Evaluating Similarity and Difference in Service Matchmaking *

Devis Bianchini[1] Valeria De Antonellis[1] Michele Melchiori[1]

Università degli Studi di Brescia - Dip. di Elettronica per l'Automazione
Via Branze, 38 - 25123 Brescia - Italy
{bianchin|deantone|melchior}@ing.unibs.it

Abstract. Recently, enterprise interoperability has been improved by the Web Service technology, making available an ever-growing number of services. Service discovery is considered a crucial issue; in particular, flexibility of the discovery process, that is, the ability of recognizing not only exact matches between the requests and offers, but also partial ones, should be enhanced. We propose a composite approach to flexible service matchmaking focused on different matching models that are able to evaluate similarity and difference between offers and requests. The approach is based on an ontological framework adding semantics to service descriptions. Optimization and ranking techniques are provided.

1 Introduction

Recently, enterprise interoperability in distributed environments has been improved by adopting the emerging Web Service standards and technology, making available on the net an ever-growing number of services. Therefore, searching for specific service capabilities, many services can be found able to fully or partially satisfy the request. As a consequence, advanced service discovery approaches have been developed in order to find the best suitable offers for a given request. In particular, flexibility of the discovery process, that is, the ability of recognizing not only exact matches, but also partial ones by evaluating the degree of match between the request and each offer, should be considered. To be effective, matchmaking techniques should rely on semantic characterization of requested and provided services, obtained by exploiting typical Semantic Web tools such as the ontologies. Other aspects related to service matchmaking techniques are the definition of optimization strategies to make more efficient the discovery process and ranking criteria to allow for ordering the discovered services on the basis of their capability to satisfy the request.

In this paper, we propose a flexible service matchmaking approach characterized in terms of four components: matching model, metrics, ranking and optimization. We consider different matching models that are able to take into

* This work has been partially supported by the ESTEEM PRIN Project funded by the Italian Ministry of Education, University and Research, and by NoE INTEROP IST Project n. 508011 - 6th EU Framework Program.

account and to analyze similar or different elements between the request and the offer, performing a flexible service comparison: (i) a similarity-based approach, exploiting retrieval metrics to measure the degree of match between services, (ii) a novel deductive approach, that is able to find the missing information among the request and each offer by applying a logic-based difference operator, and (iii) their possible combination into a hybrid approach. The matchmaker is based on an ontological framework adding semantics to service descriptions. Finally, optimization and ranking techniques are defined.

The paper is organized as follows: in Section 2 the ontological framework is presented, while in Section 3 the flexible matchmaker is described in detail; related work are discussed in Section 4 and Section 5 concludes the paper.

2 Service description and the ontological framework

Looking for a service, a user is interested *in primis* in what a service is able to provide him and secondarily on how the service goal is achieved or what he must provide to the service in order to enable its execution [10]. According to this consideration, a request is expressed by listing the expected capabilities from the service. For what concerns the service description, we want to maintain full backward compatibility with existing standards. The description of service capabilities is then extracted from the WSDL interface of services (without requiring additional efforts for semantic annotation of service descriptions or for adding new description elements such as those performed by the OWL-S coalition [6] or by the WSMO task group [10]).

Starting from the WSDL document, the expected capabilities of the service can be identified in the concepts related to the names of operations provided by the service and in the concepts related to the names of output parameters associated to operations. These elements are automatically extracted from the WSDL document together with service category and a Description Logic expression is built to represent a service. The DL expression is a conjunction of:

- a concept in the form $\exists\texttt{hasCategory}.CAT$, where CAT is a concept which represents the associated service category;
- one or more concepts in the form $\exists\texttt{hasOperation}.OP$, where OP is a concept described as a conjunction of:
  - an atomic concept representing the name of the service operation;
  - a conjunction of one or more concepts $\exists\texttt{hasOutput}.OUT$, where OUT is a concept representing an output parameter of the operation and can be defined as an atomic concept, an enumeration $\{o_1, o_2, \ldots o_n\}$ of nominals or a complex concept obtained by applying the *intersection* operator ($\sqcap$), the *union* operator ($\sqcup$) and the *negation* operator ($\neg$).

