World Wide Web (2008) 11:227–251
DOI 10.1007/s11280-007-0040-y

Flexible Semantic-Based Service Matchmaking and Discovery

Devis Bianchini · Valeria De Antonellis · Michele Melchiori

Received: 3 April 2006 /Revised: 18 September 2007 /
Accepted: 26 November 2007 / Published online: 23 January 2007

Abstract Automated techniques and tools are required to effectively locate services that fulfill a given user request in a mobility context. To this purpose, the use of semantic descriptions of services has been widely motivated and recommended for automated service discovery under highly dynamic and context-dependent requirements. Our aim in this work is to propose an ontology-based hybrid approach where different kinds of matchmaking strategies are combined together to provide an adaptive, flexible and efficient service discovery environment. The approach, in particular, exploits the semantic knowledge about the business domain provided by a domain ontology underlying service descriptions, and the semantic organization of services in a service ontology, at different levels of abstraction.

Keywords hybrid service matchmaking approach · ontology-based service discovery · service ontology

1 Introduction

The continuous and rapid evolution of service-oriented and mobile technologies drives the development of new methods and techniques to improve serving of nomadic user requests. Architectural and functional requirements for designing multi-channel adaptive information systems, where services can be accessed through mobile terminals, in an ubiquitous and itinerant way, from different kinds of devices (e.g., laptops, palmtops, cellphones, smart cards) through different channels

D. Bianchini · V. De Antonellis · M. Melchiori (✉)
Dipartimento di Elettronica per l'Automazione,
Università degli Studi di Brescia, Via Branze, 38 - 25123 Brescia, Italy
e-mail: melchior@ing.unibs.it

D. Bianchini
e-mail: bianchin@ing.unibs.it

V. De Antonellis
e-mail: deantone@ing.unibs.it

(internet, wireless networks, etc.) have been widely investigated in [20, 22]. In particular, service discovery is considered one of the major crucial issues. Specifically, totally or partially automated techniques and tools are required to effectively locate services that fulfill a given user request.

Ability of understanding service requests and advertisements is strictly necessary to adapt and enhance service provisioning. To this purpose, the use of semantic descriptions of services has been widely motivated and recommended for automated service discovery under highly dynamic and context-dependent requirements in distributed environments. Semantic-enriched frameworks are considered a key issue to enforce timely discovery and dynamic composition of services. The ontology description languages OWL, OWL-S and, more recently, WSML have been proposed and several approaches based on these languages are being developed, as discussed in Section 2.

Our aim in this paper is to propose an ontology-based hybrid approach where different kinds of matchmaking strategies are combined together to provide an adaptive, flexible and efficient service discovery environment. We extend the keyword-based approach of UDDI Registry to obtain a semantic matchmaking approach based on the use of: a domain ontology, that provides the general knowledge about concepts of the business domain in which services are used; a service ontology, where services are organized by means of semantic relationships at multiple levels of abstraction. The approach has been originally developed within the MAIS Project [20] and then extended [2, 3] to combine different matchmaking models, metrics, ranking and optimization techniques as illustrated in this paper. We consider different matchmaking models: (1) a deductive model to determine the kind of match; (2) a similarity-based approach, exploiting retrieval metrics to measure the degree of match between services and (3) a hybrid model combining the previous ones to mix deductive precision with similarity-based flexibility. In the novel hybrid model, first a description logic-based classification is performed to precisely establish the kind of match between the request and each advertised service, then services with partial match are ranked on the basis of their similarity degree. The use of different matchmaking models aims at improving searching results and can be used in conjunction with optimization and ranking strategies. The application of the different models produces different results depending on the level of flexibility expected from the requester.

The paper is organized as follows: in Section 2 related works on service semantic description and discovery are discussed and compared with our approach; Section 3 describes the proposed ontological framework for service modeling using ontologies; Section 4 introduces the different matchmaking models to compare service descriptions, while Sections 5 presents the overall matchmaking algorithm; experimental results are shown in Section 6; finally, Section 7 provides concluding remarks.

2 Related work

Semantic service description. Several approaches have been proposed in literature for semantic service description and for service matchmaking purposes. In particular, for what concerns service semantic description, OWL-S [21] and WSMO [6] provide the most general frameworks. OWL-S [21] uses the OWL language to describe services

by means of three key elements: (1) a *service profile*, to describe the functional interface of the service, what it requires from the user and what it provides, that is, the input/output parameters and the service category; (2) a *service model*, to model the service structure, giving a detailed description of how the service operates by means of its constituent processes; (3) a *service grounding*, to map the abstract functional interface of the service into the concrete implementation and to provide details about how to interact with the service by means of message exchanges.

The Web Service Modeling Ontology (WSMO) [6] provides a formal language (Web Service Modeling Language WSML [18]) to semantically describe services through four core elements: (1) *ontologies*, to provide the terminology used by other WSMO elements in terms of machine-processable formal definitions; (2) *Web Services*, to provide a conceptual model describing all the aspects of Web Services, including their *non functional properties*, their *capabilities* and their *interfaces* (in WSMO a Web Service is defined as a computational entity that is able to achieve an user goal, while a service is the actual value provided by the invocation of the Web Service); (3) *goals*, to represent the user's requirements in terms of expected outputs and effects; (4) *mediators*, to deal with interoperability problems between different WSMO elements; a WSMO mediator acts as a third party component connecting heterogeneous elements.

Both OWL-S and WSMO consider several aspects for service semantic description. They require an high expressiveness of the description language with high computational complexity. Both of the approaches use domain ontologies to express semantics of service description elements, while the organization of services into service ontologies is not completely addressed to improve the efficiency of the discovery process. The only effort in this sense has been made by the OWL-S team, that aims at organizing service profiles into profile hierarchies and at classifying them by means of service categories.

In our approach, services are described, according to a description logic formalism, in terms of their functional interface. A domain ontology is used to express semantics of service description elements. Moreover, a three-layer service ontology is defined to organize services at different levels of abstraction.

Service discovery and matchmaking. Different matchmaking approaches have been developed aiming at improving keyword-based techniques provided by the UDDI registry. In general, service matchmaking strategies that are based on purely logic deductive facilities [7, 15] present high precision and recall and are characterized by a good trade-off between expressiveness and computational complexity, but are often characterized by low flexibility. By flexibility it is meant that the matchmaker is able to recognize not only exact matches but also the degree of similarity between a service request and a service advertisement that do not match exactly. Moreover, these approaches usually suffer from scalability problems. In [15] 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 deductive facilities are applied to check if the description of request is equivalent, subsumed or consistent with the descriptions of service advertisements. In [7] 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 service request that is not consistent with an element in the service advertisement is found, it is removed (*contraction*) and each required element that is not present in the service advertisement 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.

On the contrary, with respect to 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 found, there is no way to know if it is due to the fact that the required functionalities are more than the provided ones or viceversa. In [11] a Web Service description is expressed through Web Service name with its textual description, names of operations and their textual descriptions, input/output parameter descriptions, that is, their name, data type and cardinality, 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 service request and the service advertisement, but do not provide a quantification of how much the advertisement matches with the request, while the latter approaches are symmetric, not distinguishing between the request and the advertisement, but provide a quantification of the degree of match. In our approach, a hybrid matchmaking model is proposed to combine advantages from deductive and similarity-based models.

Also in [13] a mixed service matchmaking approach, called OWLS-MX, has been recently proposed. Services are described using OWL-S Service Profiles and the degree of match of a service advertisement S with a service request R is based not only on the semantic relationships between DL constructs that express service description elements, but also on frequencies of indexed terms of these descriptions, that are evaluated through traditional Information Retrieval metrics. We note that the similarity-based part of this approach exploits IR techniques that are not tailored to service description comparisons and do not use lexical knowledge to enhance matchmaking effectiveness. In our approach, deductive and similarity-based techniques are combined. Moreover, besides the domain ontology, a thesaurus is introduced to relate names of concepts with other terms by means of terminological relationships (e.g., synonymy or hypernymy). In this way, it is possible to extend matchmaking capabilities when looking for correspondences between elements in service requests and advertisements and concepts in the domain ontology.

3 The ontological framework

Ontologies are used to extend the functional description of services with semantic knowledge. The provider's and requester's perspectives are distinguished:

(1) Web Services are advertised by using as service element names (service category, input/output parameters and corresponding operations) the atomic

concepts defined in an OWL-DL ontology; this ontology expresses the domain specific knowledge of the provider and we will refer it as the *domain ontology* $\mathcal{DOM}\mathcal{ONT}$; we consider services in a specific domain of interest and we are constrained to a single domain ontology; services are semantically organized in a *service ontology* $\mathcal{SERV}\mathcal{ONT}$;

(2) The requester expresses his requirements in terms of desired categories, operations, input and output parameters, but it is not expected that a service request conforms to the domain ontology; a *thesaurus* $\mathcal{TH}$ is used to relate atomic concepts in the domain ontology with additional terms by means of terminological relationships (e.g., *synonymy*, *hypernymy*, etc.); the service ontology is exploited to improve the discovery process.

To keep backward compatibility with existing service description technologies, the representation of functional interface of requested and provided services is based on the WSDL specification. According to the WSDL document, Web Service functional interface is identified by a set of operations, with their input and output parameters. Services are advertised in an UDDI registry, where they are associated by means of `tModel` mechanism to service categories from standard taxonomies like UNSPSC or NAICS.

