Available online at www.sciencedirect.com

ScienceDirect

International Journal of Approximate Reasoning
48 (2008) 808–828

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

A hybrid approach to semantic web services matchmaking

Giuseppe Fenza, Vincenzo Loia *, Sabrina Senatore

Dipartimento di Matematica e Informatica, Universitá degli Studi di Salerno, via Ponte Don Melillo 84084, Fisciano, SA, Italy

Received 27 August 2007; received in revised form 21 December 2007; accepted 10 January 2008
Available online 1 February 2008

Abstract

Deploying the semantics embedded in web services is a mandatory step in the automation of discovery, invocation and composition activities. The semantic annotation is the "add-on" to cope with the actual interoperability limitations and to assure a valid support to the interpretation of services capabilities. Nevertheless many issues have to be reached to support semantics in the web services and to guarantee accurate functionality descriptions. Early efforts address automatic matchmaking tasks, in order to find eligible advertised services which appropriately meet the consumer's demand. In the most of approaches, this activity is often entrusted to software agents, able to drive reasoning/planning activities, to discover the required service which can be single or composed of more atomic services.

This paper presents a hybrid framework which achieves a fuzzy matchmaking of semantic web services. Central role is entrusted to task-oriented agents that, given a service request, interact to discover approximate reply, when no exact match occurs among the available web services. The matchmaking activity exploits a mathematical model, the fuzzy multiset to suitably represent the multi-granular information, enclosed into an OWLS-based description of a semantic web service.

Keywords: Semantic web services; Fuzzy multiset; Hybrid system; Fuzzy C-mean (FCM) clustering; Agents; Ontology web language service (OWLS)

1. Introduction

Semantic web stresses the binomial "meaning and content", adding semantics to web resources which become "machine understandable". The traditional content-oriented web view and the emergent innovative semantic web description are two facets of the same coin, whose contribution is to expedite the direct and effective access to web resources and, in particular, to the information. In fact, our previous studies [19] aimed at the reconciliation and collaborative processing of semantics and contents of web information.

The introduction of semantics in the description of web resources reflects new achievements in web services technologies, through extensive specifications, automation of services selection, composition and translation of message content, self-describing service monitoring and recovery from failure [21]. Semantic web services assure more machine-oriented expressive power and usage of services, completely transforming the

* Corresponding author. Tel.: +39 089 963377.
E-mail addresses: gfenza@unisa.it (G. Fenza), loia@unisa.it (V. Loia), ssenatore@unisa.it (S. Senatore).

0888-613X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2008.01.005

web information access from the usual content-based retrieval to semantic annotated functionalities, exposed by the web services.

This work presents a hybrid architecture which attains a synergy between the agent-based paradigm and the fuzzy modeling for the matchmaking of semantic web services. The goal is to improve the mediation activity among service providers through proactive integration for providing automated semantic interoperability. The approach exploits the agent paradigm for achieving integration, matchmaking and brokerage activities. The web services are defined and semantically "deployed" by OWL-S specifications [23], which provide a high level description of the services capabilities. The services matchmaking activity is modeled by a flexible mathematical model, the fuzzy multiset, suitable for representing the multi-granular information framed into the OWL-S description of the advertised web services.

The paper is organized as follows: Section 2 briefly introduces the actual web scenario and the role of the semantics for the description of web service content. A short presentation of OWL-S is given, through an applicative example of OWL-S Profile. Then, the fuzzy multiset model is presented in Section 3. A whole overview of the architecture describes the layer-based components and the complete working flow (in Section 4 and relative subsections), then the theoretic model on the basis of the matchmaking activity are presented in Section 5. Section 6 describes a hypothetical scenario of matchmaking, through a simple example. Experimental results in Section 7 complete and validate the approach. Finally, conclusions close the paper.

2. The scenario of semantic web services

Nowadays web services have attained a leading role, promising interoperability between different applications of heterogeneous environments. Although last trends emphasize the development of service-oriented architecture (SOA), the loosely coupled interaction between components and applications still represents an open question.

The web service technology is often too restricted to syntactical interoperability; thus the interpretation of messages exchanged by services as well as the understanding of the service capabilities totally depend on the knowledge embedded in application programs, often designed for an enterprise *ad hoc* use. The web services Description Language (WSDL) [30] is mainly based on the syntactical specification of the input and output messages of a service. In addition, WSDL does not support the specification of workflows composed of basic services.