Example 1. We consider a running example in the domain of geographic information services, where are required services displaying maps with different kinds of information such as aerial photos, streets, gas pipes, water pipes and so on. Let consider the following Description Logic expressions representing a request and two offers `DisplayGasInfrastructure` and `DisplayTransportInfrastructure`:

$$
\begin{aligned}
\texttt{request} \equiv\ & \exists\texttt{hasCategory.GeographyInformationService} \sqcap \\
& \exists\texttt{hasOperation.(viewDetailedMap} \sqcap \\
& \quad \exists\texttt{hasOutput.gasPipes} \sqcap \\
& \quad \exists\texttt{hasOutput.waterPipes} \sqcap \\
& \quad \exists\texttt{hasOutput.urbanStreets)} \\
\\
\texttt{DisplayGasInfrastructure} \equiv\ & \exists\texttt{hasCategory.GeographyInformationService} \sqcap \\
& \exists\texttt{hasOperation.(displayMoreGrainedMap} \sqcap \\
& \quad \exists\texttt{hasOutput.gasPipes} \sqcap \\
& \quad \exists\texttt{hasOutput.streets)} \\
\\
\texttt{DisplayTransportInfrastructure} \equiv\ & \exists\texttt{hasCategory.GeographyInformationService} \sqcap \\
& \exists\texttt{hasOperation.(viewDetailedMap} \sqcap \\
& \quad \exists\texttt{hasOutput.gasPipes} \sqcap \\
& \quad \exists\texttt{hasOutput.transportWays)}
\end{aligned}
$$

□

To improve the effectiveness and the efficiency of service matchmaking, additional semantics is associated to service description. In particular, an ontological framework is defined for semantic enrichment [3]. The ontological framework is constituted by three components: (i) a Domain Ontology $\mathcal{D}om\mathcal{ONT}$, used to conceptualize the domain knowledge related to the names of elements used in service descriptions (operation names and output parameters), expressing it in terms of *concepts* and *semantic relationships* between them; (ii) a Thesaurus $\mathcal{TH}$ (automatically derived from available lexical systems such as WordNet), used to relate names of concepts of the Domain Ontology to other terms by means of terminological relationships (e.g., *synonymy*, *hypernymy*, etc.), in order to extend matching possibilities between the concept names used in the service request and the names used in the descriptions of provided services; (iii) a Service Ontology $\mathcal{S}erv\mathcal{ONT}$, that organizes services on three layers of abstraction; in the middle layer we have the *Abstract services*, that are introduced to summarize the functionalities of sets of similar *Concrete services*; these ones are positioned in the lower layer of the ontology and are directly invocable services that implement the functionalities represented by Abstract ones; finally, at the top layer of the ontology, with the higher abstraction, we have *Subject Categories*, that organize Abstract services into standard available taxonomies to provide a topic-driven access to them. Abstract services can also be organized into generalization hierarchies: we say that an Abstract service *is a specialization of* another one if it provides at least the same outputs and performs the same capabilities or provides/performs more specific outputs/capabilities.

Example 2. Figure 1 shows a portion of the ontological framework in the domain of geographic information services, where are represented the two advertised services considered in the Example 1. □

3 Service matchmaker

The service matchmaker is characterized by four components.

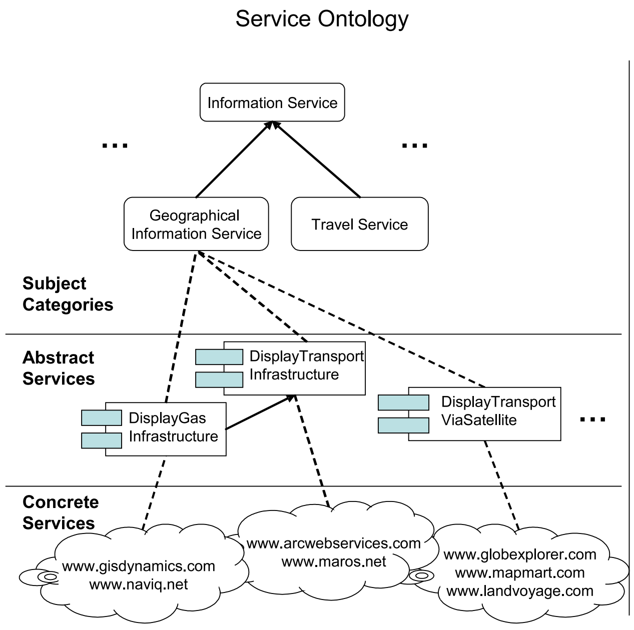


Fig. 1. A portion of ontological framework for geographic information service domain.

Matching model - We consider (i) a similarity-based model, where retrieval metrics are applied to measure the degree of match between services, (ii) a deductive model, exploiting deduction algorithms for reasoning on service descriptions and (iii) a hybrid model, that combines the similarity-based and the deductive models; in particular, we define a new deductive model for reasoning on service descriptions by difference evaluation.

