Capability Matching and Similarity Reasoning in Service Discovery *

D. Bianchini, V. De Antonellis, M. Melchiori

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

Abstract. Ontologies play a relevant role to support service matchmaking in the discovery process. In fact, the elements used for service capability description refer to concepts that can be properly defined and semantically related in domain ontologies. Semantic relationships between concepts are then exploited to establish the type of matching between advertisements and requests. In this paper we propose an ontology-based approach to service discovery characterized by a hybrid multimode matching, that is, a deductive capability matching extended with a flexible similarity evaluation scheme. In the approach, the semantic service description results from the cooperation of several components: a UDDI registry is responsible for managing offered service descriptions, a Domain Ontology provides the general knowledge about concepts from the business domain in which services are used, a Service Ontology organizes services at different levels of abstraction. The proposed approach provides service advice at multiple levels of granularity and rates adviced services according to different kinds of comparison strategies.

1 Introduction

In the recent years many companies have heavily invested in Web Service technologies and, as a consequence, a growing number of services is being made available. A Web Service is a set of related functionalities that can be accessed through the Web and provides a way for interoperating independently developed Web applications. Service proliferation over the Web has been facilitated by the development of several standards, like WSDL for service description, UDDI for service registry, SOAP for message exchange and BPEL4WS for service orchestration [5]. In the future, it is envisaged that application development will be mainly based on the composition of services published and made available by third-party providers. Following this direction, advanced techniques and tools for enabling semantic service discovery are highly desired and required. In particular, it is necessary that services are described in a formal way and service semantics is well captured. In the literature, ontology-based approaches are being developed to exploit the benefits of the ontology technology, such as inferencing,

* This work has been partially supported by the European EU NoE INTEROP [14] and by the MAIS (Multichannel Adaptive Information Systems) FIRB Project funded by the Italian Ministry of Education, University and Research [17].

in the context of service discovery. In the Semantic Web, the ontology description languages OWL [8] and OWL-S [7] have been recently proposed. A sub-language of OWL, OWL-DL, implements in a XML-based format the $\mathcal{SHOIN}(\mathcal{D}+)$ Description Logic, that is a knowledge representation formalism with a sound, complete and decidable inference procedure [1]. Further, the trade-offs between expressivity and computational complexity have been widely studied. OWL-S is a service ontology specified in OWL. Service description is composed by a service profile (what the service does), a service model (how the service works) and a service grounding (how to invoke the service).

The use of ontologies specifically enables service matchmaking in the discovery process. In fact, the elements used for service capability description refer to concepts that can be properly defined and semantically related in domain ontologies. Semantic relationships between concepts are then exploited to establish the type of matching between advertisements and requests. In this paper we propose an ontology-based approach to service discovery. In the approach, the semantic service description results from the cooperation of several components: a UDDI registry is responsible for managing offered service descriptions, a Domain Ontology provides the general knowledge about concepts from the business domain in which services are used, a Service Ontology organizes services at different levels of abstraction. The service discovery process is based on a hybrid multimode matching, that is, a deductive capability matching extended with a flexible similarity evaluation scheme. The proposed approach has the advantage of providing service advice at multiple levels of granularity and of rating adviced services according to different kinds of comparison strategies. In Section 2 we briefly discuss the related work. Section 3 formally defines the semantic service modeling components. Section 4 describes the hybrid multimode matching approach. Section 5 presents the Service Ontology architecture and its deployment. In Section 6 we conclude with advantages of the proposed approach and future work.

2 Related work

Service matchmaking has been addressed by several approaches in literature: given a request $\mathcal{R}$ and a set of advertisements $\mathcal{S}$, the matching procedure must return the set of advertised services that match better with $\mathcal{R}$, possibly ranked with respect to their level of matching (if it can be evaluated). In most approaches the starting point is the UDDI Registry, where service descriptions are published; UDDI Registry offers searching functionalities that use traditional keyword-based techniques, featured by low precision and recall. To provide semantic matching between service descriptions, some approaches consider concept definitions within ontologies (concept-based techniques). In [19] a framework for semi-automatically marking up Web service descriptions with ontologies is proposed with algorithms to match and annotate WSDL files with relevant ontologies; domain ontologies are used to categorize Web services into domains. In [16] a Web Service Modeling Ontology (WSMO) is expressed by using the formal F-Logic language to describe various aspects related to Semantic Web Services. They start from the Web Service Modeling Framework (WSMF) [10], that consists of four elements: ontologies that provide terminology used by other elements (concepts, axioms, relations and instances), goals of Web Services (by

