S . $f_a(PC_S) = f_a(\varphi_1) \cup \dots \cup f_a(\varphi_n)$ and $f_a(EF_S) = f_a(\psi_1) \cup \dots \cup f_a(\psi_m)$ are hold.

Intuitively, the SRSF has more semantic constraint than GRSF. Hence in order to improve recall on the premise that precision is little influenced, we weaken the semantic constraints of the service and request both, i.e. using GRSF of input and output to describe the semantic constraint.

Then we also define the matchmaking of concept as definition 6 and the matchmaking of assertion is defined as definition 7.

Definition 6: (the matchmaking of concept) In the semantic matching of web services, let A be the concept which is target, B be the concept which is candidate (or source), i.e. checking whether B can match A .

- (1) B and A are strong matchmaking (denoted by $M_S(B, A)$) iff: $B \sqsubseteq A \vee B \equiv A$.
- (2) B and A are weak matchmaking (denoted by $M_W(B, A)$) iff: $B \sqsupseteq A$. Strong matchmaking and weak matchmaking are also all called approximate matchmaking, denoted by $M(B, A)$.
- (3) B and A are not matchmaking (denoted by $M_N(B, A)$) iff: $B \not\sqsubseteq A \wedge A \not\sqsupseteq B$.

Proposition 1: In the semantic matching of web services, the matching relation between two concepts is non-symmetric.

Proof. Referring to rule 2 and definition 6, it is naturally proved.

Definition 7: (the matchmaking of assertion) In the semantic matching of web services, let φ be the assertion which is target, ψ be the assertion which is candidate (or source), i.e. checking whether can ψ match φ .

ψ and φ are matchmaking (denoted by $M_A(\psi, \varphi)$) iff: $(\downarrow f(\psi)|_C \sqsubseteq \downarrow f(\varphi)|_C) \wedge (\downarrow f(\psi)|_R \sqsupseteq \downarrow f(\varphi)|_R) \wedge (\forall \alpha \in f(\varphi), \exists \beta \in f(\psi), \beta \sqsubseteq \alpha \text{ if } \alpha \text{ and } \beta \text{ are concepts, or } \beta \equiv \alpha \text{ if } \alpha \text{ and } \beta \text{ are roles})$. Otherwise ψ and φ are not matchmaking.

Hence, we can define the functional capabilities matchmaking.

Definition 8: (the functional capabilities matchmaking) In the semantic matching of web services, let $S \equiv (F_S, P_S, N_S)$ be a Web service and $F_S \equiv (I_S, O_S, PC_S, EF_S)$, $Q \equiv (F_Q, P_Q, N_Q)$ be a request and $F_Q \equiv (I_Q, O_Q, PC_Q, EF_Q)$, ε be the concept miss error.

- (1) S has Exact functional capabilities matching relation with Q iff: $(\forall A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M_S(B, A)) \wedge$

$$(\forall A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M_S(B, A)) \wedge (\downarrow I_S \sqsupseteq \downarrow I_Q) \wedge (\downarrow O_S \sqsupseteq \downarrow O_Q) \wedge M_A(PC_S, PC_Q) \wedge M_A(EF_S, EF_Q).$$

- (2) S has Strong functional capabilities matching relation with Q iff: $(\forall A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M_S(B, A)) \wedge$

$$(\forall A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M_S(B, A)) \wedge (\downarrow I_S \sqsupseteq \downarrow I_Q) \wedge (\downarrow O_S \sqsupseteq \downarrow O_Q) \wedge M_A(PC_S, PC_Q) \wedge M_A(EF_S, EF_Q).$$

- (3) S has Weak functional capabilities matching relation with Q , if one of the following conditions comes to be true:

$$\textcircled{1} (\forall A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M(B, A)) \wedge (\exists A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M_W(B, A)) \wedge (\forall A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M_S(B, A)).$$

$$\textcircled{2} (\forall A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M_S(B, A)) \wedge (\forall A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M(B, A)) \wedge (\exists A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M_W(B, A)).$$

$$\textcircled{3} (\forall A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M(B, A)) \wedge (\exists A \in I_S^\Gamma, \exists B \in I_Q^\Gamma, M_W(B, A)) \wedge (\forall A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M(B, A)) \wedge (\exists A \in O_Q^\Gamma, \exists B \in O_S^\Gamma, M_W(B, A)).$$

$$\textcircled{4} (1 \leq \left| \left\{ A \mid A \in I_S^\Gamma, \forall B \in I_Q^\Gamma, M_N(B, A) \right\} \right| \leq \varepsilon) \wedge$$

$$(1 \leq \left| \left\{ A \mid A \in O_Q^\Gamma, \forall B \in O_S^\Gamma, M_N(B, A) \right\} \right| \leq \varepsilon).$$

- (4) If it is out of all the above situations, S has Not-Match functional capabilities matching relation with Q .