Metrics - Different metrics are introduced to compute similarity between request and offers.

Ranking - A ranking scheme is defined to quantify the established degree of match between the service request and each suitable offer.

Optimization - An optimization policy is used to reduce the number of comparisons to be performed during the matchmaking process.

3.1 The similarity-based matching model

A natural way of matching is to look for similarity. For this matching model, we experimented in [4] that it can be sufficient to exploit terminological relationships to compare service descriptions. The thesaurus $\mathcal{TH}$ is therefore used to compute the *Affinity* coefficient between names of output parameters and operations by measuring their terminological relationships. According to the name affinity values, the similarity between a service request $\mathcal{R}$ and an offer $\mathcal{S}$ is computed through two properly defined coefficients: an *entity-based similarity* coefficient ($ESim$), that measures how much the compared service descriptions provide the same outputs, and a *functionality-based similarity* coefficient ($FSim$), that aims at measuring how much the two services provide the same capabilities. These coefficients are based on the Dice's information retrieval formula and are tailored to compare service descriptions presented in Section 2. They are normalized into the range [0,1] and combined together to give a global evaluation of service similarity:

$$\begin{aligned} GSim(\mathcal{R},\mathcal{S}) = w_1 \cdot NormESim(\mathcal{R},\mathcal{S}) + \\ +w_2 \cdot NormFSim(\mathcal{R},\mathcal{S}) \in [0,1] \end{aligned} \tag{1}$$

Weights w_1 and w_2, with w_1, $w_2 \in [0,1]$ and $w_1 + w_2 = 1$, are introduced to assess the relevance of each kind of similarity in computing the global similarity coefficient.

Example 3. Considering the request and the advertisements in the Example 1, if we evaluate the values for the global similarity coefficient through the formulas presented in [4], we obtain $GSim(\texttt{request}, \texttt{DisplayGasInfrastructure}) = 0.5 * [0.72+0.86] = 0.79$ and $GSim(\texttt{request}, \texttt{DisplayTransportInfrastructure}) = 0.5 * [0.656 + 0.828] = 0.742$. The difference is due to the fact that there is a double specialization between the `transportWays` and `urbanStreets` concepts with respect to the `streets` and `urbanStreets` concepts and this affects the values of the global similarity. □

3.2 The deductive matching model

In the deductive matching model, both the thesaurus $\mathcal{TH}$ and the semantic relationships in the Domain Ontology are exploited to classify the kind of match between the request and the advertisements; following general guidelines in the current literature, we consider five kinds of match, that can be intuitively described as follows:
- *exact match*, when the request and the offer present the same functionalities (this is a strong condition);
- *plug-in match*, when the offer provides at least the required functionalities and possibly adds new ones;
- *subsume match*, when the functionalities provided by the offer are less than the required ones (it is like *plug-in match*, but with the roles of the request and the offer exchanged);
- *intersection match*, when the request and the offer present some common functionalities;
- *mismatch*, when no common functionalities exist between the request and the offer.

The deductive strategy relies on a non standard subsumption test (named $C \sqsubseteq_{\mathcal{TH}} D$) that is based both on the semantic relationships between concepts in the Domain Ontology and the terminological affinity according to the thesaurus.

Definition 1 (Affinity-based subsumption test). *Given the Domain Ontology $\mathcal{D}om\mathcal{ONT}$, the thesaurus $\mathcal{TH}$ and a pair of concepts C and D with names c_n and d_n, respectively, C is subsumed by D with respect to $\mathcal{TH}$, denoted by $C \sqsubseteq_{\mathcal{TH}} D$, if and only if one of the following conditions holds:*

- *$C, D \in \mathcal{D}om\mathcal{ONT}$ and $(C \sqsubseteq D)$ is satisfied in $\mathcal{D}om\mathcal{ONT}$;*
- *only $C \in \mathcal{D}om\mathcal{ONT}$ and $GC_N(C) \cap D_{\mathcal{TH}} \neq \emptyset$, where $GC_N(C) = \{Name(X)$ such that $X \in \mathcal{D}om\mathcal{ONT}$ and $C \sqsubseteq X\}$ is the set of names of concepts ancestors of $C \in \mathcal{D}om\mathcal{ONT}$ and $D_{\mathcal{TH}}$ is the set of terms that have name affinity (in symbols, $\sim$) with d_n, that is, $D_{\mathcal{TH}} = \{y \in \mathcal{TH} | (d_n \sim y)\}$;*
- *only $D \in \mathcal{D}om\mathcal{ONT}$ and $SC_N(D) \cap C_{\mathcal{TH}} \neq \emptyset$, where $SC_N(D) = \{Name(X)$ such that $X \in \mathcal{D}om\mathcal{ONT}$ and $X \sqsubseteq D\}$ is the set of names of the concepts descendants of $D \in \mathcal{D}om\mathcal{ONT}$ and $C_{\mathcal{TH}}$ is the set of terms that have name affinity with c_n, that is, $C_{\mathcal{TH}} = \{y \in \mathcal{TH} | (c_n \sim y)\}$.*