This way, the semantic web services promise a new level of interoperability, adding semantic annotations to specific business functionality, in order to facilitate the interpretation and the representation of non-trivial statements (input, output, constraints, etc.). Typically, a web service is based on the description of its contents, i.e. its static information and the changes in the world that it induces. To enable automation of the discovery, manipulation and composition of services, it is necessary to add a layer of semantics to the contents description through reasoning-based approaches and formal specification languages. Bypassing these constraints means to reinforce the traditional web service technologies by exploiting a well-tight semantic expressiveness of the ontologies for the conceptualization of the knowledge and the explicit specification of a domain of discourse.

In the next future, with the maturity of semantic web service technologies, a lot of public and private registries will request and provide semantic web services, but the sharing of knowledge, integrating the semantic web design principles as well as design principles for distributed, service-orientated web computing, will be necessarily based on ontologies [11].

The last trend exploits the "formal semantics" of both service advertisements (i.e. the description which the service provides) and requests (i.e. the description provided by the final user when looks for a certain service) [17]. There are several studies and projects that aim at adding semantics to web service infrastructures. OWL-S [23] provides a qualified OWL-based support for semantic web services advertising and process capabilities; METEOR-S [16] has moved to the same direction, adding an interface to UDDI [28] for concept-based querying. Web Service Modeling Framework (WSMF)[5] constitutes another attempt of defining the ontology specification, aimed at stressing the mapping between different ontologies, in order to solve the interoperability problems between heterogeneous web services.

In the web service matchmaking domain, the description of services and the characterization of searching criteria are the main factors for determining the quality of the output. The similarity between provider's service and service request is often expressed in terms of software signatures, activities and capabilities, syntax

and semantics of services. In [31] a similarity-based approach for web service matchmaking is based on the comparison of signature specifications through WSDL, without exploiting the semantic description. Otherwise, the matchmaking approach proposed in [15] achieves a mapping between UDDI and OWL-S, by using fuzzy logic to extract the data content embedded in the web services; it gets high level of abstraction for representing services capabilities by enabling the formulation of approximate queries (using imprecise terms) and then, the optimization of the discovery process of services. The common leading line is to accomplish hybrid approaches that exploit the synergy of combining different methodologies and technologies. For instance, some frameworks in this area develop semantic web matchmakers based on some decidable description logic ontology language. Logic-based reasoning often helps to automatically discover services that semantically match with a given service request, evaluated on the basis of the relationships among the concepts defined in the participating ontologies. For instance, OWLS-UDDI matchmaker [26] is a logic-based approach to discovery of semantic services: it enhances UDDI with semantic capability matchings, through the implementation of reasoner-based matching engine.

On the other hand, the work in [18] presents an agent-based system for web services discovery. The matchmaking exploits a Description Logic reasoner (Racer) for semantic comparisons of services descriptions (in DAML + OIL ontology language [6]).

In [4] the matchmaking activity is based on an "enrichment" of service request by additional extra information, not provided by the existing services. In particular, the service matching activity is based on the best covering problem: it computes the right subset of web services analyzing the part of services that is semantically common with a given service request and the part that are semantically different from the request.

Matchmaking approaches often achieve not only the lists of matching web services but also provide a ranking of them for category [14], by considering both individual elements of the OWL-S description and the aggregation of them.

2.1. Semantic description of web services

Web services interoperability aims at supporting the interpretation of heterogeneous information, in order to automate the discovering of suitable services in open environment applications. Syntactical limitations of languages, such as WSDL, often hinder the elicitation of service description and semantics. Through a proper "abstraction" of web services, a well-formed, semantic request finds the eligible service among all the available ones.

This work exploits the OWL-S description [23] for describing the web services capabilities. OWL-S is a Web Ontology Language (OWL) for semantic web services which supports the dynamic discovery, invocation, composition and mediation of web services. OWL-S allows the providers to deploy a complete description of the web service's capabilities, through the following three modules (see Fig. 1):

- the *Service Profile* provides a concise representation of web service capabilities (i.e. what the service does), through the advertising of the functionalities description;
- the *Service Model* gives a detailed description of how the service operates, specifically describing the transformations (i.e. the processes) that it undertakes;
- the *Service Grounding* supplies the details on how interoperate with a service, mapping the messages (according to the format and input/output specification provided in the process model) to the syntactic WSDL compliant form.