The matching directions of inputs and outputs are contrary. For the inputs matchmaking, it is better to match request's inputs with service's inputs, i.e. whether the request provides sufficient input parameters to the service. On the contrary, it's better to match service's outputs with request' outputs, namely, whether the service can produce appropriate outputs required by the request. Since the preconditions and effects of the service have little effect on functional semantics, for simplicity, in the definition of weak matching relation of functional capabilities, the matchmaking of precondition and effects have been omitted. That is, there exists weak match relation between service and request if the weak match relation of the input/output semantics has been satisfied, in spite of the match of precondition and effects.

The *Concept Miss Error* refers to the amount of missing concepts in the input semantics or in the output semantics, between the request and the service. This parameter could be assigned by users or by service matching engine according to the empirical value. In the second approach, the service matching engine could make some recommendatory conclusions (listing those services whose concept miss error is within a given scope) when it could not find any service that closely matching the request. The search recall could be adjusted by the scope parameter, obviously, the concept miss error in inverse proportion to the recall.

Thus through the DL reasoning machine, the semantics-based web service functional capabilities matchmaking procedure is composed of two steps: (1) executing the input/output semantics matching algorithm between the service and the request. (2) if the result of step 1 satisfies the strong match and precondition and effect requirements exists in the request, executing the precondition and effect matching sub-procedure. Besides,

the organization of the OWL-S base is a hierarchical structure, which classifies the OWL-S according to the domain taxonomy. And the *Domain* parameter defines the special domain that the discovery request interests. As a result, matching engine that occupies a more precise domain ontology base could only search OWL-S belonged to the same domain, which improves the matching efficiency and recall.

In the following algorithm, we denote how to implement the functional capabilities matchmaking.

Algorithm FunctionMatchmaking($Q, \varepsilon, Domain$);
input: a request $Q \equiv (F_Q, P_Q, N_Q)$ and
 $F_Q \equiv (I_Q, O_Q, PC_Q, EF_Q)$, concept miss error ε ,
the correlative *Domain* of request,
the set of candidate services
 $T = \{S_1, S_2, \dots, S_n\}$
output: a set of services whose function capabilities are matchmaking with request

function RSFMatchmaking(BM, TM)
input: target concept description BM ,
source concept description TM
output: $StrongMatch(-1)$, $WeakMatch(-2)$, $NotMatchNumber$

3.2 Action matchmaking

Action matchmaking is defined as follows.

Definition 9: (the action matchmaking) In the semantic matching of web services, let $S \equiv (F_S, P_S, N_S)$ be a Web service, $Q \equiv (F_Q, P_Q, N_Q)$ be a request.

- (1) S has Exact action matching relation with Q iff: $P_Q \equiv P_S$, i.e. $P_Q \sqsubseteq P_S, P_S \sqsubseteq P_Q$.
- (2) S has Strong action matching relation with Q iff: $P_Q \sqsubseteq P_S, P_S \not\sqsubseteq P_Q$.
- (3) S has Weak action matching relation with Q iff: $P_S \sqsubseteq P_Q, P_Q \not\sqsubseteq P_S$.
- (4) If it is out of all the above situations, S has NotMatch action matching relation with Q .

Therefore, action matchmaking of Web service depends on subsumption relation between actions of service and request. We state the following method that checks the subsumption relation between actions w.r.t. ABox and TBox.

Theorem 1: Let $\alpha = (P_\alpha, E_\alpha)$ and $\beta = (P_\beta, E_\beta)$ be actions. There are not any individual variables in α and β . We call $\alpha \sqsubseteq \beta$ iff:

- (1) For $\forall A \in P_\beta$, if A is a constant condition (no variables), A should fulfill one of these two cases at least: (i) $\exists B \in P_\alpha, B$ is a constant condition, and $A \equiv B$. (ii) $\exists B \in P_\alpha, B$ is a variable condition, and A unifies with B , i.e. there is a permutation θ such that $A = B \theta$. Otherwise, if A is a variable condition (has variables), $\exists B \in P_\alpha, B$ is a variable condition, and A unifies with B .