Note that we pose $C \equiv_{\mathcal{TH}} D$ if both $C \sqsubseteq_{\mathcal{TH}} D$ and $D \sqsubseteq_{\mathcal{TH}} C$ hold. □

In [2] we proposed a DL-based reasoning procedure to classify the five kinds of match. We have also shown that the deductive and similarity-based approaches can be combined into a hybrid matching model to obtain benefits of precision and flexibility from the two matching processes and to provide users with a measure of similarity. Firstly, the deductive strategy is applied to classify the kind of match between the request $\mathcal{R}$ and each offered service $\mathcal{S}$, then similarity evaluation is performed according to the following rules:
- if *exact* or *plug-in match* occurs, from the request viewpoint the offer provides completely the required functionalities, so $GSim(\mathcal{R}, \mathcal{S})$ is directly set to 1 (full similarity) without computing the similarity coefficients;
- if *mismatch* occurs, $GSim(\mathcal{R}, \mathcal{S})$ is directly set to zero;
- if *subsume* or *intersection match* occurs, the offer fulfills the request only partially and similarity coefficients are computed to quantify how much the offer satisfies the request; in this case, $GSim(\mathcal{R}, \mathcal{S}) \in (0, 1)$.
Only available services for which the $GSim(\mathcal{R}, \mathcal{S})$ is equal or greater than a given threshold ϕ are proposed among the searching results and they are ranked with respect to the $GSim$ values. The application of this hybrid approach ensures flexibility since when *exact* or *plug-in* matches are not found, partial matches are taken into consideration by evaluating similarity of request and offers. In this work, we describe an extended deductive procedure, that is able not only to identify the kind of match, but also to return what are the exceeding and the missing information among the request and the offer. In this way, it is possible for the users to choose the most suitable offers.

3.2.1 Deductive difference-based matching model

Another possible way to state the matching problem is to find the information contained in the request $\mathcal{R}$ and not in the offer $\mathcal{S}$ and viceversa, to verify if all the required capabilities are supplied by the advertisement, if only a portion of them is provided or no requirement is satisfied. To do this, we take inspiration from the *difference operator* proposed in [1] for comparing DL expressions of natural language queries and we adapt it to the problem of service matchmaking.

Definition 2 (Difference Operator). *The difference of two Description Logic expressions, that is, the information contained in the first expression and not in the second one, is defined in [11] as the syntactic minimum and expressed as:*

$$\mathcal{C} - \mathcal{D} := min\{\mathcal{X} \mid \mathcal{X} \sqcap \mathcal{D} \equiv \mathcal{C} \sqcap \mathcal{D}\} \tag{2}$$

where min is defined with respect to a subdescription ordering. □

The intuitive meaning of the difference operator is that it allows for removing from a given description C all the information contained in the D description.

The idea behind our approach is that, if we represent the capabilities of $\mathcal{R}$ and $\mathcal{S}$ by means of DL-based expressions and compare them, we can obtain three kinds of information:

– $\mathcal{MD}$ (*Missing Description*), it is the information required, but not provided by the advertisement;
– $\mathcal{ED}$ (*Exceeding Description*), it is the information provided, but not explicitly required;
– $\mathcal{CD}$ (*Common Description*), it is the information required and effectively provided, that is, common to the request and to the offer.

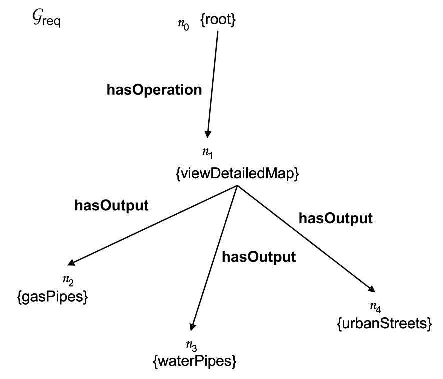
Actually, the expressions of $\mathcal{MD}$, $\mathcal{CD}$ and $\mathcal{ED}$ can be obtained from the Description Logic representation of $\mathcal{R}$ and $\mathcal{S}$ by means of the differences:

$$\mathcal{MD} := \mathcal{R} - \mathcal{S} \tag{3}$$
$$\mathcal{ED} := \mathcal{S} - \mathcal{R} \tag{4}$$
$$\mathcal{CD} := \mathcal{R} - \mathcal{MD} := \mathcal{S} - \mathcal{ED} \tag{5}$$