The OWL-S Profile represents the high level description of the specifications of a semantic web services. It encloses a textual description and contact information, aimed at human interpretation. Moreover, it declares the functional description of an advertised web service, through its own IOPR (*Input–Output-Precondition-Result*) specifications. Indeed, a set of conditions holds in order to guarantee the proper execution of a service (*Precondition*), a set of postconditions is defined too, after the service execution (*Result*), and finally the *Input* and *Output* describes I/O functional descriptions. The OWL-S Profile often maintains an abstract description of the actual specifications, whereas the lower level OWL-S modules generally provides more "functional"

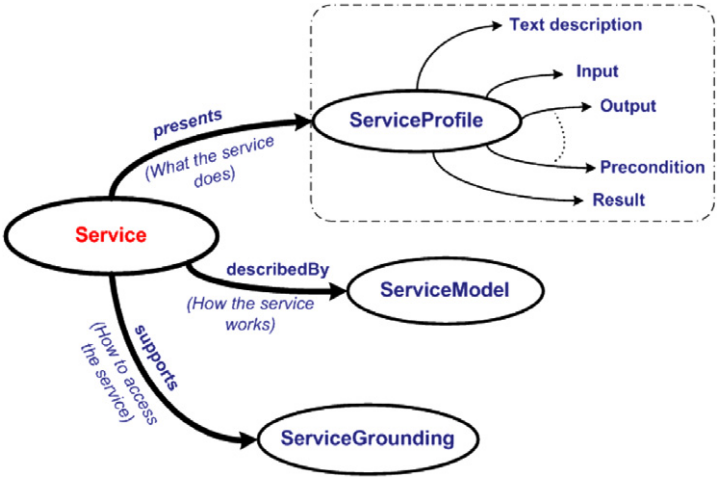

Fig. 1. OWL-S-based description of a web service.

details about the service capabilities. Anyway, the role of the IOPR specifications in the OWL-S Profile is descriptive of the service, additional details to the IOPR information are further explained in the OWL-S Process description layer (whose description is out of the scope of this work).

Our approach focuses on the *Service Profile* representation. In order to provide an example of OWL-S Profile content, let us consider a simple web service description, given in the following sketched code.

```
1.  <rdf:RDF xml:base="http://127.0.0.1/services/1.1/citycountry_destinationhotel_service.owls">
2.      <owl:Ontology rdf:about="">
3.          <owl:imports rdf:resource="http://127.0.0.1/ontology/Profile.owl"/>
            ...
4.          <owl:imports rdf:resource= "http://127.0.0.1/ontology/travel.owl"/>
5.      </owl:Ontology>
6.      <service:Service rdf:ID= "CITYCOUNTRY_DESTINATIONHOTEL_SERVICE">
7.          <service:presents rdf:resource="#CITYCOUNTRY_DESTINATIONHOTEL_PROFILE"/>
8.          <service:describedBy rdf:resource="#CITYCOUNTRY_DESTINATIONHOTEL_PROC-MOD"/>
9.          <service:supports rdf:resource="#CITYCOUNTRY_DESTINATIONHOTEL_GROUNDING"/>
10.     </service:Service>
11.     <!-- Profile description- - ->
12.     <profile:Profile rdf:ID="CITYCOUNTRY_DESTINATIONHOTEL_PROFILE">
13.         <service:isPresentedBy rdf:resource="#CITYCOUNTRY_DESTINATIONHOTEL_SERVICE"/>
14.         <profile:serviceName xml:lang="en"> CityCountryInfoService </profile:serviceName>
15.         <profile:textDescription xml:lang="en">
16.             This service returns hotel of the capital city of the country and it
17.             returns the destination of the city also. The service checks
18.             if the given country is valid; finally loads on the browser the relative web page.
19.         </profile:textDescription>
20.         <profile:hasPrecondition rdf:resource="#_COUNTRY_CODE_VALIDITY"/>
21.         <profile:hasInput rdf:resource="#_COUNTRY_CODE"/>
22.         <profile:hasInput rdf:resource="#_CITY"/>
23.         <profile:hasOutput rdf:resource="#_DESTINATION"/>
24.         <profile:hasOutput rdf:resource="#_HOTEL"/>
25.         <profile:hasResult rdf:resource="#_CITY_DESTINATION_WEBPAGE_LOADING"/>
26.         <profile:has_process rdf:resource="CITYCOUNTRY_DESTINATIONHOTEL_PROCESS"/>
27.     </profile:Profile>
28.         <!- -Process Model description- - ->
                ...
29.         <!- - Grounding description- - ->
                ...
30.     </rdf:RDF>
```

