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 $S(x_1,\ldots,x_n) = (F_s, P_s, N_s)$ where:
- $S$ is the name of the service,
- $x_1,\ldots,x_n$ are individual variables,
- $F_s$ describes the functional capabilities of the service. In more detail, $F_s = (I_s, O_s, PC_s, EF_s)$ 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 $(Pr_s, Ef_s)$ where $Pr_s$ is
the set of preconditions and $E_f$ 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. $F_s = (I_s, O_s, P_{CEF_s})$. 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 $f_s : A \rightarrow \{\Delta\}$ be a mapping from assertions to the set of concepts and roles.

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

(2) $|f_s(\varphi)|_c^l$ is the number of concepts in $f_s(\varphi)$, $|f_s(\varphi)|_r$ is the number of roles in $f_s(\varphi)$, and $|f_s(\varphi)|_c + |f_s(\varphi)|_r$.

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

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

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

Definition 4: (Structural subsumption and Structural equivalent) Let $\{A_1, \ldots, A_n\}$ and $\{B_1, \ldots, B_m\}$ be the RSFs of description $A = A_1 \cap \cdots \cap A_n$ and description $B = B_1 \cap \cdots \cap B_m$.

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

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

Definition 5: (the semantic of functional capabilities of Web service): Let $S = (F_s, P_s, N_s)$ be a Web service described with OWL-S, $F_s = (I_s, O_s, P_{CEF_s})$ describes the semantic of functional capabilities of $S$:
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 \( M_s(A, B) \)) if: \( B \subseteq A \lor A = B \).

(2) \( B \) and \( A \) are weak matchmaking (denoted by \( M_w(A, B) \)) if: \( B \sqsupseteq A \). 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 \( M_n(A, B) \)) if: \( B \not\sqsubseteq A \land A \not\sqsubseteq B \).

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 6: (the matching 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 \( M_s(A, B) \)) if: \( B \subseteq A \lor A = B \).

(2) \( B \) and \( A \) are weak matchmaking (denoted by \( M_w(A, B) \)) if: \( B \sqsupseteq A \). 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 \( M_n(A, B) \)) if: \( B \not\sqsubseteq A \land A \not\sqsubseteq B \).

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 matching of assertion) In the semantic matching of web services, let \( \phi \) be the assertion which is target, \( \psi \) be the assertion which is candidate (or source), i.e. checking whether \( \psi \) can match \( \phi \).

\( \psi \) and \( \phi \) are matchmaking (denoted by \( M_s(\phi, \psi) \)) if:

\[
\langle \phi(\psi) \rangle_s \equiv \mathbf{f}(\mathbf{f}(\phi \vert \psi)) \land (\forall \alpha \in f(\phi), \exists \beta \in f(\psi), \beta \subseteq \alpha \text{ if } \alpha \text{ and } \beta \text{ are concepts}, \text{or } \beta \equiv \alpha \text{ if } \alpha \text{ and } \beta \text{ are roles}).
\]

Otherwise \( \psi \) and \( \phi \) 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 \( S = (F_s, P_s, N_s) \) be a Web service and \( F_s = (I_s, O_s, \text{PC}_s, \text{EF}_s) \), \( Q = (F_0, P_0, N_0) \) be a request and \( F_0 = (I_0, O_0, \text{PC}_0, \text{EF}_0) \). \( \varepsilon \) be the concept miss error.

(1) \( S \) has Exact functional capabilities matchmaking relation with \( Q \) if:

\[
(\forall A \in I_s, \exists B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_s(B, A)) \land M_s(\text{EF}_s, \text{EF}_0).
\]

(2) \( S \) has Strong functional capabilities matching relation with \( Q \) if:

\[
(\forall A \in I_s, \exists B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_s(B, A)) \land M_s(\text{EF}_s, \text{EF}_0).
\]

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

1. \( (\forall A \in I_s, \exists B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_n(B, A)) \land (\forall A \in I_s, B \in I_0, M_n(B, A)) \land M_s(\text{EF}_s, \text{EF}_0).
\]

2. \( (\forall A \in I_s, \exists B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_n(B, A)) \land (\forall A \in I_s, B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_n(B, A)) \land M_s(\text{EF}_s, \text{EF}_0).
\]

3. \( (\forall A \in I_s, \exists B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_s(B, A)) \land (\forall A \in I_s, B \in I_0, M_n(B, A)) \land (\forall A \in I_s, B \in I_0, M_n(B, A)) \land M_s(\text{EF}_s, \text{EF}_0).
\]

4. \( (1 \leq \|A \mid A \in I_s, \forall B \in I_0, M_s(B, A)] \leq \varepsilon \).
the organization of the OWL-S base is a hierarchical structure, which classifies the OWL-S according to the domain taxoogy. 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 FunctionCapabilitiesMatchmaking(Q, ε, Domain);
input: a request \( Q=(F_0, P_0, N_0) \) and \( F_0=(I_0,O_0,PC_0,EF_0) \), concept miss error \( ε \), the correlative Domain of request, the set of candidate services \( T=\{S_1,S_2,...,S_n\} \);
output: StrongMatch(−2), 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 \( S=(F_s, P_s, N_s) \) be a Web service, \( Q=(F_0, P_0, N_0) \) be a request.