We can now define precisely the five kinds of match previously introduced. Firstly, we consider the service categories CAT_R of the request and CAT_S of the offer and we verify if $CAT_R \sqsubseteq_{\mathcal{TH}} CAT_S$. If this is not verified, a `Mismatch` is established, otherwise other matches can be progressively defined in terms of the $\mathcal{MD}$, $\mathcal{CD}$ and $\mathcal{ED}$ differences.

Match	**Definition in terms of $\mathcal{MD}$ and $\mathcal{ED}$**
`Exact`	if ($\mathcal{MD} \sqsubseteq_{\mathcal{TH}} \perp$) *and* ($\mathcal{ED} \sqsubseteq_{\mathcal{TH}} \perp$)
`Plug-in`	if *not*(`Exact`) *and* ($\mathcal{MD} \sqsubseteq_{\mathcal{TH}} \perp$)
`Subsume`	if *not*(`Plug-in`) *and* ($\mathcal{ED} \sqsubseteq_{\mathcal{TH}} \perp$)
`Mismatch`	if *not*(`Subsume`) *and* ($\mathcal{MD} \equiv_{\mathcal{TH}} \mathcal{R}$)
`Intersection`	Otherwise

An algorithm that performs the difference between two Description Logic expressions using a tree-based characterization of subsumption has been proposed in [1]. We will introduce the definition of *description tree* to express the functional description of services given in Section 2 and we will propose an algorithm to compute the difference between service descriptions exploiting the affinity-based subsumption test to reason on description trees.



$$\mathcal{G}_{req} = (N_{req}, E_{req}, n_0, \varrho_{req})$$

$$\begin{aligned}
N_{req} &= \{n_0, n_1, n_2, n_3, n_4\} \\
E_{req} &= \{n_0(\exists\texttt{hasOperation})n_1, \\
&\quad n_1(\exists\texttt{hasOutput})n_2, \\
&\quad n_1(\exists\texttt{hasOutput})n_3, \\
&\quad n_1(\exists\texttt{hasOutput})n_4\} \\
\varrho(n_0) &= \texttt{root} \\
\varrho(n_1) &= \texttt{viewDetailedMap} \\
\varrho(n_0) &= \texttt{gasPipes} \\
\varrho(n_0) &= \texttt{waterPipes} \\
\varrho(n_0) &= \texttt{urbanStreets}
\end{aligned}$$

Fig. 2. An example of description tree.

Definition 3 (Description tree). *A description tree for a Description Logic representation of a service $\mathcal{S}$ is a tree of the form $\mathcal{G}_\mathcal{S} = (N_\mathcal{S}, E_\mathcal{S}, n_0, \varrho_\mathcal{S})$, where:*

- *$N_\mathcal{S}$ is a finite set of nodes of $\mathcal{G}_\mathcal{S}$;*
- *$E_\mathcal{S} \subseteq N_\mathcal{S} \times \{\texttt{hasOperation}, \texttt{hasOutput}\} \times N_\mathcal{S}$ is a finite set of edges labeled with role names $r \in \{\texttt{hasOperation}, \texttt{hasOutput}\}$ ($\exists$-edges); an $\exists$-edge from n to m labeled r is written as $n(\exists r)m$; by construction, we can have the following kinds of $\exists$-edges:*
  - *$n_0(\exists\texttt{hasOperation})n_i \in E_\mathcal{S}$, with $i = 1 \dots |OP_\mathcal{S}|$, where $OP_\mathcal{S}$ is the set of operation names;*
  - *$n_i(\exists\texttt{hasOutput})p_{ih} \in E_\mathcal{S}$, with $i = 1 \dots |OP_\mathcal{S}|$ and $h = 1 \dots |OUT^i_\mathcal{S}|$, where $OUT^i_\mathcal{S}$ is the set of names of the output parameters associated to the operation $op^i_\mathcal{S} \in OP_\mathcal{S}$;*
- *$n_0 \in N_\mathcal{S}$ is the root of $\mathcal{G}_\mathcal{S}$;*
- *$\varrho_\mathcal{S} : N_\mathcal{S} \rightarrow [\bigcup_{i=1}^{|OP_\mathcal{S}|} OUT^i_\mathcal{S}] \cup OP_\mathcal{S} \cup \{\texttt{root}\}$ is a labeling function mapping the nodes in $N_\mathcal{S}$ to the concepts related to service description elements, with $\varrho(n_0) = \texttt{root}$.*