means of pre- and post-conditions), Web Service description (non functional properties, choreography and orchestration aspects) and mediators which bypass interoperability problems. Sycara et al. [15] use OWL-S service profile (inputs, outputs, preconditions and effects) and exploit semantic relationships between concepts in the ontology to verify if inputs and outputs of request and advertisement are related by any generalization hierarchy in the ontology. Moreover, a logic formalism based on Horn clauses is provided to verify logical implication between preconditions and effects. The ATLAS matchmaker [18] considers DAML-S ontologies and utilizes two separate sets of filters: 1) matching non functional attributes by performing conjunctive pair-wise comparison to determine the applicability of advertisements (i.e., do they deliver sufficient quality of service, etc); 2) matching service functionalities to determine if the advertised service matches the requested service by using a DAML-based subsumption inference engine to compare input and output sets. Also other approaches use logical formalisms to enhance service matchmaking. Horrocks et al. [12] express OWL-S service profile by means of Description Logics and exploit inference mechanisms of a Description Logic reasoner (such as Racer [20]) to establish the degree of match between advertisements and requests. Deductive approaches offer high precision and recall, but can present high complexity. Benatallah et al. [2] formalize service discovery as a new instance of the problem of rewriting concepts using terminologies in the framework of DL-based ontologies and propose a hypergraph-based matching algorithm that takes as input a service request and an ontology of services and find a set of services whose descriptions contain as much as possible of common information with the request and less as possible extra information w.r.t. the request. Bernstein and Klein [3] propose process ontologies to describe the behaviour of services and query such ontologies using a Process Query Language (PQL). In [11] is reported an experience in building matching prototype based on Description Logic reasoner which considers service descriptions in DAML+OIL and applies a matching algorithm based on simple subsumption and consistency tests.

Our aim in this work is to propose an ontology-based hybrid approach where different kinds of comparison strategies are combined together to provide a flexible and efficient matchmaking between service descriptions. We extends the keyword-based approach of UDDI Registry by considering semantic matching on the basis of a Domain Ontology (where elements used for service description refer to concepts semantically described and related) and a Service Ontology (where services are organized by means of semantic relationships at multiple levels of granularity). Moreover, we consider service operation names as meaningful for matching purposes. Service comparison is performed on the basis of the Domain Ontology by means of a DL-based approach, made more flexible by exploiting a similarity evaluation scheme and more efficient by exploiting semantic relationships in the Service Ontology. Furthermore, similarity evaluation scheme is used also to make more precise ranking of searching results.

3 Service modeling using ontologies

It is widely accepted that ontologies are a formal tool useful for modeling semantics and support inferencing procedure. For effective service discovery, the

use of ontologies for modeling service semantics is recommended. In our approach, a Domain Ontology is used to conceptualize domain knowledge and to provide semantics to service descriptions. The Domain Ontology contains definitions of concepts to which elements used to describe services (input/output parameters, operation names, service categories) refer. Concepts are related by means of the usual semantic relationships: *concept generalization/specialization*, *concept equivalence*, *concept disjunction* and *instance-of*. The Domain Ontology therefore supports service matchmaking. Reasoning on service properties viewed as concepts in the Domain Ontology allows to identify semantic relationships also among services. In our approach, we explicitly represent such semantic relationships in a Service Ontology where services are described at different levels of abstraction to support multimode matching and querying. The Service Ontology architecture is described in Section 5. Ontologies are described using $\mathcal{SHOIN}(\mathcal{D}+)$ Description Logic which has been implemented using the XML-based syntax of OWL-DL sublanguage. In the literature OWL-S and WSMO are the emerging languages for semantic service description, but there is no accepted standard. In our approach, a service is semantically described by means of a category, representing the domain of interest of the service, and the service capability in terms of provided operations. Formally, the description is given by a conjunction of:

- a concept in the form $\exists$`hasCategory`.CAT, where CAT is a concept which represents the associated service category;
- one or more concepts in the form $\exists$`hasOperation`.OP, where OP is a concept representing an operation of the current service; each operation OP is described as a conjunction of:
  - the operation name, expressed by means of an atomic concept;
  - a conjunction of one or more concepts IN, where IN is a concept representing an input parameter;
  - a conjunction of one or more concepts OUT, where OUT is a concept representing an output parameter.