(1) \( S \) has Exact action matching relation with \( Q \) iff: \( P_0 \equiv P_s \), i.e. \( P_0 \subseteq P_s \), \( P_s \subseteq P_0 \).

(2) \( S \) has Strong action matching relation with \( Q \) iff: \( P_0 \cap P_s \neq \emptyset \).

(3) \( S \) has Weak action matching relation with \( Q \) iff: \( P_0 \subseteq P_s \), \( P_s \cap P_0 \neq \emptyset \).

(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 theorem 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 \( \alpha \) and \( \beta \). We call \( \alpha \sqsubseteq \beta \) iff:

(1) For \( \forall A \in P_\alpha \), if \( A \) is a constant condition (no variables), \( A \) should fulfill one of these two cases at least: (i) \( \exists B \in P_\beta \), \( B \) is a constant condition, and \( A=B \). (ii) \( \exists B \in P_\beta \), \( B \) is a variable condition, and \( A \) unifies with \( B \), i.e. there is a permutation \( \theta \) 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 \( \forall \{A_1,\cdots,A_n\}/B \in E_\beta \), every condition in the set of \( \{A_1,\cdots,A_n\} \cup \{B\} \) 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 \( Q=(F_0, P_0, N_0) \) and \( P_0=(PR_0,EF_0) \), the set of candidate services \( T=\{S_1,S_2,...,S_n\} \);
output: a set of services whose action is matchmaking with request \( Q \).

function CheckActionSubsumption(\( \alpha , \beta \) )
input: two actions \( \alpha \) and \( \beta \).
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 \( S=(F_s, P_s, N_s) \) be a Web service, and \( N_s=A_1 \cap \cdots \cap A_m \), \( N_s^f=\{A_i,A_j,...,A_n\},1\leq i,j \leq n \) be the GRSF of \( N_s \), \( Q=(F_0, P_0, N_0) \) be a request, and \( N_0=B_1 \cap \cdots \cap B_m \), \( N_0^f=\{B_i,B_j,...,B_m\},1\leq i,j \leq m \) be the GRSF of \( N_0 \), \( \varepsilon \) be the concept miss error.

(1) \( S \) has Exact non-functional attributes matching relation with \( Q \) iff: \( (\forall A \in N_0^f, \exists B \in N_s^f, \\varepsilon M_s(B,A)) \land (|N_s|-|N_0|) \).

(2) \( S \) has Strong non-functional attributes matching relation with \( Q \) iff: \( (\forall A \in N_0^f, \exists B \in N_s^f, \varepsilon M_s(B,A)) \land (|N_s|-|N_0|) \).

(3) \( S \) has Weak non-functional attributes matching relation with \( Q \) iff: \( ((\forall A \in N_0^f, \exists B \in N_s^f, M_s(B,A)) \land (\exists \Omega \in N_0^f, \exists B \in N_s^f, M_s(B,A)) \land (|A| A \in N_0^f \exists \in N_s^f, M_s(B,A) \leq \varepsilon) \).

(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 NonFuncCapabilitiesMatchmaking(Q, ε);
input: a request \( Q = (F_0, P_0, N_0) \) and \( N_0 = B_1 \cap \cdots \cap B_n \), concept miss error \( \varepsilon \), the set of candidate services \( T = \{S_1, S_2, \ldots, S_n\} \) output: a set of services whose non-functional attributes are matchmaking with request

3.4 Semantic based Web service matchmaking

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 matchmaking: functional capabilities matching; action matching; non-functional attributes matching. Thus, the matching algorithm of semantic web service is presented as follows: (a) functional capabilities matchmaking 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 \((Q, \varepsilon, \text{Domain})\); input: a request \( Q = (F_0, P_0, N_0) \), concept miss error \( \varepsilon \), the correlative \( \text{Domain} \) of request, the set of candidate services \( T = \{S_1, S_2, \ldots, S_n\} \) output: a set of services which are the most appropriate matchmaking with request \( Q \)