This OWL-S specifications describe a service which returns all the hotels of a capital city, given the country and the city names. The code line numbers are just added to facilitate our discussion, obviously they are not a part of the code of an OWL-S service specification. Lines 1–5 specify the basis of XML coding and the namespaces declaration of the ontologies exploited by the ontology description of the current service. Lines 6–10 introduce instead, a description of the service, identified as *CITYCOUNTRY_DESTINATIONHOTEL_SERVICE*, showing the relationships among the three OWL-S layers (i.e. the service *presents* a Profile, is *described By* a process model and *supports* a given grounding mapping). Lines 12–27 focus on the OWL-S Profile description: the service profile, named *CityCountryInfoService* (line 14) provides a textual description (lines 15–19), the precondition, the input, the output, the result specifications (from the release OWL-S 1.1 the tag *hasEffect* is deprecated and has been substituted by *hasResult* and finally the process name (lines 20-26). In other words, the service takes as input the city (*_CITY*, line 22) of a country (through its international code *_COUNTRY_CODE*, line 21) and returns the hotel (*_HOTEL*, line 24) and the destination type (*_DESTINATION*, line 23) in the given city. As precondition, the service checks the validity of the input country code (*_COUNTRY_CODE_VALIDITY*, line 20), whereas as result, it loads the web page of the given city *_CITY_DESTINATION_WEBPAGE_ LOADING*, line 25). The variable names *_CITY, _HOTEL, _DESTINATION*, used in this OWL-S description, are then associated to the effective ontological entities, which are appropriately described (or more formally, defined) in the reference ontology. Just to clarify, the next piece of OWL-S Profile provides some details about the description of the output type parameter *_DESTINATION*.

```
<process:Output rdf:ID= "_DESTINATION">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
    http://127.0.0.1/ontology/travel.owl#Destination
  </process:parameterType>
  <rdfs:label>Destination</rdfs:label>
</process:Output>
```

Herein, _DESTINATION identifies the variable name whose associated label is "Destination". It is available at http://127.0.0.1/ontology/travel.owl#Destination, where the actual ontological entity is completely defined in the ontology *travel.owl*. For sake of simplicity, in the following the ontology terms (or concepts) are used as identifiers of variable names, unless of the beginning character underscore "_".

The OWL-S language represents an upper ontology for semantically describing web services. In this approach, the IOPRs specifications of the OWL-S Profile are translated in *fuzzy multiset*. This mathematical model is particularly flexible and fits to describe multi-granule knowledge [20]. Specifically, it provides a suitable formalism to represent the ontology-based concepts of the IOPR functionalities. Each concept (related to some reference, OWL-S-based ontologies) is represented as a multi-valued entity, according to the meaning and the role which it plays in the OWL-S Profile declaration. Herein, the grained information just represents the multi-facets OWL-S description of the web services. The following section details the modeling of semantic web services through of the fuzzy multiset representation. Then, a conclusive example will detail the role and interpretation of the fuzzy multiset.