3.1 Semantic infrastructure for service publication

Description logics have been adopted to formally represent services in the service ontology and are exploited during service discovery. Basic elements in description logics are concepts and roles, complex descriptions can be built from them inductively with concept constructors. Description languages are distinguished by the constructors they provide [1]. Table 1 summarizes the syntax and semantics of $\mathcal{SHOIN}(\mathcal{D})$ family [10]. A concept C is defined as follows:

- An *atomic concept*, identified by its name A, is a concept;
- An enumeration of individuals $\{i_1, i_2, \ldots i_n\}$ is a concept;
- Given two concepts C_1 and C_2, $C_1 \sqcap C_2$ (*intersection*), $C_1 \sqcup C_2$ (*union*), $\neg C$ (*negation*) and (C) are concepts;
- $\exists R.C$ (*existential role restriction*) is a concept, where R is a role name.

The semantics of description logics is defined by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \bullet^{\mathcal{I}})$, consisting of a *domain of the interpretation* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\bullet^{\mathcal{I}}$ which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

When a new Web Service is provided, operations and input/output parameters in the WSDL document are mapped to atomic concepts defined in the domain ontology. The new Service is published in the UDDI registry by means of traditional UDDI Publish API, in order to keep compatibility with existing UDDI standard, together with its WSDL document. Moreover, a new concept representing the provided service is loaded in a OWL-DL service ontology $\mathcal{SERV}\mathcal{ONT}$. The concept is represented using description logic formalism, as a conjunction of:

- One or more concepts in the form $\exists \texttt{hasCategory}.CAT$, where CAT is a concept representing an associated service category;

Table 1 Abstract syntax and semantics of $\mathcal{SHOIN}(\mathcal{D})$.

Constructs	Syntax	Semantics
Atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Role name	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Individuals	i	$i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
Enumeration	$\{i_1, i_2, \ldots i_n\}$	$(\{i_1, i_2, \ldots i_n\})^{\mathcal{I}} = \{(i_1)^{\mathcal{I}}, (i_2)^{\mathcal{I}}, \ldots (i_n)^{\mathcal{I}}\}$
Intersection	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}}$
Union	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = (C_1)^{\mathcal{I}} \cup (C_2)^{\mathcal{I}}$
Negation	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C^{\mathcal{I}}$
Existential role restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}} \mid \exists d \in \Delta^{\mathcal{I}} s.t. (c, d) \in R^{\mathcal{I}} \wedge d \in C^{\mathcal{I}}\}$
Universal role restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}} \mid \forall d \in \Delta^{\mathcal{I}} s.t. (c, d) \in R^{\mathcal{I}} \Rightarrow d \in C^{\mathcal{I}}\}$
Universal concept	$\top$	$(\top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$
Empty concept	$\bot$	$(\bot)^{\mathcal{I}} = \emptyset$
Equivalence	$C_1 \equiv C_2$	$(C_1)^{\mathcal{I}} = (C_2)^{\mathcal{I}}$
Subsumption	$C_1 \sqsubseteq C_2$	$(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$
Disjointness	$C_1 \sqsubseteq \neg C_2$	$\mathcal{I}$ if $(C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} = \emptyset$
Inverse role	R^-	$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^-$
Role transitivity	$(Tr(R))$	$(Tr(R))^{\mathcal{I}} = (R^{\mathcal{I}})^+$
Role hierarchy	$R_1 \sqsubseteq R_2$	$(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$
At-most cardinality constraint	$\leq nR$	$(\leq nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y\ s.t.\ (x, y) \in R^{\mathcal{I}}\}) \leq n\}$
At-least cardinality constraint	$\geq nR$	$(\geq nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y\ s.t.\ (x, y) \in R^{\mathcal{I}}\}) \geq n\}$
Exactly cardinality constraint	$= nR$	$(= nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y\ s.t.\ (x, y) \in R^{\mathcal{I}}\}) = n\}$

- One or more concepts in the form $\exists$`hasOperation`$.OP$, where OP is described as a conjunction of:
  - A concept representing the operation name;
  - One or more concepts $\exists$`hasInput`$.IN$, where IN is a concept representing an input parameter of the operation;
  - One or more concepts $\exists$`hasOutput`$.OUT$, where OUT is a concept representing an output parameter of the operation.

These expressions can be represented in OWL-DL, corresponding to the $\mathcal{SHOIN}(\mathcal{D})$ family of description logics.

Example 1 In the following, the concepts representing two service advertisements, taken from the geographic service domain, are shown; Figure 1 shows the reference domain ontology:

```
DisplayStreets ≡ ∃hasCategory.MapVisualization ⊓
                 ∃hasOperation.(MapVisualization ⊓
                   ∃hasInput.Region ⊓
                   ∃hasInput.GeographicCoordinates ⊓
                   ∃hasOutput.Street)

DisplayGasInfrastructure ≡ ∃hasCategory.MapVisualization ⊓
                           ∃hasOperation.(Visualization ⊓
                             ∃hasInput.GeographicCoordinates ⊓
                             ∃hasOutput.RoadInfrastructure ⊓
                             ∃hasOutput.GasPipe)
```

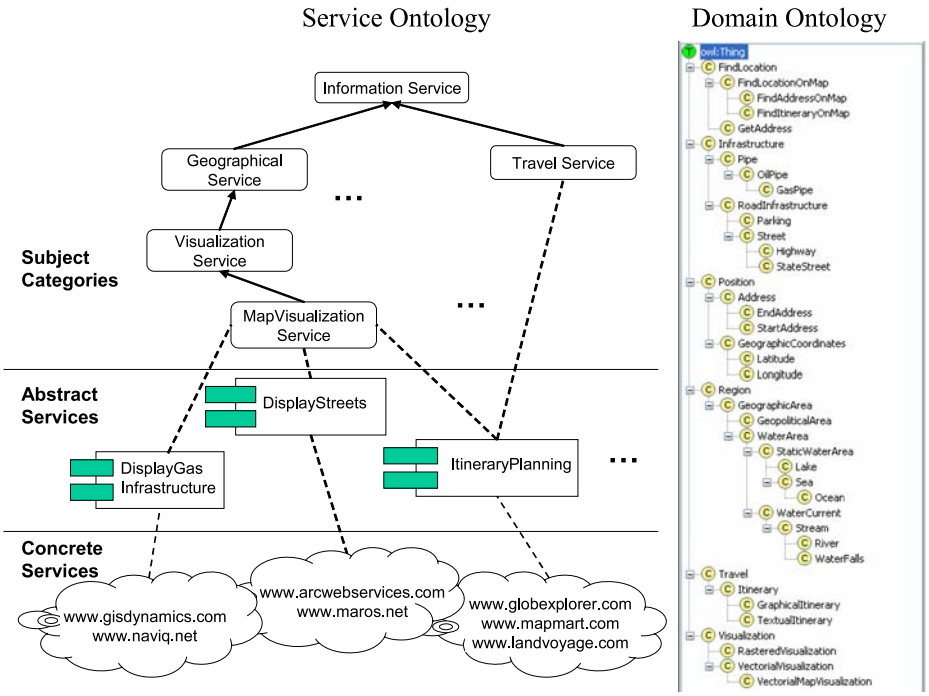

Figure 1 A portion of a service ontology for geographic information service domain.

Both services belong to the `MapVisualization` category. The `DisplayGasInfrastructure` service performs a generic visualization of the `RoadInfrastructure` and `GasPipe` given the `GeographicCoordinates`. The `DisplayStreets` service performs on a map a visualization of `Street` in a `Region` given the `GeographicCoordinates`.

To improve the effectiveness and the efficiency of service discovery, additional semantics is associated to the service ontology $\mathcal{ServONT}$, that organizes services at different levels of abstraction by means of semantic relationships that can be fruitfully exploited to support service discovery. Starting from the bottom layer, we distinguish between *Concrete Services*, *Abstract Services* and *Service Categories*, organized into *Concrete*, *Abstract* and *Category* layer, respectively.

Concrete Services are services published into the UDDI registry by providers and referenced in the *Concrete* layer of the service ontology. *Concrete Services* are grouped into *clusters* according to properly defined coefficients that evaluate their functional similarity. Similarity coefficients are the same used in the similarity-based matchmaking model presented in Section 4.1.

Abstract Services are not directly invocable services, but are introduced to summarize the functionalities of clusters of similar *Concrete Services*. *Abstract Service* functionalities are also described by means of operations and I/O parameters obtained by integrating the functional descriptions of *Concrete Services* belonging to the same cluster. Mapping rules are maintained among the operations and I/O

parameters of each *Abstract Service* and of the corresponding *Concrete Services*. Due to the integration process, names of operations, input/output parameters and service categories of *Abstract Services* correspond to atomic concepts defined in the domain ontology.

Abstract Services are semantically organized in a hierarchy according to the following criteria: an *Abstract Service* $\mathcal{S}_a^i$ is a *generalization* of another *Abstract Service* $\mathcal{S}_a^j$, with in general $i \neq j$, if, informally stated, $\mathcal{S}_a^j$ provides at least the functionalities of $\mathcal{S}_a^i$. This relationship between *Abstract Services* $\mathcal{S}_a^i$ and $\mathcal{S}_a^j$ conforms to the `plug-in` match between service descriptions presented in Section 4.1.

Service Categories organize *Abstract Services* into standard available taxonomies to provide a topic-driven access to the underlying *Abstract* and *Concrete Services*; these categories are the same that are used in the UDDI registry to classify published *Concrete Services*; in $\mathcal{SERVONT}$ an *Abstract Service* $\mathcal{S}_a^i$ is associated to the set of *Service Categories* related to the *Concrete Services* belonging to the cluster that corresponds to $\mathcal{S}_a^i$. Construction and evaluation of the service ontology are outside the scope of this paper and are described in [4].