begin
\( S_i, \text{FuncMatchdgree} = \text{NotMatch} \);
\( S_i, \text{ActionMatchdgree} = \text{NotMatch} \);
\( S_i, \text{NonFuncMatchdgree} = \text{NotMatch} \);
\( T_1 = \text{FunctionMatchmaking}(Q, \varepsilon, \text{Domain}) \);
\( \text{service} = \text{ChooseBestServices}(T_1, \text{FuncMatch}) \);
if \( P_0 \) is described then
\( T = \text{service} \);
\( T_2 = \text{ActionMatchmaking}(Q) \);
\( \text{service} = \text{ChooseBestServices}(T_2, \text{ActionMatch}) \);
if \( N_0 \neq \perp \) then
\( T = \text{service} \);
\( T_3 = \text{NonFuncMatchmaking}(Q, \varepsilon) \);
\( \text{service} = \text{ChooseBestServices}(T_3, \text{NonFuncMatch}) \);
if |\( \text{service} | = 1 \) then return \( \text{service} \);
\( T = \text{service} \);
for every \( S_i \) in \( T \)
\( S_i, \text{Reputation} = \text{ComputeReputation}(S_i) \);
\( \text{service} = \text{ChooseBestServices}(T, \text{Reputation}) \);
return \( \text{service} \);
end

function ChooseBestServices \((T, \text{ctype})\); input: the set of services \( T \), the criterion \( \text{ctype} \) for selection output: pick out the services which have the highest matching rank of \( \text{ctype} \) from \( T \)
begin
\( \text{ResultSet} = \emptyset \); //sort the services in a sorting order that starts with the highest value of \( \text{ctype} \) and proceeds to the lowest
\( S = \) the first element of \( T \); for every \( S_i \) in \( T \), if \( S_i \) is
if the \( \text{ctype} \) value of \( S_i \) and \( S \) are equivalent then \( \text{ResultSet} = \text{ResultSet} \cup \{ S_i \} \);
return \( \text{ResultSet} \);
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 \( Q = (F_P, F_O, N_Q) \), \( F_P = (I_O, O_Q, PC, EF_P) \),
\( I_O \geq \text{VISACard} \), \( \sqcap \text{DeliverDate} \),
\( O_Q \geq (\text{Titanium} \sqcap \text{Steel}) \sqcap (\text{Human} \sqcap \text{Electromotor}) \),
\( \sqcap \text{1100To300} \), \( \sqcap \text{1Backseat} \),
\( PC \geq (\text{ShopDistance}, 20km) \), \( \geq (\text{BalanceInCC}, 300) \),
\( EF \geq (\text{BalanceInCC}, \text{BalanceInCC} - \text{thePrice}) \),
\( P_O \) is not specified,
\( N_Q \geq \exists \text{hasDeliverPeriod} \) \( \Rightarrow \) \( \exists \text{1HomeDelivery} \).

(1) \( S_1 = (F_{S_1}, F_O, N_{S_1}) \), \( F_{S_1} = (I_{S_1}, O_{S_1}, PC_{S_1}, EF_{S_1}) \),
\( I_{S_1} \geq \text{1CreditCard} \), \( \sqcap \text{1VIPMan} \),
\( O_{S_1} \geq (\text{Titanium} \sqcap \text{Steel}) \), \( \sqcap (\text{Human} \sqcap \text{Electromotor}) \),
\( \sqcap \text{1100To300} \), \( \sqcap \text{1Backseat} \),
\( PC_{S_1} \geq (\text{ShopDistance}, 500) \),
\( EF_{S_1} \geq (\text{BalanceInCC}, \text{BalanceInCC} - \text{thePrice}) \),
\( N_{S_1} \geq \exists \text{1HomeDelivery} \).

(2) \( S_2 = (F_{S_2}, F_O, N_{S_2}) \), \( F_{S_2} = (I_{S_2}, O_{S_2}, PC_{S_2}, EF_{S_2}) \),
\( I_{S_2} \geq \text{1CreditCard} \), \( \sqcap \text{1VIPMan} \),
\( O_{S_2} \geq (\text{Titanium} \sqcap \text{Steel}) \), \( \sqcap (\text{Human} \sqcap \text{Electromotor}) \),
\( \sqcap \text{1100To300} \), \( \sqcap \text{1HomeDelivery} \),
\( PC_{S_2} \geq (\text{ShopDistance}, 20km) \), \( \leq (\text{BalanceInCC}, 300) \),
\( EF_{S_2} \geq (\text{BalanceInCC}, \text{BalanceInCC} - \text{thePrice}) \),
\( N_{S_2} \geq \exists \text{3hasDeliverPeriod} \) \( \Rightarrow \) \( \exists \text{2HomeDelivery} \).

(3) \( S_3 = (F_{S_3}, F_O, N_{S_3}) \), \( F_{S_3} = (I_{S_3}, O_{S_3}, PC_{S_3}, EF_{S_3}) \),
\( I_{S_3} \geq \text{1CreditCard} \), \( \sqcap \text{11HomeDelivery} \),
\( O_{S_3} \geq (\text{Steel} \sqcap \text{1HomeDelivery}) \), \( \sqcap \text{11Backseat} \),
\( PC_{S_3} \geq (\text{ShopDistance}, 20km) \), \( \geq (\text{BalanceInCC}, 300) \),
\( EF_{S_3} \geq (\text{BalanceInCC}, \text{BalanceInCC} - \text{thePrice}) \),
\( N_{S_3} \geq \exists \text{1HomeDelivery} \).

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