- (2) For $\forall \{A_1, \dots, A_k\} / B \in E_\beta$, every condition in the set of $\{A_1, \dots, A_k\} \cup \{B\}$ must follow the same rule as (1).

And then, we denote how to implement the action matchmaking in the following algorithm.

Algorithm ActionMatchmaking(Q);

input: a request $Q \equiv (F_Q, P_Q, N_Q)$

and $P_Q \equiv (Pr_Q, Ef_Q)$, the set of candidate

services $T = \{S_1, S_2, \dots, S_n\}$

output: a set of services whose action is matchmaking with request

function CheckActionSubsumption(α, β)

input: two actions α and β

output: if $\alpha \sqsubseteq \beta$, return true, else return false

3.3 Non-functional attributes matchmaking

In a sense, non-functional attributes semantics of semantic web service and QoS semantics are synonyms. The attributes that affect QoS of web service are classified into two types: common attributes including executing rate, price and network bandwidth, application-specific attributes that depend on specific application domains. QoS semantics could be described by DL formula, especially the GRSF of the concepts. In the following definition, we define the non-functional attributes matchmaking.

Definition 10: (the non-functional attributes matchmaking) In the semantic matching of web services, let $S \equiv (F_S, P_S, N_S)$ be a Web service, and $N_S = A_1 \sqcap \dots \sqcap A_n, N_S^\Gamma = \{A_i, A_j, \dots\}, 1 \leq i, j \leq n$ be the GRSF of N_S , $Q \equiv (F_Q, P_Q, N_Q)$ be a request, and $N_Q = B_1 \sqcap \dots \sqcap B_m, N_Q^\Gamma = \{B_i, B_j, \dots\}, 1 \leq i, j \leq m$ be the GRSF of N_Q , ε be the concept miss error.

- (1) S has Exact non-functional attributes matching relation with Q iff: $(\forall A \in N_Q^\Gamma, \exists B \in N_S^\Gamma, M_S(B, A) \wedge (|N_S \models N_Q|))$.
- (2) S has Strong non-functional attributes matching relation with Q iff: $(\forall A \in N_Q^\Gamma, \exists B \in N_S^\Gamma, M_S(B, A) \wedge (|N_S \not\models N_Q|))$.
- (3) S has Weak non-functional attributes matching relation with Q iff: $((\forall A \in N_Q^\Gamma, \exists B \in N_S^\Gamma, M(B, A)) \wedge (\exists A \in N_Q^\Gamma, \exists B \in N_S^\Gamma, M_w(B, A))) \vee (|A| A \in N_Q^\Gamma, \forall B \in N_S^\Gamma, M_w(B, A)) \leq \varepsilon$.
- (4) If it is out of all the above situations, S has NotMatch non-functional attributes matching relation with Q .

The following algorithm denotes the non-functional attributes matchmaking.

Algorithm NonFuncMatchmaking(Q, ε);

input: a request $Q \equiv (F_Q, P_Q, N_Q)$
and $N_Q = B_1 \sqcap \dots \sqcap B_m$, concept miss error
 ε , the set of candidate services
 $T = \{S_1, S_2, \dots, S_n\}$
output: a set of services whose non-
functional attributes are matchmaking
with request

3.4 Semantic based Web service match- making

Based on the three algorithms above, the optimal service should be the service in the intersection of the service sets calculated by these algorithms respectively. If the intersection is null, the service that owns highest matching degree is selected as the optimal service. As mentioned in the *matchmaking principle* before, in most cases, the optimal service should satisfy the following three priority-ranked matchmakings: functional capabilities matchmaking; action matchmaking; non-functional attributes matchmaking. Thus, the matching algorithm of semantic web service is presented as follows: (a) functional capabilities matching between the web service of the OWL-S library and the request Q is firstly performed, the matching result --- a service set named T contains all the services with the highest matching rank (in other words, if the highest functional capabilities matching rank in the OWL-S library is HR , then the matching rank of each service belonged to T equals to HR); (b) action matching or non-functional matching is performed according to the specific requirements in the request; (c) select the optimal service relied on the rules listed below:

- (1) prior selecting the service with the higher action matching degree if different action matching ranks exist.
- (2) prior selecting the service with the higher non-functional attributes matching rank if different non-functional attributes matching rank exist on the premise of the same action matching rank.
- (3) if two or more services have the same matching rank in all three aspects, prior selecting the service with higher reputation value (service reputation is taken into consideration in this step because the final discovery result closely relates to it which would be described in detail in [8]).