Example 2 Figure 1 shows a portion of a service ontology and a domain ontology for geographic information services; for simplicity, only subsumption relationships between concepts of the domain ontology are represented; the service ontology contains the advertised *Abstract Services* whose descriptions have been introduced in Example 1 and a third service is added:

```
ItineraryPlanning  ≡  ∃hasCategory.MapVisualization ⊓
                      ∃hasCategory.TravelService ⊓
                      ∃hasOperation.(Visualization ⊓
                        ∃hasInput.GeographicCoordinates ⊓
                        ∃hasOutput.Street) ⊓
                      ∃hasOperation.(FindItineraryOnMap ⊓
                        ∃hasInput.StartAddress ⊓
                        ∃hasInput.EndAddress ⊓
                        ∃hasOutput.TextualItinerary ⊓
                        ∃hasOutput.GraphicalItinerary)
```

This service performs a visualization of the `Street` given the `GeographicCoordinates` and finds on a map an itinerary given the `StartAddress` and the `EndAddress`; the service provides both a textual and a graphical representation of the itinerary.

3.2 Semantic infrastructure for service discovery

The service request is represented by means of the same description logic expression we presented in the previous section. To support the matchmaking between the request and the advertisement, the knowledge contained in the domain ontology $\mathcal{DOMONT}$ is extended by means of a thesaurus $\mathcal{TH}$, where terms used as names of ontological concepts are related to other terms by means of terminological relationships (e.g., *synonymy*, *hypernymy*, *hyponymy*, etc.).

Following a procedure similar to that proposed by [9], the thesaurus $\mathcal{TH}$ is automatically derived by considering the set T of terms denoting atomic concepts in $\mathcal{DOMONT}$ and the lexical system WordNet [12]. To be compliant with real ontologies, where names can be composed by one or more terms, we select terms and terminological relationships to be stored in $\mathcal{TH}$ as follows.

Simple terms. A term $t \in T$ is a *simple term*, denoted by st, if an entry for st exists in WordNet; an entry is defined in the thesaurus $\mathcal{TH}$ for each simple term $st_i \in T$.

Composite terms. A term $t \in T$ is a *composite term*, denoted by $ct = \langle st_1, st_2, \ldots st_n \rangle$, if ct is composed by more than one simple term st_i and an entry for ct does not exist in WordNet; following the intuition in [17], as usual in English, in a composite term ct the rightmost simple term st_n denotes the central concept represented bt ct, while the remaining simple terms (st_i, with $i = 1 \ldots n-1$) are used to better specify the meaning of st_n (e.g., in term `TextualItinerary`, simple term `Itinerary` is further specified by `Textual` adjective). Up to now, we assume that ontology designers should not use shortcuts for ontology concepts; moreover, we exploit WordNet to recognize and separate simple component words in composite terms; given a composite term $ct = \langle st_1, st_2, \ldots st_n \rangle$, an entry is defined in $\mathcal{TH}$ for ct and for each constituent simple term st_i.

Terminological relationships. Given the thesaurus $\mathcal{TH}$ containing a set of simple and composite terms, terminological relationships in $\mathcal{TH}$ are established by considering terminological relationships among synsets in WordNet. Given two simple terms st_i, st_j in $\mathcal{TH}$, a terminological relationship tr between them is defined in $\mathcal{TH}$ as follows:

- tr = SYN if st_i and st_j belong to the same synset in WordNet;
- tr = BT/NT if a hypernymy/hyponymy relationship exists in WordNet between the synsets of st_i and of st_j, respectively;
- tr = RT if a meronymy relationship exists in WordNet between the synsets of st_i and st_j or st_i and st_j are coordinate terms in WordNet.

Given a composite term $ct = \langle st_1, st_2, \ldots st_n \rangle$, the following terminological relationships are defined in $\mathcal{TH}$:

- BT between st_n and ct, to denote that ct is a specialization of st_n;
- RT between each st_i (with $i = 1 \ldots n-1$) and ct, to denote that st_i and ct are related terms.

The rules applied to build the terminological relationships in $\mathcal{TH}$ are summarized in Table 2.

Table 2 Terminological relationships in the thesaurus $\mathcal{TH}$.

Rule	Thesaurus entries	Relationships in WordNet	Relationships added to the thesaurus
R1	st_i, st_j simple terms	a) st_i, st_j belongs to the same synset	st_i SYN st_j
		b) a hypernymy/hyponymy relationship exists between synsets of st_i, st_j	st_i BT/NT st_j
		c) a meronymy relationship exists between synsets of st_i and st_j or st_i and st_j are coordinate terms	st_i RT st_j
R2	$ct = \langle st_1, st_2, \ldots st_n \rangle$ composite term		st_n BT ct st_i RT $ct, i = 1 \ldots n-1$

Example 3 Given the composite term `VectorialMapVisualization` in $\mathcal{TH}$, the following relationships are also added by applying the rule R2 for composite terms:

- `Visualization` BT `VectorialMapVisualization`
- `Map` RT `VectorialMapVisualization`
- `Vectorial` RT `VectorialMapVisualization`

Example 4 The user is supported by a graphical interface during submission of service request, without being constrained to the use of the atomic concepts of the domain ontology. A template is suggested to the user to formulate the request, for example

"find `VisualizationService` for/to `VectorialMapVisualization` to obtain `Road` given `Position,Area`"

can be easily formalized using the DL expression proposed in Section 3.1

```
Request_1  ≡  ∃hasCategory.VisualizationService ⊓
              ∃hasOperation.(VectorialMapVisualization ⊓
                ∃hasInput.Position ⊓
                ∃hasInput.Area ⊓
                ∃hasOutput.Road)
```

while the following request

"find `TravelService` for/to `Find` to obtain `Path` given `InitialAddress,Destination`"

can be formalized as

```
Request_2  ≡  ∃hasCategory.TravelService ⊓
              ∃hasOperation.(Find ⊓
                ∃hasInput.InitialAddress ⊓
                ∃hasInput.Destination ⊓
                ∃hasOutput.Path)
```

Note that the requester could use some terms that are not included in the domain ontology. These terms are related with terminological relationships to the terms in the domain ontology using WordNet, for example:

```
 Street   SYN   Road
   Area   SYN   Region
Address    BT   InitialAddress
Initial    RT   InitialAddress
Address    BT   Destination
   Path   SYN   Itinerary
```

3.3 Joint use of thesaurus and domain ontology

The thesaurus introduced in Section 2 is exploited to compute the *Name Affinity* coefficient between names of input/output parameters and operations. A weight $\sigma_{tr} \in [0, 1]$ is associated to each kind of terminological relationship $tr \in \{\text{SYN}, \text{BT/NT}, \text{RT}\}$

in the thesaurus, in order to evaluate its implication for name affinity; in particular, we have $\sigma_{SYN} > \sigma_{BT/NT} > \sigma_{RT}$ (in our experimentation, $\sigma_{SYN} = 1, \sigma_{BT/NT} = 0.8$ and $\sigma_{RT} = 0.5$). The thesaurus can be viewed as a matrix, where each cell is represented in the form $\langle (t, t'), tr, \sigma_{tr} \rangle$, where t, t' are two entries of the thesaurus (respectively, the source and the target terms of the relationship tr). Two terms can be related by one or more chains of terminological relationships: we call *path* of length l between two terms t, t', denoted with $t \rightarrow^l t'$, a finite ordered sequence of l terminological relationships $\langle tr_1, tr_2, \ldots tr_l \rangle$, where the source term of tr_1 is t and the target term of tr_l is t'. The *strength* of $t \rightarrow^l t'$ is the product of the weights of all the relationships belonging to the path, that is, $\tau(t \rightarrow^l t') = \prod_{k=1}^{l} (\sigma_{tr_k}) \in [0, 1]$. Since between two terms in the thesaurus there can exist more than one path, the one with the highest strength is chosen. The *Name Affinity* coefficient between t and t', denoted by $NA(t, t')$, is computed as follows:

$$NA(t, t') = \begin{cases} 1 & if \;\; t = t' \\ \max_l(\tau(t \rightarrow^l t')) & if \;\; t \neq t' \wedge \\ & t \rightarrow^l t', l \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We say that t and t' have *name affinity* ($t \sim t'$) if and only if $NA(t, t') \geq \alpha$, where $\alpha > 0$ is a threshold given by experimental results to select only terms with high values of the *Name Affinity* coefficient. The choice of the actual value of α is done during a training phase where α is set initially to a given value (i.e., 0.5), that is increased or decreased until a satisfactory trade-off between recall and precision is obtained. That is, increasing α leads to be more selective by identifying a name affinity between two terms only if they are very similar according to the thesaurus. Viceversa, by decreasing α, name affinities are established also between pairs of terms that are related by a weaker path of terminological relationships.

The combined use of the domain ontology and the thesaurus allows to identify a subsumption relationship between terms not necessarily defined in $\mathcal{DOMONT}$, as shown in the following definitions. Firstly, we define the set of concepts in $\mathcal{DOMONT}$ that have name affinity with a given term.

Definition 1 Given a term X not necessarily denoting an atomic concept of $\mathcal{DOMONT}$, we define the set $X_{\mathcal{TH}}$ as:

$$X_{\mathcal{TH}} = \begin{cases} \{X\} & \text{if } X \in \mathcal{DOMONT} \\ \{\text{atomic concept } Y \in \mathcal{DOMONT} \mid Y \sim X\} & \text{otherwise} \end{cases} \tag{2}$$

$X_{\mathcal{TH}}$ represents the set of atomic concepts in the domain ontology that have name affinity with X.