3. Modeling view

The fuzzy multiset is a mathematical framework that captures multiple occurrences of an element (or subject) with a certain degree of relevance. In general, in a fuzzy multiset, an element may appear more than once with possibly the same or different membership values [22]. More formally,

Definition 1. A fuzzy multiset A of X (often called fuzzy bag [32]) is characterized by the function $Count_A(\bullet)$ that takes the values of a finite multiset in $I = [0, 1]$. Namely, given $x \in X$,

$$Count_A(x) = \{\mu, \mu', \ldots, \mu''\}, \quad \text{where } \mu, \mu', \ldots, \mu'' \in I. \tag{1}$$

Thus, the fuzzy multiset A has the following form:

$$A = \{\{\mu, \mu', \ldots, \mu''\}/x, \ldots\} \quad \text{or} \quad A = \{(x, \mu), (x, \mu'), \ldots, (x, \mu''), \ldots\}. \tag{2}$$

For simplicity, the multisets of $I = [0, 1]$ are considered finite. The collection of all the fuzzy multisets of X is denoted by $\mathscr{FM}(X)$. In order to apply basic operations of fuzzy multisets, the *membership sequence* is defined: it is a decreasingly ordered sequence of $Count_A(x)$, denoted by

$$Count_A(x) = (\mu_A^1(x), \mu_A^2(x), \ldots \mu_A^p(x)) \tag{3}$$

where

$$\mu_A^1(x) \geqslant \mu_A^2(x) \geqslant \cdots \geqslant \mu_A^p(x) \quad \forall j \tag{4}$$

The sorted sequence for $Count_A(x)$ is called the standard form for a fuzzy multiset. In the light of the web services context, a fuzzy multiset is exploited to represent a semantic web service profile: here the basic element of a OWLS Profile is a concept which can appear many time in the OWLS specifications, but playing different roles (as Input, Precondition, etc.). This way, the fuzzy multiset permits us to consider each concepts as multi-granular information, according to its role in the OWL-S documents. Thus, each fuzzy multiset is composed of multiple occurrences (defined by the associated membership values) of each concept. More formally, the fuzzy multiset definition should be as follows.

Definition 2. Let us denote $\mathscr{P} = \{p_1, p_2, \ldots, p_m\}$ a set of web services, described by the relative OWL-S Profile documents. Let $\mathscr{FM}(T)$ be the collection of all fuzzy multiset on the set of concepts (i.e. the ontology terms filtered on the OWL-S specifications) $T = \{t_1, \ldots, t_n\}$. Let $\mathscr{S} : \mathscr{P} \to \mathscr{FM}(T)$ be a function such that $\forall p_i \in P$, with $i = 1, \ldots, m$, $\mathscr{S}(p_i)$ is the fuzzy multiset on T defined as

$$\mathscr{S}(p_i) = \{(\mu_1(t_1), \ldots, \mu_{n_1}(t_1))/t_1, \ldots, (\mu_1(t_n), \ldots, \mu_{n_n}(t_n))/t_n\} \tag{5}$$

where $(\mu_1(t_j), \ldots, \mu_{n_j}(t_j))$ is the membership sequence associated to the concept t_j $(1 \leqslant j \leqslant n)$.

Each membership sequence represents the occurrences of the ontology concept t_j, with the degree of relevance $\mu_h(t_j) \in [0, 1]$ with $h = 1, \ldots, n_j$ in the Profile document p_i.

Given a set of IOPRs functionalities, some heuristics are defined for computing the relevance degree associated to each occurrence of the concept t_j, in the IOPRs tagged elements (further details are given in the following). This way, $\mu_h(t_j)$ is the membership associated to the concept t_j evaluated on the hth IOPR functionality.

Because each web service of the set P is described by an associated OWL-S Profiles, each p_i is the corresponding OWL-S description of each web service.

In this context, the structure of the fuzzy multiset emerges as a suitable model to represent multi-granular information of a web service through the related concepts.

3.1. Example

In order to show the practical use of fuzzy multiset formalism, the OWL-S profile document, given previously in Section 2.1 is considered. Focusing on the IOPRs of the OWL-S profile, let us suppose the following ontology terms are gathered (as said, the ontology terms have the names equal to the considered parameters names, unless of the beginning character "_"):

t_1=COUNTRY_CODE_VALIDITY, t_2=COUNTRY_CODE, t_3=CITY, t_4=DESTINATION, t_5=HOTEL, t_6=CITY_DESTINATION_WEBPAGE_LOADING.

Some simple, heuristic-based rules are built in order to evaluate the ontology terms, according to the role they play in the OWL-S profile.