IN and OUT are specified in the form $\exists R.C$, where R represents the name of the parameter and C is a concept representing possible parameter values. C can be defined as an atomic concept, an enumeration $\{i_1, i_2, \ldots i_n\}$ of individuals or a complex concept obtained by applying the *intersection* operator ($\sqcap$), the *union* operator ($\sqcup$) and the *negation* operator ($\neg$) to other concepts.

Example. We consider a ticket reservation service for flights from European to US cities. Using Description Logic formalism, we can write:

$$\texttt{FlightBookingEuropeToUSA} \sqsubseteq \exists \texttt{hasCategory.AirTransfer} \sqcap \exists \texttt{hasOperation.(flightBooking} \sqcap \exists \texttt{departureCity.EuropeanCity} \sqcap \exists \texttt{arrivalCity.USCity} \sqcap \exists \texttt{departureDate.Date} \sqcap \exists \texttt{arrivalDate.Date} \sqcap \exists \texttt{ticket.flightTicket)}$$

while in the Domain Ontology we have the following assertions, to express that `EuropeanCity` and `USCity` are cities and represent disjoint sets of objects, that `flightBooking` and `bookFlight` and that `flightTicket` and `ticketReceipt` constitute equivalent concepts:

$$\begin{aligned} \mathcal{DOMONT}\ & \texttt{EuropeanCity} \sqsubseteq \texttt{City} \\ & \texttt{USCity} \sqsubseteq \texttt{City} \\ & \texttt{EuropeanCity} \sqsubseteq \neg \texttt{USCity} \\ & \texttt{flightBooking} \equiv \texttt{bookFlight} \\ & \texttt{flightTicket} \equiv \texttt{ticketReceipt} \end{aligned}$$

In this example, we considered only one operation (`flightBooking`). □

4 Hybrid multimode service matchmaking

4.1 Deductive matching

In a deductive approach based on Description Logics the request $\mathcal{R}$ is compared with an offer $\mathcal{S}$ by exploiting the underlying Domain Ontology.

To verify matching between a description $\mathcal{S}$ and a request $\mathcal{R}$, we verify the satisfiability w.r.t. $\mathcal{DOMONT}$ between concepts to which the service description elements refer. As in [12], 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 $\mathcal{R}$ and $\mathcal{S}$ 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.

Note that, from the request viewpoint, the first two kinds of match can be considered equivalent, since in both cases the offer fulfills the request; in the case of *subsume* and *intersection match*, otherwise, the offer satisfies only partially the request.

To verify the five kinds of match listed above, we separate service description components by considering service categories, operation names and input/output parameter names for each operation. Firstly, we consider the service categories CAT_R of the request and CAT_S of the offer and we verify if

$$\mathcal{DOMONT} \models CAT_R \sqsubseteq CAT_S$$

that is, $CAT_R \sqsubseteq CAT_S$ is true in $\mathcal{DOMONT}$. If this is not verified, then the match fails (*mismatch*), otherwise the other kinds of match are investigated.

Exact match. It occurs when, for each operation $OP_{i\,R}$ there exists a corresponding operation $OP_{j\,S}$ such that

- $\mathcal{DOMONT} \models OP_{i\,R}.name \equiv OP_{j\,S}.name$;
- for each output $OP_{i\,R}.OUT_h$ there exists a corresponding output $OP_{j\,S}.OUT_q$ such that $\mathcal{DOMONT} \models OP_{i\,R}.OUT_h \equiv OP_{j\,S}.OUT_q$;
- for each input $OP_{j\,S}.IN_p$ there exists a corresponding input $OP_{i\,R}.IN_k$ such that $\mathcal{DOMONT} \models OP_{j\,S}.IN_p \equiv OP_{i\,R}.IN_k$.

Each operation of $\mathcal{R}$ is compared with each operation of $\mathcal{S}$. Note that in the matching 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 corresponding operations (corresponding parameters) such operations (parameters) do not participate in further comparisons.