Definition 2 (Affinity-based subsumption test) Given the domain ontology $\mathcal{DOMONT}$, the thesaurus $\mathcal{TH}$ and a pair of terms C and D, C is subsumed by D with respect to $\mathcal{TH}$, denoted by $C \sqsubseteq_{\mathcal{TH}} D$, if and only if there exists $X \in C_{\mathcal{TH}}$ and $Y \in D_{\mathcal{TH}}$ such that $(X \sqsubseteq Y)$ is satisfied in $\mathcal{DOMONT}$. Note that we pose $C \equiv_{\mathcal{TH}} D$ if both $C \sqsubseteq_{\mathcal{TH}} D$ and $D \sqsubseteq_{\mathcal{TH}} C$ hold.

Example 5 Given the domain ontology shown in Figure 1 and the portion of thesaurus built in the examples 3 and 4, $\texttt{TextualItinerary} \sqsubseteq_{\mathcal{TH}} \texttt{Path}$ is satisfied since:

- $\texttt{TextualItinerary}_{\mathcal{TH}}$={TextualItinerary} and $\texttt{Path}_{\mathcal{TH}}$={Itinerary}
- $\texttt{TextualItinerary} \sqsubseteq \texttt{Itinerary}$ is satisfied in $\mathcal{DOMONT}$

4 Service matchmaking

We propose a composite service matchmaking approach characterized by the following components.

Matchmaking model. Different models are considered: (1) a similarity-based model, where retrieval metrics are applied to measure the degree of match between services [4], (2) a deductive model, based on deduction algorithms for matching service functional descriptions [3] and (3) a hybrid model, that combines the similarity-based and the deductive models [5]. In particular, the hybrid model combines the precision of deductive model with flexibility of similarity-based one providing a trade-off between precision and recall.

Metrics. Different metrics are introduced to compute similarity between services.

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

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

4.1 Deductive matchmaking model

The affinity-based subsumption test between service description elements is exploited to verify the kind of match between an advertisement $\mathcal{S}$ and a request $\mathcal{R}$. Following general guidelines in the current literature (for instance [15]), we consider five kinds of match. To establish the kind of match, service description components are considered separately. A pre-filtering phase considers the service category $CAT_{\mathcal{R}}$ of the request and the service categories $CAT^i_{\mathcal{S}}$ of the advertisement: if there exists $CAT^i_{\mathcal{S}}$ such that $CAT^i_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} CAT_{\mathcal{R}}$, the kind of match between $\mathcal{R}$ and $\mathcal{S}$ is investigated, otherwise the match fails (`mismatch`).

Let be $\mathcal{R}$ a service request with a set of operations $\{op^i_{\mathcal{R}}\}$, where $op^i_{\mathcal{R}}$ is the i-th operation of $\mathcal{R}$ with inputs $\{in^{ih}_{\mathcal{R}}\}$ and outputs $\{out^{ik}_{\mathcal{R}}\}$. Furthermore, let be $\mathcal{S}$ an advertisement with a set of operations $\{op^j_{\mathcal{S}}\}$, where $op^j_{\mathcal{S}}$ is the j-th operation of $\mathcal{S}$ with inputs $\{in^{jp}_{\mathcal{S}}\}$ and outputs $\{out^{jt}_{\mathcal{S}}\}$. `Exact` match denotes that $\mathcal{R}$ and $\mathcal{S}$ present the *same* functionalities, that is, what is required by $\mathcal{R}$ is exactly provided by $\mathcal{S}$. Formally, we say that

$$
\begin{aligned}
&match(\mathcal{R}, \mathcal{S}) = \text{`}\texttt{exact}\text{'} \text{ if and only if}\\
&\quad \forall op^i_{\mathcal{R}}\ \exists op^j_{\mathcal{S}} \text{ such that}\\
&\qquad (name\_op^j_{\mathcal{S}} \equiv_{\mathcal{TH}} name\_op^i_{\mathcal{R}}) \wedge\\
&\qquad (\forall in^{jp}_{\mathcal{S}}\ \exists in^{ih}_{\mathcal{R}}: in^{ih}_{\mathcal{R}} \equiv_{\mathcal{TH}} in^{jp}_{\mathcal{S}}) \wedge\\
&\qquad (\forall out^{ik}_{\mathcal{R}}\ \exists out^{jt}_{\mathcal{S}}: out^{jt}_{\mathcal{S}} \equiv_{\mathcal{TH}} out^{ik}_{\mathcal{R}})
\end{aligned}
$$

Note that this is a very strong condition. Each operation of $\mathcal{R}$ is compared with each operation of $\mathcal{S}$. In the matchmaking process (for each kind of match) we require that for each comparison between two operations (respectively, between corresponding parameters of two operations) when a kind of match is established for a pair of operations (respectively, corresponding parameters) such operations (respectively, parameters) do not participate in further comparisons.

Service advertisement $\mathcal{S}$ presents a `plug-in` match with the request $\mathcal{R}$ when $\mathcal{S}$ provides *at least* the required functionalities, but also further capabilities or information that are not required. Formally, we say that

$$
\begin{aligned}
&match(\mathcal{R}, \mathcal{S}) = \text{`}\texttt{plug-in}\text{'} \text{ if and only if}\\
&\quad \forall op^i_{\mathcal{R}}\ \exists op^j_{\mathcal{S}} \text{ such that}\\
&\qquad (name\_op^i_{\mathcal{R}} \sqsubseteq_{\mathcal{TH}} name\_op^j_{\mathcal{S}}) \wedge\\
&\qquad (\forall in^{jp}_{\mathcal{S}}\ \exists in^{ih}_{\mathcal{R}}: in^{jp}_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} in^{ih}_{\mathcal{R}}) \wedge\\
&\qquad (\forall out^{ik}_{\mathcal{R}}\ \exists out^{jt}_{\mathcal{S}}: out^{ik}_{\mathcal{R}} \sqsubseteq_{\mathcal{TH}} out^{jt}_{\mathcal{S}})
\end{aligned}
$$

Example 6 `DisplayStreets` (DS) presents a `plug-in` match with `Request_1` (R1), since

$\forall op^i_{R1}\ \exists op^j_{DS}$ such that $(name\_op^i_{R1} \sqsubseteq_{\mathcal{TH}} name\_op^j_{DS}) \wedge (\forall out^{ik}_{R1}\ \exists out^{jt}_{DS}: out^{ik}_{R1} \sqsubseteq_{\mathcal{TH}} out^{jt}_{DS})$

`VectorialMapVisualization`	$\sqsubseteq_{\mathcal{TH}}$	`MapVisualization`
`Road`	$\sqsubseteq_{\mathcal{TH}}$	`Street`

$\forall op^i_{R1}\ \exists op^j_{DS}$ such that $(\forall in^{jp}_{DS}\ \exists in^{ih}_{R1}: in^{jp}_{DS} \sqsubseteq_{\mathcal{TH}} in^{ih}_{R1})$

`Region`	$\sqsubseteq_{\mathcal{TH}}$	`Area`
`GeographicCoordinates`	$\sqsubseteq_{\mathcal{TH}}$	`Position`

Abstract Services in $\mathcal{ServONT}$ are semantically organized according to the `plug-in` match. In the service ontology shown in Figure 1, `DisplayGasInfrastructure` (DGI) presents a `plug-in` match with `DisplayStreets` (DS), since

$\forall op^i_{DS}\ \exists op^j_{DGi}$ such that $(name\_op^j_{DS} \sqsubseteq_{\mathcal{TH}} name\_op^i_{DGI}) \wedge (\forall out^{ik}_{DS}\ \exists out^{jt}_{DGI}: out^{jt}_{DS} \sqsubseteq_{\mathcal{TH}} out^{ik}_{DGI})$

`MapVisualization`	$\sqsubseteq_{\mathcal{TH}}$	`Visualization`
`Street`	$\sqsubseteq_{\mathcal{TH}}$	`RoadInfrastructure`

$\forall op^i_{DS}\ \exists op^j_{DGI}$ such that $(\forall in^{jp}_{DGI}\ \exists in^{ih}_{DS}: in^{ih}_{DGI} \sqsubseteq_{\mathcal{TH}} in^{jp}_{DS})$

`GeographicCoordinates`	$\sqsubseteq_{\mathcal{TH}}$	`GeographicCoordinates`

Following the same procedure, `ItineraryPlanning` has a `plug-in` match with `DisplayStreets`. The *Abstract Services* are then organized in $\mathcal{ServONT}$ as shown in Figure 2.

From the requester viewpoint, the `exact` and `plug-in` match constitute the best situations, because in both cases all the required operations are provided. The

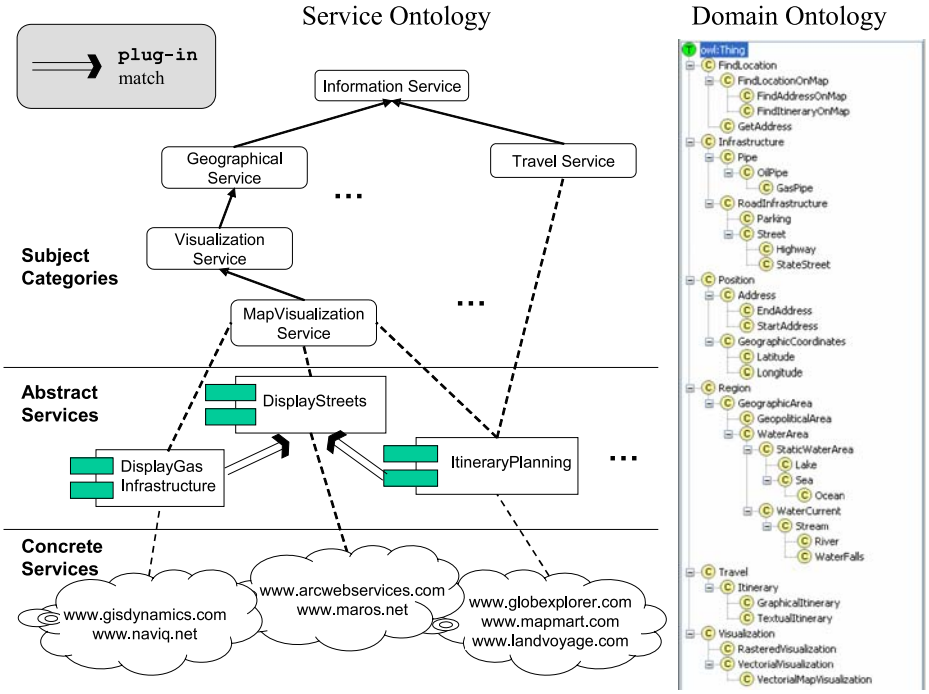

Figure 2 *Abstract Services* in the service ontology organized through `plug-in` matches.