If an ontology term is in the *Input*, the relevance degree is 1.
If an ontology term is in the *Output*, the relevance degree is 1.
If an ontology term is in the *Precondition*, the relevance degree is 0.8.
If an ontology term is in the *Result*, the relevance degree is 0.5.
If an ontology term is in the *Text Description*, the relevance degree is 0.3.

Assigning a degree to each ontology term provides information about the role and consequently the importance of that term in the profile. For instance, herein, we have assigned maximal relevance (in the range [0, 1])

to Input and Output specifications, in order to give equal importance to them, whereas the other IOPRs assume lower values, because they are considered less meaningful in the service matching.

In our IOPRs description, the ontology term t_1 appears as Precondition, t_2 and t_3 are in the Input, t_4 and t_5 are in Output and, finally t_6 appears as Result. In addition, concepts t_3, t_4 and t_5 occur in the textual description, too. The fuzzy multiset derived from the OWL-S Profile, associated to the given service is as follows:

$$\mathscr{S}(p_i) = \{(0.8, 0.0, 0.0, 0.0, 0.0)/t_1, (1.0, 0.0, 0.0, 0.0, 0.0)/t_2, (1.0, 0.3, 0.0, 0.0, 0.0)/t_3, (1.0, 0.3, 0.0, 0.0, 0.0)/t_4, (1.0, 0.3, 0.0, 0.0, 0.0)/t_5, (0.5, 0.0, 0.0, 0.0, 0.0)/t_6\}$$

In summary, a fuzzy multiset can represent the multi-granular information enclosed into the OWL-S description of a web service; each ontology term t_i is multi-evaluated, according to the role played in the IOPRs. Fig. 2 shows a global view of the considered web service, through the main steps: the analysis of the OWL-S Profile emphasizes the role of participating variables and then, the relative ontology terms are elicited. Simple heuristics support the estimation of the relevance degree of each selected ontology term, according to its own role in the IOPR specifications of the OWL-S profile.

4. Architectural overview

This architecture proposes a hybrid solution to cope with the service matchmaking problem: on the one hand, the web services, equipped with semantic description provide clearer explanation about the data taken into account and, on the other hand, the participating ontological entities assume many and different meanings, with respect to the roles played in the IOPRs.

The architecture, shown in Fig. 3, is based on two distinct layers: the *knowledge* and the *agent* layers. The *knowledge layer* contains all the knowledge processed and maintained by the system. Ontologies and taxonomies stored in the local databases as well as the information processed by the agents or deducted by the fuzzy multisets are the building blocks of this layer. On the other hand, the *agent layer* attends to the process-oriented view of the system. It is built on the agent-based platform Jade [12] and specifically implements two kinds of agents: the *advertiser agent* and the *broker agent*. The agents represent the dynamic components of the system; during their activities and interactions, they enhance their local acquaintance. This acquaintance represents a common shared knowledge which "feeds" the knowledge layer. The whole flow of execution is supervised by the agents. They transparently interact with the external environment (requesters and providers), hiding all the mediation activities.

4.1. Knowledge layer

The main components of the *knowledge layer* are the ontologies exploited by the agents, during their activities. They are downloaded (when are not present locally) and stored in the system databases, then analyzed wherever it is necessary for discovering taxonomic relationships among the concepts in the web service descriptions. The ontology-based knowledge (composed of OWL-S ontologies and the taxonomies) represents the *domain ontologies*. Generally, they describe the concepts and the relative relationships applied on a specific domain. The domain ontologies represent the static side of the system, because once downloaded, it unlikely changes.

On the other hand, the system collects all the information related to the OWL-S specifications of the supervised services. This knowledge instead, is dynamically updated, according to the changing of the web services in an open environment such as Internet. This knowledge is modeled by the fuzzy multisets and shared in the whole system thanks to the agents' action.