Plug-in match. It occurs when, for each operation OP_{iR}, there exists a corresponding operation OP_{jS} such that

- $\mathcal{DOMONT} \models OP_{iR}.name \sqsubseteq OP_{jS}.name$;
- for each output $OP_{iR}.OUT_h$ there exists a corresponding output $OP_{jS}.OUT_q$ such that $\mathcal{DOMONT} \models OP_{iR}.OUT_h \sqsubseteq OP_{jS}.OUT_q$;
- for each input $OP_{jS}.IN_p$ there exists a corresponding input $OP_{iR}.IN_k$ such that $\mathcal{DOMONT} \models OP_{jS}.IN_p \sqsubseteq OP_{iR}.IN_k$.

The *subsume match* is verified in the same way, with the roles of $\mathcal{R}$ and $\mathcal{S}$ exchanged.

Intersection match. If neither *exact* or *plug-in* or *subsume match* occurs, but there exist pairs of operations OP_{iR} and OP_{jS} with the following conditions verified

- $\mathcal{DOMONT} \models \neg(OP_{iR}.name \sqcap OP_{jS}.name \sqsubseteq \bot)$;
- for at least one output $OP_{iR}.OUT_h$ there exists a corresponding output $OP_{jS}.OUT_q$ such that $\mathcal{DOMONT} \models \neg(OP_{iR}.OUT_h \sqcap OP_{jS}.OUT_q \sqsubseteq \bot)$;
- for at least one input $OP_{jS}.IN_p$ there exists a corresponding input $OP_{iR}.IN_k$ such that $\mathcal{DOMONT} \models \neg(OP_{jS}.IN_p \sqcap OP_{iR}.IN_k \sqsubseteq \bot)$;

then *intersection match* is recognized between $\mathcal{R}$ and $\mathcal{S}$.

If all the previous comparisons fail, then the match fails (*mismatch*). We can recognize a qualitative ranking among the considered kinds of match, that is, `exact` > `plug-in` > `subsume` > `intersection` > `mismatch`. To verify these kinds of match, an automatic reasoner based on Description Logics is used (Racer [20]).

Example. Consider the `FlightBookingEuropeToUSA` service shown above ($\mathcal{S}$) and the following request, to book flights between European cities.

$\mathcal{R} \sqsubseteq$ `∃hasCategory.AirTransfer ⊓ ∃hasOperation.(bookFlight ⊓`
`∃departureCity.EuropeanCity ⊓ ∃arrivalCity.EuropeanCity ⊓ ∃ticket.ticketReceipt)`

Since the offer and the request have the same service category, we can proceed to verify the kind of match between them; in this case we have an *intersection match*. The offer partially fulfills the request, because

- $\mathcal{DOMONT} \models$ `bookFlight` $\equiv$ `flightBooking`;
- $\mathcal{DOMONT} \models$ `(∃ticket.ticketReceipt` $\equiv$ `∃ticket.flightTicket)`;
- *exact*, *plug-in* and *subsume match* fails, since $\mathcal{DOMONT} \models$ `(∃arrivalCity.EuropeanCity ⊓ ∃arrivalCity.USCity)` $\sqsubseteq \bot$, while the other input/output parameters are related in some generalization hierarchies. □

4.2 Similarity-based matching

In [9] a similarity-based approach is used for searching Web Services described in WSDL. In [4] a methodology to evaluate the degree of functional similarity between services on the basis of comparison of their descriptions has been proposed. The similarity between services is evaluated through the computation

of coefficients obtained by comparing input/output parameter names (*Entity-based similarity coefficient*) and operation names (*Functionality-based similarity coefficient*) [6]. To obtain these coefficients, we assign a weight to each kind of relationship in the Domain Ontology between concepts associated to operation and I/O parameter names. In particular, given two names n_h and n_k, we define the *Name Affinity coefficient* between them, denoted by $NA(n_h, n_k)$, in the following way:

$$NA(n_h, n_k) = \begin{cases} 1 & if\ C_{nh} \equiv C_{nk} \\ 0.8^L & if\ C_{nh} \sqsubseteq C_{nk}\ or\ viceversa\ with\ L\ levels\ of \\ & generalization/specialization \\ 0 & otherwise \end{cases} \quad (1)$$

where C_{nh} and C_{nk} are the concepts associated to the names n_h and n_k, respectively. We say that n_h and n_k have *Name Affinity* ($n_h \sim n_k$) if $NA(n_h, n_k) \geq \alpha$, where $\alpha > 0$ is a given threshold imposed to filter names with high degree of affinity.