advertisement $\mathcal{S}$ presents a `subsume` match with $\mathcal{R}$ when $\mathcal{S}$ presents *some but not all* the functionalities of $\mathcal{R}$. Formally, we say that

$$
\begin{aligned}
&match(\mathcal{R},\mathcal{S}) = \texttt{'subsume'} \text{ if and only if}\\
&\quad \forall op^{j}_{\mathcal{S}}\ \exists op^{i}_{\mathcal{R}} \text{ such that}\\
&\qquad (name\_op^{j}_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} name\_op^{i}_{\mathcal{R}}) \wedge\\
&\qquad (\forall in^{ih}_{\mathcal{R}}\ \exists in^{jp}_{\mathcal{S}}\colon in^{ih}_{\mathcal{R}} \sqsubseteq_{\mathcal{TH}} in^{jp}_{\mathcal{S}}) \wedge\\
&\qquad (\forall out^{jt}_{\mathcal{S}}\ \exists out^{ik}_{\mathcal{R}}\colon out^{jt}_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} out^{ik}_{\mathcal{R}})
\end{aligned}
$$

Note that `subsume` match constitutes the inverse situation with respect to the `plug-in` match. The intermediate situation, in which $\mathcal{R}$ and $\mathcal{S}$ have *some common* functionaties, that is, there are some capabilities or information that are required but not provided and some capabilities or information that are provided but not required, is the most general case and is referred as `intersection` match. Formally, we say that

$$
\begin{aligned}
&match(\mathcal{R},\mathcal{S}) = \text{`}\texttt{intersection}\text{'} \text{ if and only if}\\
&\quad \exists op^{i}_{\mathcal{R}}, op^{j}_{\mathcal{S}} \text{ such that}\\
&\qquad (name\_op^{i}_{\mathcal{R}} \sqsubseteq_{\mathcal{TH}} name\_op^{j}_{\mathcal{S}} \vee name\_op^{j}_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} name\_op^{i}_{\mathcal{R}}) \wedge\\
&\qquad [(\exists out^{ik}_{\mathcal{R}}, out^{jt}_{\mathcal{S}}\colon out^{ik}_{\mathcal{R}} \sqsubseteq_{\mathcal{TH}} out^{jt}_{\mathcal{S}} \vee out^{jt}_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} out^{ik}_{\mathcal{R}}) \vee\\
&\qquad (\exists in^{ih}_{\mathcal{R}}, in^{jp}_{\mathcal{S}}\colon in^{ih}_{\mathcal{R}} \sqsubseteq_{\mathcal{TH}} in^{jp}_{\mathcal{S}} \vee in^{jp}_{\mathcal{S}} \sqsubseteq_{\mathcal{TH}} in^{ih}_{\mathcal{R}})]
\end{aligned}
$$

Input and output parameters of two operations are compared if there exists at least a logical implication between operation names. The `intersection` match is verified if some pairs of operations and some pairs of parameters, respectively, are related in any generalization hierarchy.

Example 7 `ItineraryPlanning` (IP) presents an `intersection` match with `Request_2` (R2) since they do not present an `exact`, a `plug-in` or a `subsume` match, but

$\exists op^i_{R2}, op^j_{IP}$ such that $(name\_op^i_{R2} \sqsubseteq_{\mathcal{TH}} name\_op^j_{IP} \vee name\_op^j_{IP} \sqsubseteq_{\mathcal{TH}} name\_op^i_{R2})$		
`FindItineraryOnMap`	$\sqsubseteq_{\mathcal{TH}}$	`Find`
$(\exists out^{ik}_{R2}, out^{jt}_{IP}: out^{ik}_{R2} \sqsubseteq_{\mathcal{TH}} out^{jt}_{IP} \vee out^{jt}_{IP} \sqsubseteq_{\mathcal{TH}} out^{ik}_{R2})$		
`TextualItinerary`	$\sqsubseteq_{\mathcal{TH}}$	`Path`
`GraphicalItinerary`	$\sqsubseteq_{\mathcal{TH}}$	`Path`
$(\exists in^{ih}_{R2}, in^{jp}_{IP}: in^{ih}_{R2} \sqsubseteq_{\mathcal{TH}} in^{jp}_{IP} \vee in^{jp}_{IP} \sqsubseteq_{\mathcal{TH}} in^{ih}_{R2})$		
`StartAddress`	$\equiv_{\mathcal{TH}}$	`InitialAddress`
`EndAddress`	$\equiv_{\mathcal{TH}}$	`Destination`

If all the previous comparisons fail, then the match fails (`mismatch`). A qualitative ranking among the kinds of matches is considered, that is, `exact` > `plug-in` > `subsume` > `intersection` > `mismatch`. To verify these kinds of matches, a DL-based reasoner is used.

4.2 Similarity-based matchmaking model

The comparison between the descriptions of $\mathcal{R}$ and $\mathcal{S}$ can be performed by exploiting a similarity-based matchmaking model through the application of *similarity coefficients* shown in Table 3. These coefficients are based on the Dice's metrics and have been widely experimented [8].

The *Entity-based similarity* coefficient *ESim* evaluates the similarity of I/O parameters of the whole services to measure how much they are based on the same information. Inputs and outputs are considered separately. The building block used to evaluate the similarity is the *Name Affinity* coefficient defined in Section 3.3. Similarity between two sets $OUT_{\mathcal{R}}$ and $OUT_{\mathcal{S}}$ of output parameters is computed through the $A_{tot}()$ function as the sum of values of the *Name Affinity* coefficient between each pair of parameter names, one from $OUT_{\mathcal{R}}$ and one from $OUT_{\mathcal{S}}$:

$$A_{tot}(OUT_{\mathcal{R}}, OUT_{\mathcal{S}}) = \sum_{\substack{out^i_{\mathcal{R}} \in OUT_{\mathcal{R}}, \\ out^j_{\mathcal{S}} \in OUT_{\mathcal{S}}}} NA(out^i_{\mathcal{R}}, out^j_{\mathcal{S}})$$

with the constraint that each parameter name participates at most in one name affinity evaluation. Similarity between input names is computed in the same way. Each component in $ESim()$ formula in Table 3 produces by construction a value belonging to the [0,1] range, so $ESim() \in [0, 2]$.

Table 3 Similarity coefficients between service descriptions $\mathcal{R}$ (request) and $\mathcal{S}$ (advertisement).

Entity-based similarity

$$ESim(\mathcal{R},\mathcal{S}) = \frac{2\cdot A_{tot}(IN_{\mathcal{R}},IN_{\mathcal{S}})}{|IN_{\mathcal{R}}|+|IN_{\mathcal{S}}|} + \frac{2\cdot A_{tot}(OUT_{\mathcal{R}},OUT_{\mathcal{S}})}{|OUT_{\mathcal{R}}|+|OUT_{\mathcal{S}}|} \in [0,2]$$

$IN_{\mathcal{R}}$, $IN_{\mathcal{S}}$ - sets of input parameter names of $\mathcal{R}$ and $\mathcal{S}$
$OUT_{\mathcal{R}}$, $OUT_{\mathcal{S}}$ - sets of output parameter names of $\mathcal{R}$ and $\mathcal{S}$

Operation similarity

$$OpSim(op^i_{\mathcal{R}}, op^j_{\mathcal{S}}) = NA(name\_op^i_{\mathcal{R}}, name\_op^j_{\mathcal{S}}) + \frac{2\cdot A_{tot}(IN^i_{\mathcal{R}},IN^j_{\mathcal{S}})}{|IN^i_{\mathcal{R}}|+|IN^j_{\mathcal{S}}|} + \frac{2\cdot A_{tot}(OUT^i_{\mathcal{R}},OUT^j_{\mathcal{S}})}{|OUT^i_{\mathcal{R}}|+|OUT^j_{\mathcal{S}}|} \in [0,3]$$

$IN^i_{\mathcal{R}}$, $IN^j_{\mathcal{S}}$ - sets of input parameter names of the i-th operation of $\mathcal{R}$ and the j-th operation of $\mathcal{S}$
$OUT^i_{\mathcal{R}}$, $OUT^j_{\mathcal{S}}$ - sets of output parameter names of the i-th operation of $\mathcal{R}$ and the j-th operation of $\mathcal{S}$

Functionality-based similarity

$$FSim(\mathcal{R},\mathcal{S}) = \frac{2\cdot\sum_{i,j} OpSim(op^i_{\mathcal{R}},op^j_{\mathcal{S}})}{|OP_{\mathcal{R}}|+|OP_{\mathcal{S}}|} \in [0,3]$$

$OP_{\mathcal{R}}$, $OP_{\mathcal{S}}$ - sets of operation names of $\mathcal{R}$ and $\mathcal{S}$

Global similarity

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

w_1, w_2 - weights introduced to assess the relevance of each kind of similarity ($w_1 \in [0,1]$ and $w_2 = 1 - w_1$)
$NormESim()$, $NormFSim()$ - $ESim()$ and $FSim()$ normalized to the range [0, 1]

The *Functionality-based similarity* coefficient $FSim$ compares pairs of operations together with their corresponding I/O parameters to measure how much the two services perform the same functionalities. The similarity between two operations is computed through the $OpSim()$ coefficient, that takes into account the affinity between names of operations, similarity between names of their input parameters and similarity between names of their output parameters. Each component in $OpSim()$ formula in Table 3 produces by construction a value belonging to the [0,1] range, so $OpSim() \in [0,3]$ and, accordingly, $FSim() \in [0,3]$.