Algorithm *ServiceMatchmaking*($Q, \varepsilon, Domain$);
input: a request $Q \equiv (F_Q, P_Q, N_Q)$, concept miss error ε , the correlative *Domain* of request, the set of candidate services $T = \{S_1, S_2, \dots, S_n\}$
output: a set of services which are the most appropriate matchmaking with request Q
begin
 $service = \emptyset$;
 for every S_i in T , $1 \leq i \leq n$
 $S_i.FuncMatchdgree = NotMatch$;
 $S_i.ActionMatchdgree = NotMatch$;

```

         $S_i.NonfuncMatchdgree = NotMatch$  ;
     $T_1 = FunctionMatchmaking(Q, \varepsilon, Domain)$  ;
     $service = ChooseBestServices(T_1, FuncMatch)$  ;
    if  $P_Q$  is described then
         $T = service$  ;
         $T_2 = ActionMatchmaking(Q)$  ;
         $service = ChooseBestServices(T_2, ActionMatch)$  ;
    if  $N_Q \neq \perp$  then
         $T = service$  ;
         $T_3 = NonFuncMatchmaking(Q, \varepsilon)$  ;
         $service = ChooseBestServices(T_3, NonfuncMatch)$  ;
    if  $|service|=1$  then return  $service$ ;
     $T = service$  ;
    for every  $S_i$  in  $T$ 
         $S_i.Reputation = ComputeReputation(S_i)$  ;
         $service = ChooseBestServices(T, Reputation)$  ;
    return  $service$ ;
end

```

```

function ChooseBestServices( $T, ctype$ )
input: the set of services  $T$ , the criterion  $ctype$  for selection
output: pick out the services which have the highest matching rank of  $ctype$  from  $T$ 
begin
     $ResultSet = \emptyset$ 
     $SortByCType(T)$  ; //sort the services in a sorting order that starts with the highest value of  $ctype$  and proceeds to the lowest
     $S =$  the first element of  $T$ ;
    for every  $S_i$  in  $T$ ,  $1 \leq i \leq n$ 
        if the  $ctype$  value of  $S_i$  and  $S$  are equivalent
            then  $ResultSet = ResultSet \cup \{S_i\}$  ;
    return  $ResultSet$ ;
end

```

If the result of above algorithm is a set containing multiple services with the same matching rank and reputation, the consumer may select a service at random. Besides, to automatically invoke the web service, the Semantic Web service framework may select a random service as the target, which would achieve the same effect as the requester selects a service randomly.

4.0 Illustrating example

In this section, with an example we illustrate our approach to semantic matching of Web service. There are a request and four candidate services. Figure 1 depicts

the hierarchy of the domain ontologies about the bicycle-selling service.

Let $Q \equiv (F_Q, P_Q, N_Q)$, $F_Q \equiv (I_Q, O_Q, PC_Q, EF_Q)$,

$I_Q \doteq \geq 1VISACard \sqcap \geq 1DeliverDate$,

$O_Q \doteq (Titanium \sqcup Steel) \sqcap MountainType \sqcap$

$(Human \sqcup Electromotor) \sqcap 100To300 \sqcap \geq 1Backseat$,

$PC_Q \doteq \leq (ShopDistance, 20km) \wedge \geq (BalanceInCC, 300)$,

$EF_Q \doteq = (BalanceInCC, BalanceInCC - thePrice)$,

P_Q is not specified,

$N_Q \doteq \exists hasDeliverPeriod.2days \sqcap \geq 1HomeDelivery$.

(1) $S_1 \equiv (F_{S_1}, P_{S_1}, N_{S_1})$, $F_{S_1} \equiv (I_{S_1}, O_{S_1}, PC_{S_1}, EF_{S_1})$,

$I_{S_1} \doteq \geq 1MASTERCard \sqcap VIPMan$,

$O_{S_1} \doteq MountainType \sqcap Electromotor \sqcap More300 \sqcap \geq 1Babyseat$,

$PC_{S_1} \doteq \geq (BalanceInCC, 500)$,

$EF_{S_1} \doteq = (BalanceInCC, BalanceInCC - thePrice)$,

$N_{S_1} \doteq \geq 1HomeDelivery$.