Example 4. A service description as given in Section 2 can be easily represented as a description tree. For example, the description tree for the request in the Example 1 is shown in Figure 2. □

The algorithm that computes the difference between the description tree representations of two services $\mathcal{R}$ and $\mathcal{S}$ and that uses the $\sqsubseteq_{\mathcal{TH}}$ subsumption test to establish semantic mappings among single constituents of the services is shown in Figure 3. In output the algorithm returns the type of match and possible differences (missing/exceeding elements). Here, the notation $\mathcal{G}.subtree(n)$ denotes the subtree with root node n in the description tree $\mathcal{G}$ and, given $m_0(\exists\texttt{hasOperation})m_i \in E_\mathcal{R}$ and $n_0(\exists\texttt{hasOperation})n_j \in E_\mathcal{S}$, $\mathcal{G}_\mathcal{R}.subtree(m_i) \sqsubseteq_{\mathcal{TH}} \mathcal{G}_\mathcal{S}.subtree(n_j)$ if and only if:
(i) $\varrho_\mathcal{R}(m_i) \sqsubseteq_{\mathcal{TH}} \varrho_\mathcal{S}(n_j)$, and

```
     function DIFFERENCE-MATCH($\mathcal{G_R}$,$\mathcal{G_S}$,$\mathcal{DomONT}$,$\mathcal{TH}$)
(1)  inputs: the description tree $\mathcal{G_R} = (N_\mathcal{R}, E_\mathcal{R}, m_0, \varrho_\mathcal{R})$ of a request $\mathcal{R}$
(2)          the description tree $\mathcal{G_S} = (N_\mathcal{S}, E_\mathcal{S}, n_0, \varrho_\mathcal{S})$ of an offer $\mathcal{S}$
(3)          a Domain Ontology $\mathcal{DomONT}$ and a thesaurus $\mathcal{TH}$
(4)  outputs: $\mathcal{MD} = diff_{aff}(\mathcal{G_R}, \mathcal{G_S})$
(5)          $\mathcal{ED} = diff_{aff}(\mathcal{G_S}, \mathcal{G_R})$
(6)          $Mtype \in$ {``exact'',``plug-in'',``subsume'',``intersection'',``mismatch''}

(7)  $\mathcal{MD} = \mathcal{G_R}$; $\mathcal{ED} = \mathcal{G_S}$;
(8)  if not $(CAT_R \sqsubseteq_{\mathcal{TH}} CAT_S)$
(9)      $Mtype$ = ``mismatch''; return $\mathcal{MD}$, $\mathcal{ED}$, $Mtype$;
(10) foreach $(m_0(\exists$hasOperation$)m_i \in E_\mathcal{R})$
(11)     if $\exists$ $(n_0(\exists$hasOperation$)n_j \in E_\mathcal{S})$ such that $[\mathcal{G_R}.subtree(m_i) \sqsubseteq_{\mathcal{TH}} \mathcal{G_S}.subtree(n_j)]$
(12)         delete $\mathcal{G_R}.subtree(m_i)$ from $\mathcal{MD}$;
(13)     else if $\exists$ $(n_0(\exists$hasOperation$)n_j \in E_\mathcal{S})$ such that $[\varrho_\mathcal{R}(m_i) \sqsubseteq_{\mathcal{TH}} \varrho_\mathcal{S}(n_j)]$
(14)         foreach $(m_i(\exists$hasOutput$)p_{ih} \in E_\mathcal{R})$
(15)             if $\exists$ $(n_j(\exists$hasOutput$)p_{jk} \in E_\mathcal{S})$ such that $[\varrho_\mathcal{R}(p_{ih}) \sqsubseteq_{\mathcal{TH}} \varrho_\mathcal{S}(p_{jk})]$
(16)                 delete $\mathcal{G_R}.subtree(p_{ih})$ from $\mathcal{MD}$;
     ...
     // repeat rows (10)-(16) exchanging the roles of $\mathcal{G_R}$ and $\mathcal{G_S}$ to find $\mathcal{ED}$
     ...
(17) if $(\mathcal{MD} \sqsubseteq \bot)$ and $(\mathcal{ED} \sqsubseteq \bot)$
(18)     $Mtype$ = ``exact'';
(19) else if $(\mathcal{MD} \sqsubseteq \bot)$
(20)     $Mtype$ = ``plug-in'';
(21) else if $(\mathcal{ED} \sqsubseteq \bot)$
(22)     $Mtype$ = ``subsume'';
(23) else if $(\mathcal{MD} \equiv \mathcal{G_R})$
(24)     $Mtype$ = ``mismatch'';
(25) else
(26)     $Mtype$ = ``intersection'';
(27) return $\mathcal{MD}$, $\mathcal{ED}$, $Mtype$;
```

Fig. 3. Difference algorithm between description tree representations of services.