Finally, $ESim$ and $FSim$ are normalized to the [0,1] range and combined in the *Global similarity* coefficient $GSim$. If the value of $GSim$ is equal or greater than a threshold δ, then $\mathcal{R}$ and $\mathcal{S}$ are considered *similar*. It is noteworthy to say that these coefficients are specifically oriented toward a comparison between services precisely considering the structure of a service description in terms of operations and corresponding I/O parameters rather than viewing service descriptions as pure vectors of terms. The result of this similarity evaluation is, as can be noticed, depending on the choice of δ. The actual value of δ is experimentally set during a training phase as explained in Section 6. Since actual values of $ESim$ and $FSim$ depend on the name affinity evaluation and therefore also depend on the α value, we first set this value to obtain a satisfactory evaluation of the name affinity then we vary the value of δ until an acceptable trade-off between precision and recall is obtained on a training set of service requests and advertisements.

Example 8 Let consider `Request_2` (R2) and `ItineraryPlanning` (IP) service descriptions introduced in the Examples 4 and 2, respectively. If we compute the similarity coefficients in Table 3, we obtain:

$$ESim(R2, IP) = \frac{2 \cdot (1.0 + 1.0)}{5} + \frac{2 \cdot 0.8}{4} = 0.8 + 0.4 = 1.200$$

$$OpSim(\texttt{Find}, \texttt{FindItineraryOnMap}) = 0.8 + \frac{2 \cdot (1.0 + 1.0)}{4} + \frac{2 \cdot 0.8}{3} = 2.333$$

$$FSim(R2, IP) = \frac{2 \cdot 2.333}{3} = 1.556$$

$$GSim(R2, IP) = 0.5 \cdot \frac{1.200}{2} + 0.5 \cdot \frac{1.556}{3} = 0.560$$

given the values of affinity names shown in the following table:

First term	Second term	Name affinity
`FindItineraryOnMap`	`Find`	0.8
`StartAddress`	`InitialAddress`	1.0
`EndAddress`	`Destination`	1.0
`TextualItinerary`	`Path`	0.8
`GraphicalItinerary`	`Path`	0.8

4.3 Hybrid matchmaking model

The deductive and similarity-based matchmaking models can be combined to improve their results. In fact, similarity-based matchmaking is able to quantify the similarity between a request and an advertisement, without distinguishing between the requester and the provider viewpoints. The deductive matchmaking model makes this distinction, performing a formal but qualitative comparison based on a DL reasoner, without quantifying partial matching (`intersection` or `subsume` match). The flexibility of the overall discovery process can be improved by including among the results also partial matches with high values of service similarity.

To this aim, similarity evaluation is applied after the deductive matchmaking procedure, according to the following rules:

- If `exact` or `plug-in` match occurs, from the request viewpoint the service $\mathcal{S}$ provides all the required functionalities, so $GSim(\mathcal{R}, \mathcal{S})$ is set to 1 (full similarity) without directly computing the similarity coefficients;
- If `mismatch` occurs, $GSim(\mathcal{R}, \mathcal{S})$ is directly set to zero;
- If `subsume` or `intersection` match occurs, the advertisement fulfills the request only partially and similarity coefficients are computed to quantify how much the service $\mathcal{S}$ satisfies the request $\mathcal{R}$; in this case, $GSim(\mathcal{R}, \mathcal{S}) \in (0, 1)$.

Furthermore, during the deductive matchmaking procedure, a table is built and maintained to record the correspondences between matching terms of service de-

scriptions. This table is then exploited during similarity evaluation, in which Name Affinity value for pairs of matching terms in the table is directly set to 1 without affinity evaluation.

5 Optimization and ranking

Semantic relationships between *Concrete Services*, *Abstract Services* and *Service Categories* in $\mathcal{S}erv\mathcal{ONT}$ are exploited to make more efficient the service discovery procedure. The matchmaking models (deductive, similarity-based and hybrid) are applied at the *Category* and *Abstract* layers and can be optimized by considering the semantic relationships between *Abstract Services* according to the following intuition: if an *Abstract Service* $\mathcal{S}_a^i$ matches with a given service request $\mathcal{R}$, then also *Abstract Services* providing the same functionalities of $\mathcal{S}_a^i$ (as expressed by means of the `plug-in` match) match with $\mathcal{R}$. The resulting candidate *Abstract Services* are then replaced with their corresponding *Concrete* ones without applying the matchmaking procedure for each *Concrete Service* and only *Concrete Services* that present a similarity value equal or greater than the threshold δ and a kind of match different from `mismatch` are proposed as search results. Finally, the selected *Concrete Services* are ranked with respect to their kind of match and similarity value.

Example 9 `DisplayGasInfrastructure` (DGI) presents a `plug-in` match with `Request_1` (R1), since

$\forall op^i_{R1}\ \exists op^j_{DGI}$ such that $(name\_op^i_{R1} \sqsubseteq_{TH} name\_op^j_{DGI}) \wedge (\forall out^{ik}_{R1}\ \exists out^{jt}_{DGI}: out^{ik}_{R1} \sqsubseteq_{TH} out^{jt}_{DGI})$		
`VectorialMapVisualization`	$\sqsubseteq_{TH}$	`Visualization`
`Road`	$\equiv_{TH}$	`RoadInfrastructure`
$\forall op^i_{R1}\ \exists op^j_{DGI}$ such that $(\forall in^{jp}_{DGI}\ \exists in^{ih}_{R1}: in^{jp}_{DGI} \sqsubseteq_{TH} in^{ih}_{R1})$		
`GeographicCoordinates`	$\sqsubseteq_{TH}$	`Position`

This match can be directly derived from the fact that:

$$\text{match}(\texttt{DisplayStreets}, \texttt{DisplayGasInfrastructure}) = \text{“}\texttt{plug-in}\text{”}$$
$$\text{match}(\texttt{Request\_1}, \texttt{DisplayStreets}) = \text{“}\texttt{plug-in}\text{”}$$
$$\Rightarrow \text{match}(\texttt{Request\_1}, \texttt{DisplayGasInfrastructure}) = \text{“}\texttt{plug-in}\text{”}$$

The algorithm for the overall matchmaking process is shown in Figure 3. The solution is primarily based on the functional comparison of service descriptions, so we called the algorithm Fc(Functional Comparison)-Match. The algorithm starts with some initialization instructions (rows 6-7), loading from the service ontology the list of *Abstract Services*, that constitute the service advertisements to be compared with the request. After performing subsumption checking between *Service Categories* (subsumption-checking function on row 9 checks if there exists $cat^i_{\mathcal{S}} \in CAT_{\mathcal{S}}$ such that $cat^i_{\mathcal{S}} \sqsubseteq_{TH} cat_{\mathcal{R}}$) with respect to the domain ontology $\mathcal{D}om\mathcal{ONT}$ and thesaurus

```
      FC-MATCH
(1)   Inputs: a request R;
(2)           ServONT; DomONT; TH;
(3)           a threshold δ;
(4)   Outputs: a list Results of candidate concrete services CS_S
              with GSim(R, CS_S) ≥ δ and TypeOfMatch <> 'mismatch',
              ranked with respect to TypeOfMatch and values of GSim();

(5)   begin
(6)     load from ServONT into AS the list of abstract services;
(7)     Candidates := ∅; Results := ∅;
(8)     foreach S ∈ AS do
(9)       if(SUBSUMPTION-CHECKING(cat_R,CAT_S,DomONT,TH) == true) then
(10)        if(S ∉ Candidates) then
(11)          remove S from AS;
(12)          [TypeOfMatch_S, CTable] := CLASSIFY-MATCH(R,S,DomONT,TH);
(13)          if(TypeOfMatch_S <> 'mismatch') then
(14)            GSim(R, S) := EVALUATE-SIMILARITY(R,S,TH,CTable);
(15)            add < S,TypeOfMatch_S,GSim(R, S) > to Candidates;
(16)            foreach S_i ∈ AS do
(17)              if S_i is a specialization of S in ServONT AND
                      (TypeOfMatch_S ='exact' OR TypeOfMatch_S = 'plugin') then
(18)                TypeOfMatch_Si := 'plugin';
(19)                GSim(R, S_i):=GSim(R, S);
(20)                add to Candidates the tuple < S_i, TypeOfMatch_Si,GSim(R, S_i) >;
(21)                remove S_i from AS;
(22)              endif
(23)            endforeach
(24)          else
(25)            foreach S_i ∈ AS do
(26)              if S_i is a generalization of S in ServONT then
(27)                remove S_i from AS;
(28)            endforeach
(29)          endif
(30)        endif
(31)      endif
(32)    endforeach
(33)    foreach < S, TypeOfMatch_S,GSim(R, S)>∈ Candidates do
(34)      foreach concrete service CS_S associated with S in ServONT do
(35)        add < CS_S,TypeOfMatch_S,GSim(R, S) > to Results;
(36)    endforeach
(37)    remove from Results the tuples with GSim(R, S) < δ;
(38)    order tuples in Results on the basis of TypeOfMatch and GSim(R, S) values;
(39)    return Results;
(40)  end
```

Figure 3 The Fc-Match hybrid matchmaking algorithm.