The Name Affinity coefficient is used to compute the values for the *Entity-based similarity coefficient* and the *Functionality-based similarity coefficient.*

Entity-based similarity coefficient. Given two service descriptions $\mathcal{R}$ and $\mathcal{S}$, we denote with IN_R and IN_S (resp., OUT_R and OUT_S) the sets of input parameter names (resp., output parameter names) of the overall services. The *Entity-based similarity coefficient* between $\mathcal{R}$ and $\mathcal{S}$, denoted by $ESim(\mathcal{R}, \mathcal{S})$, is the measure of affinity between names of I/O parameters, considered in their totality, that is

$$ESim(\mathcal{R}, \mathcal{S}) = \frac{2 \cdot A_{tot}(IN_R, IN_S)}{\mid IN_R \mid + \mid IN_S \mid} + \frac{2 \cdot A_{tot}(OUT_R, OUT_S)}{\mid OUT_R \mid + \mid OUT_S \mid} \in [0, 2] \quad (2)$$

where $A_{tot}(IN_R, IN_S)$ (respectively, $A_{tot}(OUT_R, OUT_S)$) denotes the total value of affinity between the pairs of input (respectively, output) parameters in $\mathcal{R}$ and $\mathcal{S}$, and $\mid \quad \mid$ denotes the set cardinality. A_{tot} is obtained by summing up the values of Name Affinity coefficients for all the pairs of input/output parameters that have Name Affinity in the Domain Ontology. Furthermore, we require that each parameter name participates at most in one pair for the A_{tot} evaluation. $ESim(\mathcal{R}, \mathcal{S})$ assumes value 0 when no pairs of I/O parameters with Name Affinity are found, one from $\mathcal{R}$ and one from $\mathcal{S}$, while it is 2 when $\mathcal{R}$ and $\mathcal{S}$ have the same input and output parameter names.

Functionality-based similarity coefficient. Given two services $\mathcal{R}$ and $\mathcal{S}$, we consider each pair of operations OP_{iR} and OP_{jS}, one from $\mathcal{R}$ and one from $\mathcal{S}$. We denote with $OP_{iR}.IN$ and $OP_{jS}.IN$ (resp., $OP_{iR}.OUT$ and $OP_{jS}.OUT$) the sets of input parameters (resp., output parameters) of OP_{iR} and OP_{jS}. The *Operation similarity coefficient* between OP_{iR} and OP_{jS} ($OpSim(OP_{iR}, OP_{jS})$) is computed as follows

$$NA(OP_{iR}, OP_{jS}) + \frac{2 \cdot A_{tot}(OP_{iR}.IN, OP_{jS}.IN)}{\mid OP_{iR}.IN \mid + \mid OP_{jS}.IN \mid} + \frac{2 \cdot A_{tot}(OP_{iR}.OUT, OP_{jS}.OUT)}{\mid OP_{iR}.OUT \mid + \mid OP_{jS}.OUT \mid} \quad (3)$$

where $NA(OP_{iR}, OP_{jS})$ evaluates the Name Affinity between operation names. We note that $OpSim(OP_{iR}, OP_{jS}) \in [0, 3]$, since it is the sum of three elements

in the range [0,1]. It is 0 when there is no affinity between operation names and I/O parameter names of the two operations, while it is 3 when two operations have the same (or equivalent) name and the same (or equivalent) I/O parameters. We say that two operations OP_{iR} and OP_{jS} are similar, denoted by $OP_{iR} \sim OP_{jS}$, if the following conditions hold: (i) $OpSim(OP_{iR}, OP_{jS}) \geq \gamma$, where $\gamma > 0$ is a similarity threshold set by the domain expert; (ii) each of the three terms on the right hand side of (3) is greater than 0.