(ii) for each $m_i(\exists\texttt{hasOutput})p_{ih} \in E_\mathcal{R}$ there exists $n_j(\exists\texttt{hasOutput})p_{jk} \in E_\mathcal{S}$ such that $\varrho_\mathcal{R}(p_{ih}) \sqsubseteq_{\mathcal{TH}} \varrho_\mathcal{S}(p_{jk})$.

Example 5. Applying the DIFFERENCE-MATCH algorithm to the service descriptions in the Example 1, we obtain:

```
request − DisplayGasInfr. := ∃hasOperation.(viewDetailedMap ⊓ ∃hasOutput.waterPipes)
DisplayGasInfr. − request := ⊥

request − DisplayTransportInfr. := ∃hasOperation.(viewDetailedMap ⊓ ∃hasOutput.waterPipes)
DisplayTransportInfr. − request := ⊥
```

Then, the request presents a `subsume` match both with `DisplayGasInfrastructure` and `DisplayTransportInfrastructure` and the difference operator returns the same results for the two comparisons. □

3.3 Optimization and ranking

The generalization hierarchy of Abstract services is exploited to make more efficient the service discovery procedure, according to the following intuition: if an Abstract service $\mathcal{S}_a$ matches with a given service request $\mathcal{R}$, then also Abstract services that provide at least the same capabilities of $\mathcal{S}_a$ (that is, are specializations of $\mathcal{S}_a$) match with $\mathcal{R}$. Once the desired Abstract service is found, its corresponding Concrete services are proposed among the searching results without any further application of the discovery algorithm.

In order to measure the degree of difference between a request and available offers we define a difference coefficient.

Definition 4 (Difference coefficient). *The difference coefficient between the services $\mathcal{R}$ and $\mathcal{S}$ is the size of their difference.*

$$diff(\mathcal{R}, \mathcal{S}) := |\mathcal{MD}(\mathcal{R}, \mathcal{S})| \tag{6}$$

where $|A|$ is the number of symbols in the expression A.

According to the difference coefficient, offers can be ranked and the more an offer fulfill the request, the best is the rank. For a given threshold of desired covery, 1-1 and 1-N mappings between a request and available offers can be determined.

4 Related work

Different matchmaking approaches have been developed aiming at improving keyword-based techniques. In general, service matchmaking strategies that are based on purely logic reasoning services [5, 9] present high precision and recall, but are often characterized by low flexibility. Moreover these approaches usually suffer from scalability problems. In [9] a service matchmaking strategy based on the OWL-S Service Profile and on a DL reasoner is proposed. The overall DAML+OIL expression representing a Service Profile is consistently mapped into a single DL expression and DL-based reasoning facilities are applied to check if the request description is equivalent, subsumed or consistent with the descriptions of service advertisements. In [5] the requested service profile and the provided one are expressed by means of DL expressions. The compared descriptions could be incomplete or not fully compatible, so when an element in the request that is not consistent with an element in the offer is found, it is removed (*contraction*) and each required element that is not present in the offer is added (*abduction*). Each time an element is removed or added, a penalty is assigned. The higher is the total penalty, the lower is the compatibility between the request and the advertisement. Moreover, the approaches based on Description Logics are characterized by a good trade-off between expressiveness and computational complexity.

On the contrary of logic-based approaches, similarity-based approaches are characterized by high flexibility, but also limited precision and recall, because, for example, if a partial match is established, there is no way to know if are required more functionalities than the provided ones or viceversa. In [7] a Web Service description is expressed through Web Service name and textual description, operation names and textual descriptions, input/output parameter descriptions, that is, the name, data type and arity, as contained in the corresponding WSDL file. The proposed algorithm evaluates the similarity of a pair of Web Service operations by exploiting a novel clustering procedure that groups parameter names into semantically meaningful concepts. A search engine, called Woogle, is implemented to support similarity search for Web Services. Moreover, similarity-based approaches exploit Information Retrieval techniques that consider service descriptions as vectors of terms and are not specifically tailored to service matchmaking. A comparison of deductive and similarity-based approaches shows that the former ones are able to distinguish between the request and the offer viewpoints, but do not provide a quantification of how much the offer matches with the request, while the latter approaches are symmetric, not distinguishing between the request and the offer, but provide a quantification of the degree of match. In any case, both of these kinds of strategies lack in performances and require high computational resources.