$\mathcal{TH}$), the comparison is performed in three steps (rows 10–36): (1) through the application of deductive matchmaking model, the kind of match between the request $\mathcal{R}$ and candidate service $\mathcal{S}$ is classified (CLASSIFY-MATCH function on row 12), as proposed in Section 4.1; (2) hybrid matchmaking procedure is applied by evaluating similarity coefficients only in case of partial matching (EVALUATE-SIMILARITY function on row 14); (3) optimization techniques based on $\mathcal{ServONT}$ are exploited (rows

16–28 and rows 33–36). Finally, filtering out of not relevant results (rows 37) and ranking (row 38) are performed. Note that a table *CTable* of correspondences between matching terms is returned as result of the deductive step (row 12) and then the table is exploited during similarity evaluation (row 14) as explained in Section 4.3.

As we discussed, FC-MATCH exploits a reasoner based on $\mathcal{SHOIN}(\mathcal{D})$ description logic and this task is considered quite hard from a computational perspective. Actually, the complexity of deductive facilities depends on the expressive power of the used description logic. For instance, complexity of subsumption checking in $\mathcal{SHOIN}(\mathcal{D})$ is NEXPTIME [16]. However, optimized DL deductive algorithms are suitable so that real systems (e.g., RACER [14]) implementing them perform well on a high percentage of tasks related to real problems.

6 Experimentation results

To support our approach we implemented the COMPAT MatchMaker Version 1.0, developed in Java as a Web application [2]. The MatchMaker exploits deductive facilities of RACER OWL-DL reasoner Version 1.9. Experiments have been performed: (1) to confirm the advantages derived from a combined use of domain ontology and thesaurus if compared with traditional approaches; (2) to demonstrate better precision-recall results of our approach with respect to other service matchmaking strategies, which use only deductive-based or similarity-based techniques or apply hybrid matchmaking without combining terminological information and domain knowledge or without using similarity coefficients tailored to services. To enable better comparison, we referred to an available test collection, OWLS-TC V1,[1] consisting of a set of more than 400 OWL-S services, retrieved from public IBM UDDI registries, covering six domains (education, healthcare, food, tourism, communication and economy). In a (semi-automatic) pre-processing phase, we extracted the WSDL interfaces from OWL-S services, considering both the OWL-S profile (containing information about I/O parameters) and concepts expressing the meaning of service operations (partially obtained from the names of atomic processes contained in the OWL-S process model). We added additional services up to a total of 480 Web Services specified in WSDL1.1 and manually classified according to the UNSPSC taxonomy. Domain ontologies for each selected domain have been obtained from the test collection. Finally, a set of ten service requests have been formulated (four for tourism, three for healthcare and three for economy domain), using both terms extracted from the domain ontologies and terms that are not defined as atomic concepts in the ontologies. For each request, suitable advertisements have been manually selected from the test collection, to allow precision-recall evaluation.

Figures 4a–b and Figure 5 compare matchmaking procedures through the precision-recall curves. We evaluated the average values of precision and recall for the set of service requests and varied the global similarity threshold δ, used to filter out not relevant results. Figure 4a shows how application of the affinity-based subsumption test produces better results than a traditional subsumption test

[1] http://projects.semwebcentral.org/projects/owls-tc/.

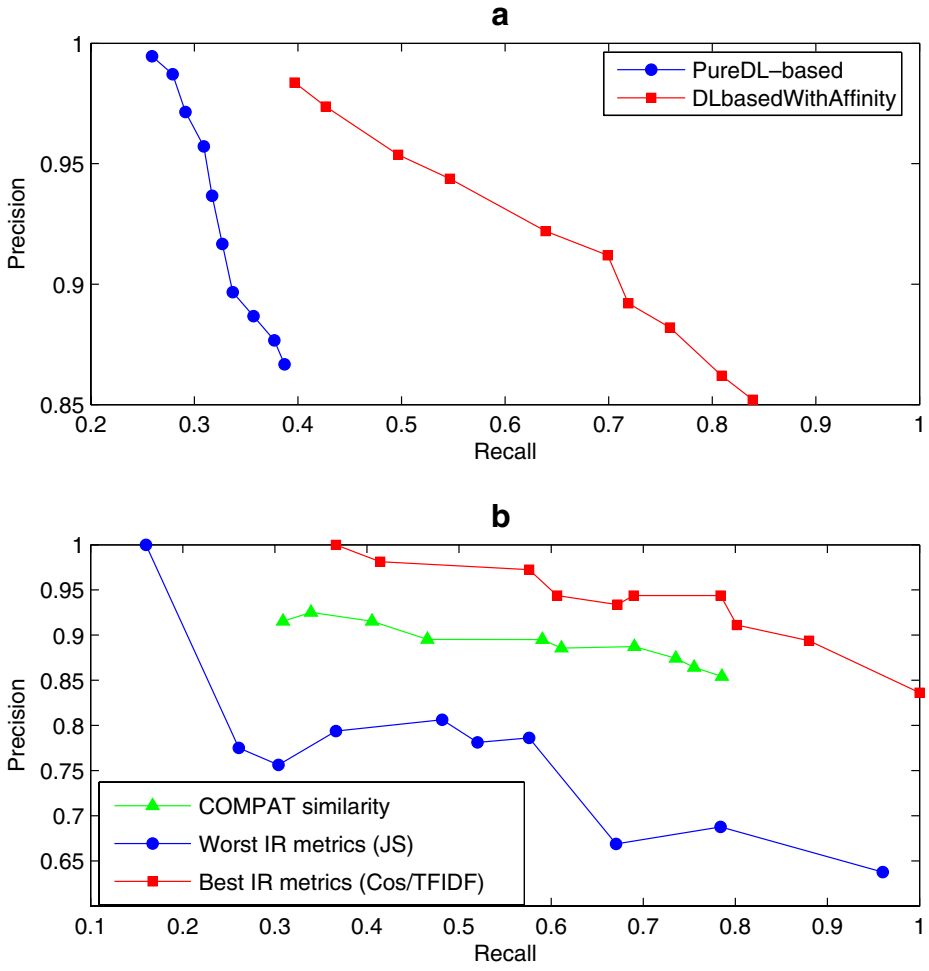

Figure 4 **a** Comparison of precision/recall for matchmaking techniques based on affinity-based subsumption test with respect to the traditional subsumption checking and **b** comparison of COMPAT similarity coefficients with other IR metrics.

based only on the domain ontology by improving the precision-recall trade-off. If all the terms used for operation and I/O parameter names of service requests are atomic concepts in the domain ontology, precision and recall values are almost the same of traditional DL-based approaches [15], featured by high precision and low recall values, otherwise recall is improved by the affinity-based subsumption test. Figure 4b underlines how similarity-based coefficients perform near the best IR similarity metrics. Comparison is made with *Cosine* and *extended Jacquard (JS)* similarity metrics with standard TFIDF term weighting scheme, computed as:

$$Sim_{COS}(\mathcal{R},\mathcal{S}) = \frac{\overline{\mathcal{R}}\cdot\overline{\mathcal{S}}}{||\overline{\mathcal{R}}||_2^2\cdot||\overline{\mathcal{S}}||_2^2} \quad Sim_{JS}(\mathcal{R},\mathcal{S}) = \frac{\overline{\mathcal{R}}\cdot\overline{\mathcal{S}}}{||\overline{\mathcal{R}}||_2^2+||\overline{\mathcal{S}}||_2^2-\overline{\mathcal{R}}\cdot\overline{\mathcal{S}}}$$

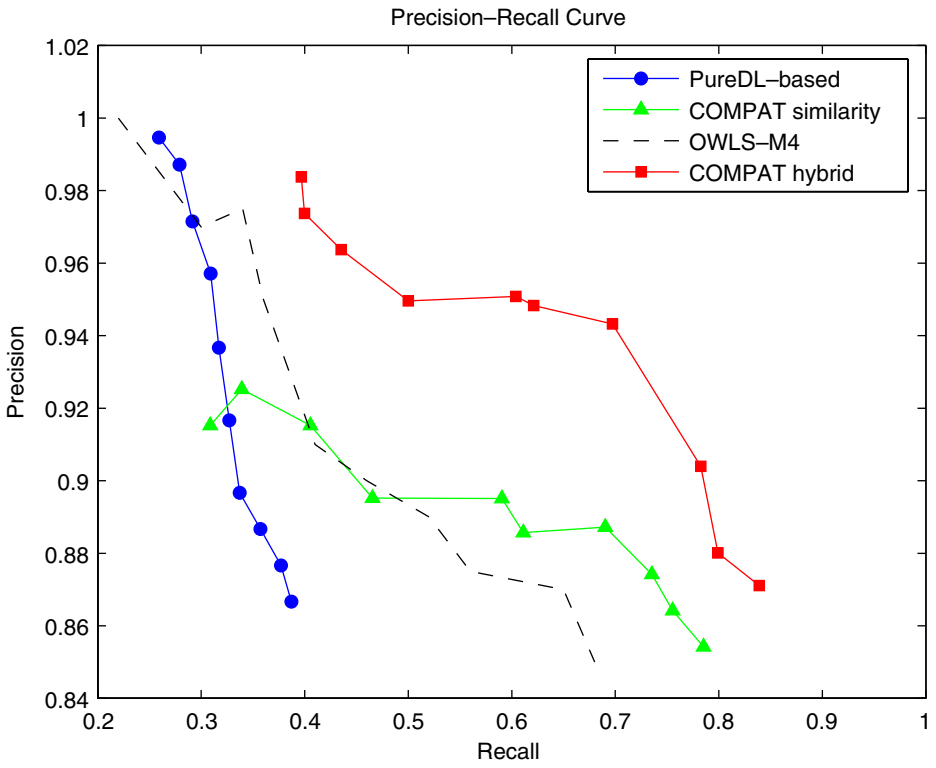

Figure 5 Comparison of COMPAT hybrid matchmaking with other approaches in literature.

where request $\mathcal{R}$ and advertisement $\mathcal{S}$ are represented as weighted index term vectors $\overline{\mathcal{R}}$ and $\overline{\mathcal{S}}$, without distinguishing among operation names or I/O parameter names, $\overline{\mathcal{R}} \cdot \overline{\mathcal{S}} = \sum_{i=1}^{n} \omega_{i,\mathcal{R}} \cdot \omega_{i,\mathcal{S}}$, $||\overline{X}||_2 = \sqrt{\sum_{i=1}^{n} \omega_{i,X}^2}$ and $\omega_{i,X}$ denotes the weight of the i-th index term in vector X. Our similarity coefficients are tailored to service description comparisons and considers also operation names with corresponding I/O parameters. For this reason, our coefficients better perform when $\mathcal{R}$ and $\mathcal{S}$ are constituted by more than one operation with corresponding I/O parameters. However, to be compliant with the available test collection, we mainly considered service functional descriptions with only one operation.