The *Functionality-based similarity coefficient* between $\mathcal{R}$ and $\mathcal{S}$, denoted by $FSim(\mathcal{R}, \mathcal{S})$, is the measure of similarity of their operations, computed as follows

$$FSim(\mathcal{R}, \mathcal{S}) = \frac{2 \cdot \sum_{h,k} OpSim(OP_{hR}, OP_{kS})}{\mid OP_R \mid + \mid OP_S \mid} \tag{4}$$

where $OP_{hR} \sim OP_{kS}$ holds and $\mid OP_R \mid$ and $\mid OP_S \mid$ denote the number of operations of $\mathcal{R}$ and $\mathcal{S}$. Note that $FSim(\mathcal{R}, \mathcal{S}) \in [0, 3]$, since each term $OpSim(OP_{hR}, OP_{kS}) \in [0, 3]$.

Finally, the *Global similarity coefficient* between $\mathcal{R}$ and $\mathcal{S}$ ($GSim(\mathcal{R}, \mathcal{S})$) is the measure of their level of overall similarity computed as the weighted sum of the Entity-based and Functionality-based similarity coefficients as follows:

$$GSim(\mathcal{R}, \mathcal{S}) = w_1 \cdot NormESim(\mathcal{R}, \mathcal{S}) + w_2 \cdot NormFSim(\mathcal{R}, \mathcal{S}) \tag{5}$$

where $GSim() \in [0, 1]$ since $NormESim()$ and $NormFSim()$ are respectively the values of $ESim()$ and $FSim()$ normalized to the range $[0, 1]$; 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 $GSim()$ coefficient. The use of weights in $GSim(\mathcal{R}, \mathcal{S})$ is motivated by the need of flexible comparison strategies, since some elements in service description could be considered more important for the similarity evaluation and must be weighted accordingly. For instance, to state that the Entity-based similarity and Functionality-based similarity have the same relevance, we choose $w_1 = w_2 = 0.5$.

Example. If we compare the two previous service descriptions by means of the similarity approach, we obtain the following values for the similarity coefficients

$$ESim(\mathcal{R}, \mathcal{S}) = \frac{2 \cdot (1 + 0 + 0 + 0)}{2 + 4} + \frac{2 \cdot 1}{1 + 1} = 1.133 \tag{6}$$

$$OpSim(OP_{1R}, OP_{1S}) = 1 + \frac{2 \cdot 1}{2 + 4} + \frac{2 \cdot 1}{1 + 1} = 2.133 = FSim(\mathcal{R}, \mathcal{S}) \tag{7}$$

$$GSim(\mathcal{R}, \mathcal{S}) = \frac{1}{2} \cdot \frac{1.133}{2} + \frac{1}{2} \cdot \frac{2.133}{3} = 0.639 \tag{8}$$

Note that in our example we have only one operation, so $FSim()$= $OpSim()$.

4.3 Hybrid matching

We use both deductive and similarity-based approach to enhance precision and flexibility of the matching process. Firstly, inference is used to classify the match

between the request $\mathcal{R}$ and available services $\mathcal{S}$ into one of the five kinds proposed in Section 4.1. Successively, similarity evaluation can be exploited to further refine and quantify the functional similarity between $\mathcal{R}$ and $\mathcal{S}$, 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 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 are proposed among the searching results, ranked with respect to the $GSim()$ values.

The application of this hybrid approach ensures flexibility, since not only *exact* or *plug-in match* are considered (this would seem quite unrealistic), but also partial matches are taken into consideration by evaluating the similarity degree. Moreover, 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 are quite similar from the functional viewpoint with the request $\mathcal{R}$. In the example presented above, the `FlightBookingEuropeToUSA` service is classified in the *intersection match*, but it is not excluded from the searching results thanks to similarity evaluation, that reveals similarity between the service and the request.

5 Service Ontology architecture

In our approach a Service Ontology is deployed to organize services in three layers of increasing abstraction by means of semantic relationships. We distinguish between *concrete services*, *abstract services* and *subject categories*.

Concrete services are directly invocable services, described by means of their public WSDL interface; they are stored in the UDDI Registry, where Web links to their concrete implementation on the net are available. Concrete services are described as explained in Section 3. Their descriptions are compared by applying similarity coefficients proposed in Section 4.2 and are clustered on the basis of their global similarity value. Concrete services constitute the *Concrete layer* of the Service Ontology.