(2) $S_2 \equiv (F_{S_2}, P_{S_2}, N_{S_2})$, $F_{S_2} \equiv (I_{S_2}, O_{S_2}, PC_{S_2}, EF_{S_2})$,

$I_{S_2} \doteq \geq 1CreditCard$,

$O_{S_2} \doteq Titanium \sqcap MountainType \sqcap (Electromotor \sqcup Human) \sqcap 100To300$,

$PC_{S_2} \doteq \leq (ShopDistance, 20km)$,

$EF_{S_2} \doteq = (BalanceInCC, BalanceInCC - thePrice)$,

$N_{S_2} \doteq \exists hasDeliverPeriod.1days$.

(3) $S_3 \equiv (F_{S_3}, P_{S_3}, N_{S_3})$, $F_{S_3} \equiv (I_{S_3}, O_{S_3}, PC_{S_3}, EF_{S_3})$,

$I_{S_3} \doteq \geq 1CreditCard \sqcap \geq 1DeliverDate$,

$O_{S_3} \doteq 28Size \sqcap Titanium \sqcap MountainType \sqcap Human \sqcap 100To300$

$\sqcap \geq 1Backseat \sqcap \geq 1Babyseat$,

$PC_{S_3} \doteq \leq (ShopDistance, 20km) \wedge \geq (BalanceInCC, 300)$,

$EF_{S_3} \doteq = (BalanceInCC, BalanceInCC - thePrice)$,

$N_{S_3} \doteq \exists hasDeliverPeriod.2days \sqcap \geq 1HomeDelivery$.

(4) $S_4 \equiv (F_{S_4}, P_{S_4}, N_{S_4})$, $F_{S_4} \equiv (I_{S_4}, O_{S_4}, PC_{S_4}, EF_{S_4})$,

$I_{S_4} \doteq \geq 1CreditCard \sqcap \geq 1DeliverDate$,

$O_{S_4} \doteq 26Size \sqcap Steel \sqcap MountainType \sqcap Electromotor$

$\sqcap 100To300 \sqcap \geq 1Backseat$,

$PC_{S_4} \doteq \leq (ShopDistance, 20km) \wedge \geq (BalanceInCC, 300)$,

$EF_{S_4} \doteq = (BalanceInCC, BalanceInCC - thePrice)$,

$N_{S_4} \doteq \geq 1HomeDelivery$.

Let concept miss error $\varepsilon = 2$. In S_1 , there are 2 concepts in I_{S_1} not be matched by I_Q , and 3 concepts in O_Q not be matched by O_{S_1} , so S_1 does not match Q in functional capabilities. There is just one concept in O_Q not be matched by O_{S_2} , so S_2 has weak functional capabilities matching relation with Q . For the rest, S_3 and S_4 all have strong matching rank with Q in the functional capabilities matchmaking. Also because P_Q is not specified, we should next perform non-functional attributes matchmaking between S_3 , S_4 and Q . It is easy to see that S_3 has exact matching rank, whereas

S_4 only has strong matching rank in non-functional attributes matchmaking. Finally, the optimal service which has highest matching rank with Q is S_3 .

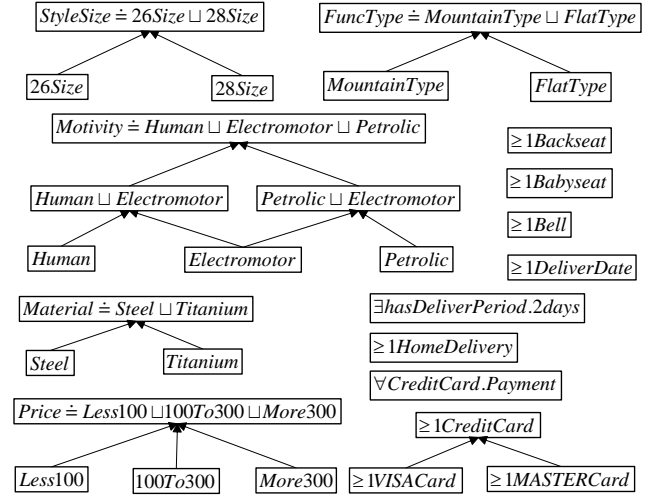


Fig. 1. The hierarchy of the domain ontologies about the bicycle-selling service.