Finally, Figure 5 shows the precision-recall curves to compare our hybrid matchmaking approach with the hybrid matchmaker OWLS-MX (OWLS-M4) exposed in [13], with a pure DL-based approach and with the similarity-based matchmaking. Hybrid matchmaking combines high precision of DL-based strategies with high recall of similarity metrics. The hybrid approach introduced in [13] does not present the same performances of COMPAT MatchMaker, since it does not use a corresponding version of affinity-based subsumption test or similarity coefficients tailored to service interface structure. Figure 6 shows how our hybrid approach performs quite linearly with the number of considered services, but presents valuable response time also for high number of available services registered in the UDDI registry. Scalability

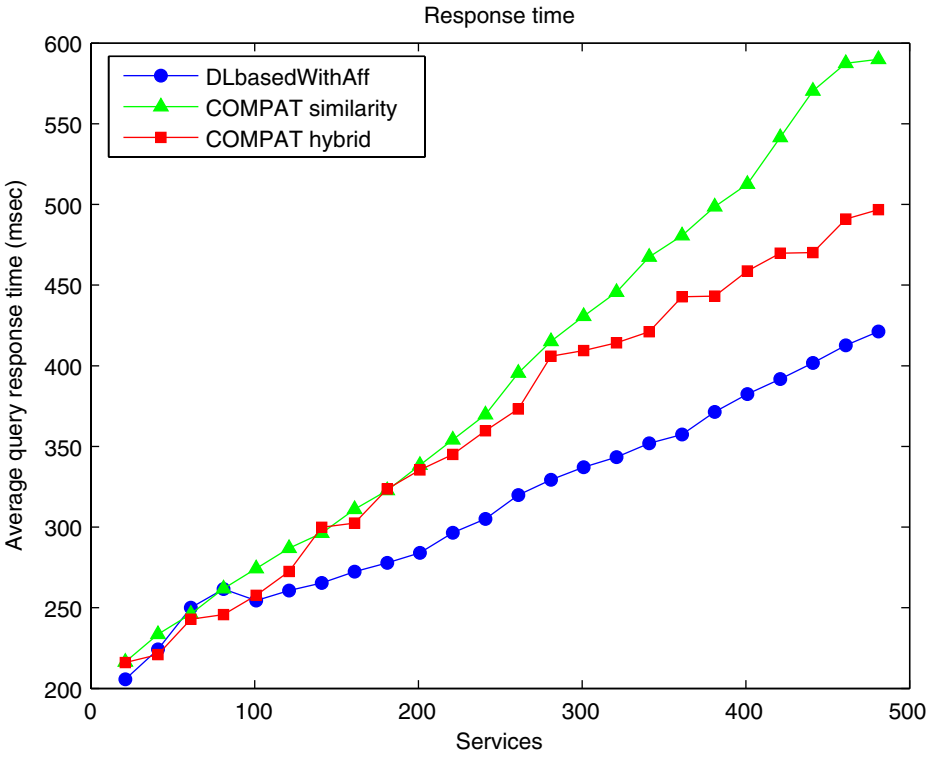

Figure 6 Average response time of hybrid, deductive and similarity-based matchmaking models.

is improved by optimization policies that exploit organization of services through semantic relationships in $\mathcal{S}erv\mathcal{O}NT$.

7 Concluding remarks

The paper presents a novel approach to service discovery that combines a fully deductive matchmaking based on description logics and a similarity-based matchmaking derived from the information retrieval area. The approach also exploits semantic organization of services in a service ontology to speed up service discovery. The combination of two types of matchmaking strategies enhances effectiveness and flexibility of the discovery process, since not only complete matches are considered, but also partial ones are taken into account by evaluating the similarity degree. As a consequence, this evaluation decreases false negatives, since it does not exclude from the searching results those services that do not have `exact` or `plug-in match`, but that are quite similar from the functional viewpoint with the request. Finally, the use of service ontology improves efficiency by reducing the number of required matchings. Future work will investigate the extension of this approach to distributed environments, such as P2P networks, where services are published on several UDDI

registries on distinct peers. In particular, P2P semantic relationships between services could be used to improve discovery process.

Acknowledgements This work has been partially supported by the Italian MIUR/MURST FIRB MAIS (Multichannel Adaptive Information Systems) Project [20] and by NoE INTEROP [19] IST Project n. 508011–6th EU Framework Program.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Theory, Implementation and Applications. Cambridge University Press (2003)
2. Bianchini, D., De Antonellis, V., Melchiori, M.: An ontology-based architecture for service discovery and advice system. In: Proceedings of the IEEE DEXA International Workshop on Web Semantics (WebS2005), pp. 551–556. Copenhagen, Denmark (2005)
3. Bianchini, D., De Antonellis, V., Melchiori, M.: Capability matching and similarity reasoning in service discovery. In: Proceedings of the CAiSE International Workshop on Enterprise Modeling and Ontologies for Interoperability (EMOI 2005), pp. 285–296. Porto, Portugal (2005)
4. Bianchini, D., De Antonellis, V., Pernici, B., Plebani, P.: Ontology-based methodology for *e*-service discovery. Journal of Information Systems, Special Issue on Semantic Web and Web Services **31**(4–5), 361–380 (2006)
5. Bianchini, D., De Antonellis, V., Melchiori, M., Salvi, D.: Semantic-enriched service discovery. In: Proceedings of IEEE ICDE International Workshop on Challenges in Web Information Retrieval and Integration (WIRI 2006), pp. 38–47. Atlanta, Georgia, USA (2006)
6. Bussler, C., de Bruijn, J., Feier, C., Fensel, D., Keller, U., Lara, R., Lausen, H., Polleres, A., Roman, D., Stollberg, M.: Web service modeling ontology. Applied Ontology, IOS Press, **1**, 77–106 (2005)
7. Calì, A., Calvanese, D., Colucci, S., Di Noia, T., Donini, F.M.: A logic based approach for matching user profiles. In: Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES 2004), Lecture Notes in Artificial Intelligence, vol. 3215, pp. 187–195 (2004)
8. Castano, S., De Antonellis V.: A framework for expressing semantic relationships between multiple information systems for cooperation. Inf. Syst. **123**(3–4), 253–277 (1998)
9. Castano, S., Ferrara, A., Montanelli, S., Racca, G.: Semantic information interoperability in open networked systems. In: Proceedings of the International Conference on Semantics of a Networked World (ICSNW 2004), in cooperation with ACM SIGMOD 2004, pp. 215–230. Paris, France (2004)
10. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL-DL vs. OWL flight: Conceptual modeling and reasoning for the semantic web. In: Proceedings Forum of the 14th International World Wide Web Conference (WWW 2005), pp. 623–632. Chiba, Japan (2005)
11. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB2004), pp. 372–383. Toronto, Canada (2004)
12. Fellbaum, C.: Wordnet: An Electronic Lexical Database. MIT Press (1998)
13. Fries, B., Khalid, M., Klusch, M., Sycara, K.: OWLS-MX: hybrid OWL-S service matchmaking. In: Proceedings of First International AAAI Symposium on Agents and Semantic Web, pp. 77–84. Arlington, VA, USA (2005)
14. Haarslev, V., Moller, R., Wessel, M.: RACER User's Guide and Reference Manual Version 1.7.19. http://www.sts.tu-harburg.de/~r.f.moeller/racer/ (2004)
15. Horrocks, I., Li, L.: A software framework for matchmaking based on semantic web technology. In: Wellman, M., Riedl, J. (eds.) International Journal of Electronic Commerce, Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce (2004)
16. Horrocks, I., Patel-Schneider, P.: Reducing OWL entailment to description logic satisfiability. In: Proceedings of the 2nd International Semantic Web Conference (ISWC 2003), Lectures Notes in Computer Science, no. 2870, pp. 17–29. Springer (2003)

17. Lauer, M.: Designing statistical language learners: experiments on noun compounds. Ph.D. thesis, Department of Computing Macquarie University NSW 2109, Australia (1995)
18. Lausen, H., de Bruijn, J., Polleres, A., Fensel, D.: WSML—a language framework for semantic web services. In: Proceedings of the W3C Workshop on Rule Languages for Interoperability, Washington DC, USA (2005)
19. The INTEROP NoE portal: http://www.interop-noe.org/
20. The MAIS Project Home Page: http://www.mais-project.it/
21. The OWL Service Coalition: OWL-S 1.1 release. http://www.daml.org/services/owl-s/1.1/ (2004)
22. Pernici, B. (ed.): Mobile Information Systems. Infrastructure and Design for Adaptivity and Flexibility. Springer (2006)