Each cluster of similar concrete services is associated to an abstract service in the middle layer of the ontology. Abstract services are not directly invocable services and represent the functionalities of similar concrete services. They are also described by means of a functional interface, with input/output parameters and operations, as explained in Section 3. Interface description of an abstract service is obtained by means of an integration process that identifies correspondences among similar operations of concrete services in the same cluster (such correspondences are recognized both between operation and I/O parameter names) and represents the corresponding operations in the abstract service; mapping rules are maintained between the operation and I/O parameter names of the

abstract functionalities and the original concrete ones. Abstract services are semantically organized according to two kinds of semantic relationships:

- an abstract service $\mathcal{S}_\alpha$ is a *generalization* of another abstract service $\mathcal{S}_\beta$ if, informally stated, $\mathcal{S}_\beta$ provides at least the operations of $\mathcal{S}_\alpha$, as established by properties of plug-in match discussed in Section 4.1;
- an abstract service $\mathcal{S}_\alpha$ *isComposedOf* a set $\Phi = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ of other abstract services if the operations of $\mathcal{S}_\alpha$ are included in the union set of operations of $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$; in particular, Φ must be minimal (no redundancy); the service $\mathcal{S}_\alpha$ is often called the *composite service*, while $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ are called the *component services*.

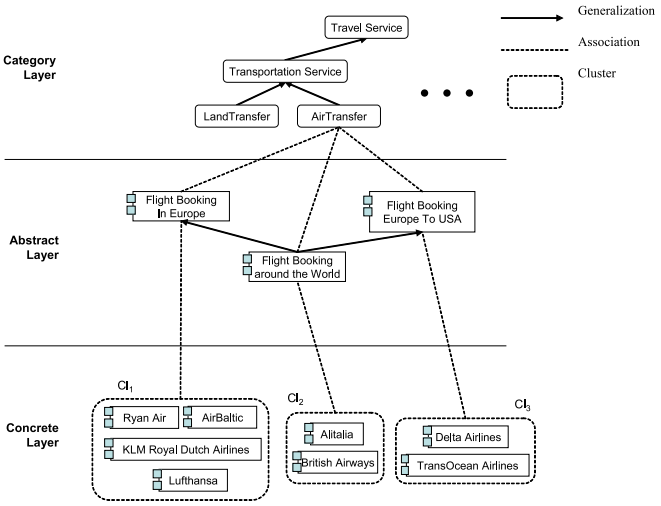

Fig. 1. A portion of three-layer service ontology.

Semantic relationships between abstract services are set by the ontology designer possibly supported by an automatic system that applies Description Logic matching techniques explained in Section 4.1 to identify candidate abstract services between which semantic relationships could be set. This approach is not new: also OWL-S specification admits profile hierarchies by sub-classing the profile model introducing new properties or placing restrictions on existing properties [13]. In a similar way, abstract services are sub-classed by introducing new operations or making restrictions on operation names and I/O parameters. Abstract services and semantic relationships between them constitute the *Abstract layer* of the Service Ontology.

Finally, subject categories correspond to categories of services organized into standard available taxonomies (such as UNSPSC or NAICS) and provide a topic-driven access to the underlying abstract services. These categories are the same that are used in the UDDI Registry to classify published concrete services; in

our Service Ontology, each abstract service is associated to the set of all the categories related to the concrete services belonging to the corresponding cluster. Subject categories constitute the *Category Layer* of the ontology. Figure 1 is an example of the portion of a Service Ontology for the touristic domain considered in this work.

5.1 Service Ontology deployment for service retrieval

Semantic relationships between concrete services, abstract services and subject categories in the Service Ontology can be exploited to make more efficient the service discovery procedure. Matching algorithm exposed in the previous sections is applied at the category and abstract layers and can be speed up by considering the semantic relationships between abstract services, 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 the same capabilities of $\mathcal{S}_a$ (as expressed by means of semantic relationships) match with $\mathcal{R}$. According to this intuition, the following rules are applied:

- if $\mathcal{S}_{ai}$ presents an *exact* or a *plug-in match* with $\mathcal{R}$ and $\mathcal{S}_{ai}$ is a generalization of another abstract service $\mathcal{S}_{aj}$, then also $\mathcal{S}_{aj}$ presents a *plug-in match* with $\mathcal{R}$ and the application of matching algorithm to $\mathcal{S}_{aj}$ is not required; we can set $GSim(\mathcal{R}, \mathcal{S}_{ai}) = GSim(\mathcal{R}, \mathcal{S}_{aj}) = 1$;
- if $\mathcal{S}_{ai}$ presents a *mismatch* with $\mathcal{R}$ and another abstract service $\mathcal{S}_{aj}$ is a generalization of $\mathcal{S}_{ai}$, then we can say that also $\mathcal{S}_{aj}$ presents a *mismatch* with $\mathcal{R}$;
- the same procedure applies when $\mathcal{S}_{ai}$ is a composite service and $\mathcal{S}_{aj}$ is the union of its component ones;
- otherwise, we cannot say anything about $GSim(\mathcal{R}, \mathcal{S}_{aj})$ and the matching procedure must be applied also to it.