With respect to the previous approaches, we have proposed in [2] a novel hybrid matchmaking strategy that first uses a Description Logic-based classification to precisely establish the kind of match between the request and each offered service, then ranks the partially matching services on the basis of their similarity. Therefore the hybrid strategy tries to get the best from both the discussed approaches and to reduce the negative aspects of each ones by providing good precision and recall, but also quantifying the degree of matching. More recently, [8] also proposes a mixed service matchmaking approach, called OWLS-MX. Services are described using OWL-S Service Profiles and the degree of match of a service advertisement with a service request is based not only on the semantic relationships between DL constructs that express service description elements, but it is also implicitly contained in the relative frequencies of indexed terms of these descriptions, that are evaluated through traditional similarity metrics used in Information Retrieval. However, during the deductive matchmaking procedure, no terminological relationships are considered between names of elements used for service descriptions. In an open world assumption, such as that of the Web, where a lot of people could search for the same information using synonyms or could look for different information using homonyms, this limitation could strongly reduce the efficacy of the discovery process.

5 Conclusions

In this paper we have shown how different matching models can be combined in order to improve efficacy and flexibility of service discovery process. A similarity-based model and a deductive model are defined to find services that best fit a given request. These matching models can be combined into a hybrid model to improve searching results and can be used in conjunction with optimization and ranking strategies. Moreover, a newly defined deductive matching model based on difference operation has been proposed, that is able not only to identify the kind of match, but also to return what are the exceeding and the missing information among the request and the offer. In this way, it is possible for the users to choose the most suitable offers. The application of the different models produces different results depending on the level of flexibility expected from the requestor. Future work will investigate in more detail the design of a toolbox based on different matching models to support user in choosing the best flexible discovery strategy.

References

1. S. Benbernou, M.S. Hacid, N. Karam, and M. Schneider. Semantic Matching of Natural Language Web Queries. In *Proceedings of ICWE2004*, pages 416–429, Munich, Germany, July 2004.
2. D. Bianchini, V. De Antonellis, and M. Melchiori. Capability Matching and Similarity Reasoning in Service Discovery. In *CAiSE Int. Workshop on Enterprise Modeling and Ontologies for Interoperability, EMOI 2005*, Porto, Portugal, June 2005.
3. D. Bianchini, V. De Antonellis, M. Melchiori, and D. Salvi. Semantic-enriched Service Discovery. In *IEEE ICDE Int. Workshop on Challenges in Web Information Retrieval and Integration, WIRI 2006*, Atlanta, Georgia, USA, April 2006.
4. D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. Ontology-based Methodology for *e*-Service discovery. *Journal of Information Systems, Special Issue on Semantic Web and Web Services*, 31(4-5):361–380, June-July 2006.
5. A. Calì, D. Calvanese, S. Colucci, T. Di Noia, and F.M. Donini. A logic based approach for matching user profiles. In *Proc. of the 8th Int. Conf. on Knowledge-Based Intelligent Information & Engineering Systems (KES 2004)*, volume 3215 of *Lecture Notes in Artificial Intelligence*, pages 187–195, September 2004.
6. The OWL Service Coalition. *OWL-S 1.1 release*, November 2004. `http://www.daml.org/services/owl-s/1.1/`.
7. X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity Search for Web Services. In *Proc. of the 30th Int. Conference on Very Large Data Bases (VLDB2004)*, pages 372–383, Toronto, Canada, August-September 2004.
8. B. Fries, M. Khalid, M. Klusch, and K. Sycara. OWLS-MX: Hybrid OWL-S Service Matchmaking. In *Proc. of First International AAAI Symposium on Agents and Semantic Web*, Arlington, VA, USA, November 2005.
9. I. Horrocks and L. Li. A Software Framework for Matchmaking Based on Semantic Web Technology. *Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Michael Wellman and John Riedl, editors, Int. Journal of Electronic Commerce (IJEC 2004)*, August 2004.
10. U. Keller, H. Lausen, and D. Roman. *Web Service Modeling Ontology (WSMO).* WSMO Working Draft, March 2004. `http://www.wsmo.org/2004/d2/v02/20040306/`.
11. R. Kusters. Non-Standard Inferences in Description Logics. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 2100, 2001.