5.0 Related work

Current Web services infrastructure, for example UDDI, provides limited search facilities allowing only a keyword-based search of services. To cope with this limitation, emerging approaches rely on Semantic Web technology to support service discovery [9]. For example, Chakraborty et al. [10] define an ontology based on DAML [11] to describe mobile devices and propose a matching mechanism that locates devices based on their features (e.g., a type of a printer). [12] proposes to use process ontologies to describe the behavior of services and then to query such ontologies using a process query language (PQL). There are other approaches based on a DAML+OIL [13] description of services that propose to exploit the DL-based reasoning mechanisms. An experience in building a matchmaking prototype based on a DL reasoner that considers DAML+OIL-based service descriptions is reported by [14]. The proposed matchmaking algorithm is based on simple subsumption and consistency tests. A more sophisticated matchmaking algorithm between services and requests described in DAML-S is proposed by [9]. The algorithm considers various degrees of matching that are determined by the minimal distance between concepts in the concept taxonomy. [15] presents an approach to tackling this problem in the context of description logics. They formalize service discovery as a new instance of the problem of rewriting concepts using terminologies which is called the best covering problem. Also a novel matchmaking algorithm is provide to find a set of services called a “best cover” of request whose descriptions contain as much common information with request as possible and as little extra information with respect to request as possible.

6.0 Conclusion

We have addressed the problem of matchmaking of web services from a knowledge representation perspective. We have presented suitable definitions of the problem. The semantic information of a Web service can be described with three factors: functional capabilities, action and non-functional attributes. We formalize these three factors matchmaking and propose a novel matchmaking algorithm. We proposed match categorization in terms of *Exact* match, *Strong* match, *Weak* match and *Not-Match* and rank of matches within categories. At last, we have presented a simple example to illustrate our approach. Our future work will be devoted to perform the more precise semantic matching of Web service, i.e. quantify the matchmaking degree in order to improve precision and recall of the service discovery.

7.0 References

- [1] DAML Services Coalition (2002) DAML-S: Web service description for the Semantic Web. In: Proceedings of the 1st international Semantic Web conference (ISWC), Sardinia, Italy, June 2002, pp 348–363
- [2] Fensel D, Bussler C, Maedche A (2002) Semantic Web enabled Web services. In: Proceedings of the international Semantic Web conference, Sardinia, Italy, June 2002, pp 1–2
- [3] Donini F, Schaerf A, Lenzerini M, Nardi D (1996) Reasoning in description logics. In: Brewka G (ed) Foundation of knowledge representation. CSLI- Publications, Stanford, CA, pp 191–236
- [4] Horrocks I, Patel-Schneider PF, van Harmelen F (2002) Reviewing the design of DAML+OIL: an ontology language for the Semantic Web. In: Proceedings of the 18th national conference on artificial intelligence (AAAI 2002), Edmonton, Alberta, Canada, 28 July–1 August 2002, pp 792–797
- [5] World Wide Web Consortium (2003) <http://www.w3.org/2001/sw/webont/>
- [6] World Wide Web Consortium (2004) <http://www.w3.org/Submission/OWL-S/>
- [7] Zhongzhi Shi, Mingkai Dong et al. (2004) The logic base of Semantic Web service. Science in china series E: Information Sciences. 2004 Vol.34 No.10, pp1123–1138
- [8] Dingjian Chen, Jian Wu et al. (2005) Design and implementation of semantic Web service reputation model. Journal of computer applications, China 2005 Vol.25 No.8, pp1878–1880
- [9] Paolucci M, Kawamura T, Payne TR, Sycara KP (2002) Semantic matching of Web services capabilities. In: Proceedings of the international Semantic Web conference, Sardinia, Italy, June 2002, pp 333–347
- [10] Chakraborty D, Perich F, Avancha S, Joshi A (2001) DReggie: semantic service discovery for M-Commerce applications. In: Proceedings of the workshop on reliable and secure applications in mobile environment, 20th symposium on reliable distributed systems, New Orleans, October 2001, pp 28–31
- [11] DAML Services. <http://www.daml.org/services/>
- [12] Bernstein A, Klein M (2002) Discovering services: towards high precision service retrieval. In: Proceedings of the CaiSE workshop on Web Services, e-business, and the Semantic Web: foundations, models, architecture, engineering and applications, Toronto, May 2002
- [13] Horrocks I (2002) DAML+OIL: a reasonable Web ontology language. In: Proceedings of EDBT'2002, Prague, Czech Republic, March 2002, pp 2–13
- [14] González-Castillo J, Trastour D, Bartolini C (2001) Description logics for matchmaking of services. In: Proceedings of the KI-2001 workshop on applications of description logics, Vienna, Austria, September 2001. <http://sunsite.informatik.rwthachen.de/Publications/CEUR-WS/Vol-44/>
- [15] B. Benatallah et al. (2004) On automating Web services discovery. VLDB Journal (2004) / Digital Object Identifier (DOI) 10.1007/s00778-003-0117-x