Finally, once abstract services that match with the request are extracted from the Service Ontology, then concrete services belonging to the corresponding clusters are included into the searching results, by setting the $GSim()$ value of each concrete service w.r.t. $\mathcal{R}$ equal to the $GSim()$ value of the corresponding abstract service.

6 Conclusions and future work

In this paper we have presented a novel service discovery approach that combines a full deductive matching based on Description Logics and a similarity-based matching, also exploiting semantic organization of services in a Service Ontology to speed up semantic discovery. The combination of two types of matching enhances flexibility and effectiveness of the matching process, while use of Service Ontology improves efficiency. The proposed approach is being experimented in the touristic domain to discover suitable available services for flight booking, hotel reservation, travel planning. Future efforts will investigate further matching modalities also according to levels of granularity in a wide usage scenario.

References

1. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. B. Benatallah, M-S. Hacid, C. Rey, and F. Toumani. Request Rewriting-based Web Service Dicovery. In *Proc. of the Int. Semantic Web Conference (ISWC 2003)*, pages 242–257, Sanibel Island, FL, USA, October 2003.
3. A. Bernstein and M. Klein. Towards high-precision service retrieval. *IEEE Internet Computing*, 8(1):30–36, 2004.
4. D. Bianchini, V. De Antonellis, and M. Melchiori. An ontology-based method for classifying and searching *e*-Services. In *Proc. Forum of First Int. Conf. on Service Oriented Computing (ICSOC 2003)*, Trento, Italy, December 15-18 2003.
5. F. Casati, M-C. Shan, and D. Georakopoulos. The VLDB Journal: Special Issue on E-Services. *Springer-Verlag Berlin Heidelberg*, 10(1), 2001.
6. S. Castano and V. De Antonellis. A Framework for expressing Semantic Relationships between Multiple Information Systems for Cooperation. *Information Systems*, 123(3-4):253–277, 1998.
7. The OWL Service Coalition. *OWL-S 1.1 beta release*, July 2004. `http://www.daml.org/services/owl-s/1.1B`.
8. M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language W3C Recommendation*, February 2004. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.
9. 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)*, Toronto, Canada, August 29th - September 3rd 2004.
10. D. Fensel, C. Bussler, Y. Ding, and B. Omelayenko. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
11. J. Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of the KI-2001 Workshop on Applications of Description Logics*, Vienna, Austria, September 2001. `http://sunsite.informatik.rwthaachen.de/Publications/CEUR-WS/Vol-44/`.
12. I. Horrocks and L. Li. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 331–339, 2003.
13. R. Hull and J. Su. Tools for Design of Composite Web Services. In *SIGMOD Conference*, pages 958–961, Paris, France, 2004.
14. The INTEROP NoE Portal. `http://www.interop-noe.org/`.
15. T. Kawamura, J.-A. D. Blasio, T. Hasegawa, M. Paolucci, and K. Sycara. Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry. In *Proc. of First Int. Conf. on Service Oriented Computing (ICSOC 2003)*, pages 208–224, Trento, Italy, 2003.
16. 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/`.
17. The MAIS Project Home Page. `http://www.mais-project.it`.
18. M. Paolucci, T. Payne, and K. Sycara. Advertising and Matching DAML-S Service Descriptions (position paper). In *International Semantic Web Working Symposium*, Stanford University, California, USA, July 2001.
19. A. Patil, S. Oundhakar, A. Sheth, and K. Verma. METEOR-S Web Service Annotation Framework. In *The Thirteenth Int. World Wide Web Conference*, New York, NY, USA, May 2004.
20. The Racer Home Page. `http://www.sts.tu-harburg.de/ r.f.moeller/racer/`.