4.2. Agents layer

The *agent layer* represents the functional side of the systems: brokerage, matchmaking and discovery are the main activities of the agents. The system includes two types of agents: the *advertiser* and the *broker* agents. They manage the provider-side and requester-side interactions respectively, by filtering and interpreting the

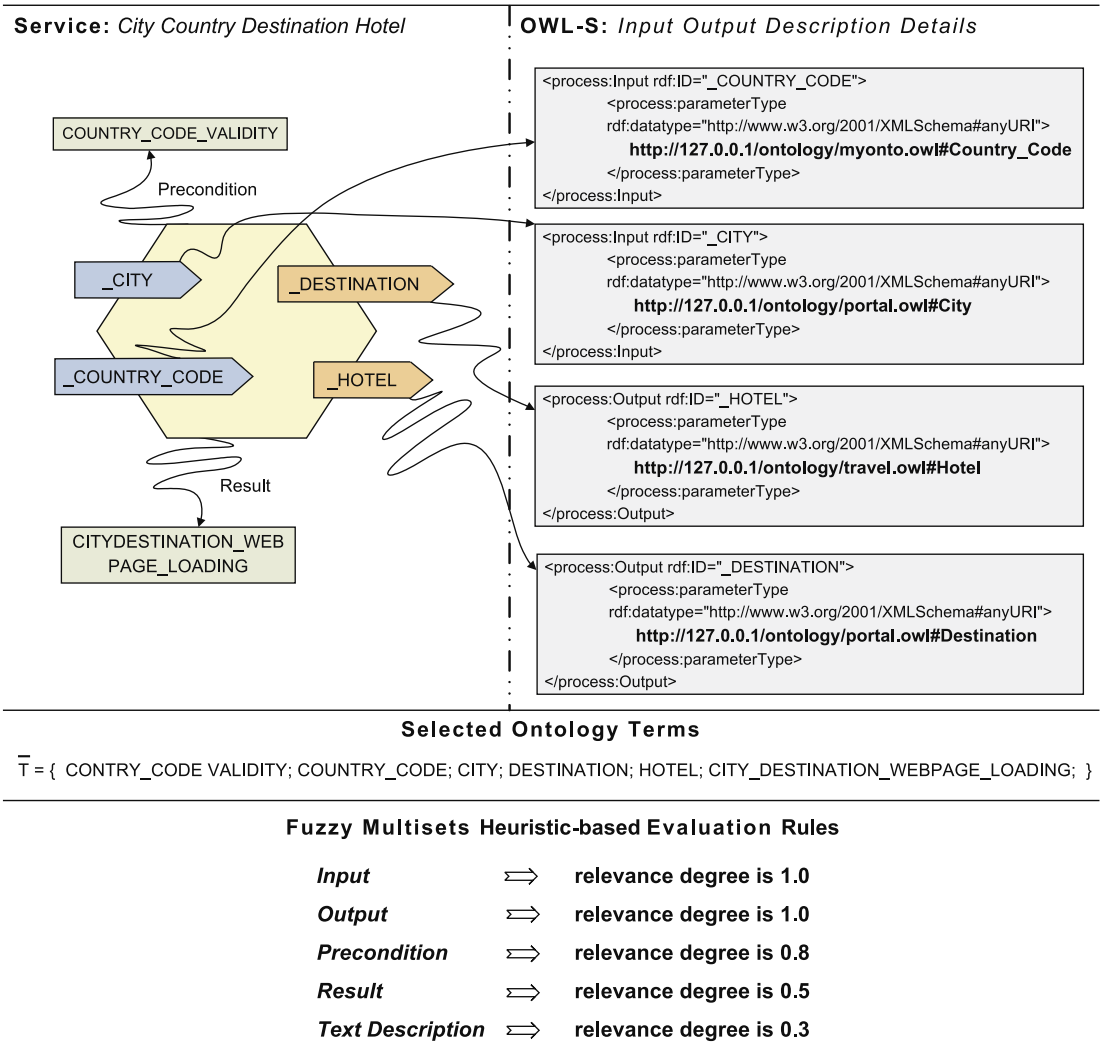

Fig. 2. Fuzzy multiset representation of a semantic web service.

provider's service capabilities and the requester's queries, and natively interact to each others in order to get collaborative tasks.

In details, the two agent typologies are detailed as follows:

– The *advertiser agents* play the role of interface between the web services providers and the *broker agent*. When an *advertiser agent* discovers a web service, it translates the relative OWL-S service specifications into a concept-based representation, by means of the fuzzy multiset model. In fact, the *advertiser agent* analyzes the OWL-S Profile and elicits the ontology terms, used in the ontologies (reached by the namespaces declared in the OWLS file). In order to evaluate the roles of these ontology terms, the agent analyzes

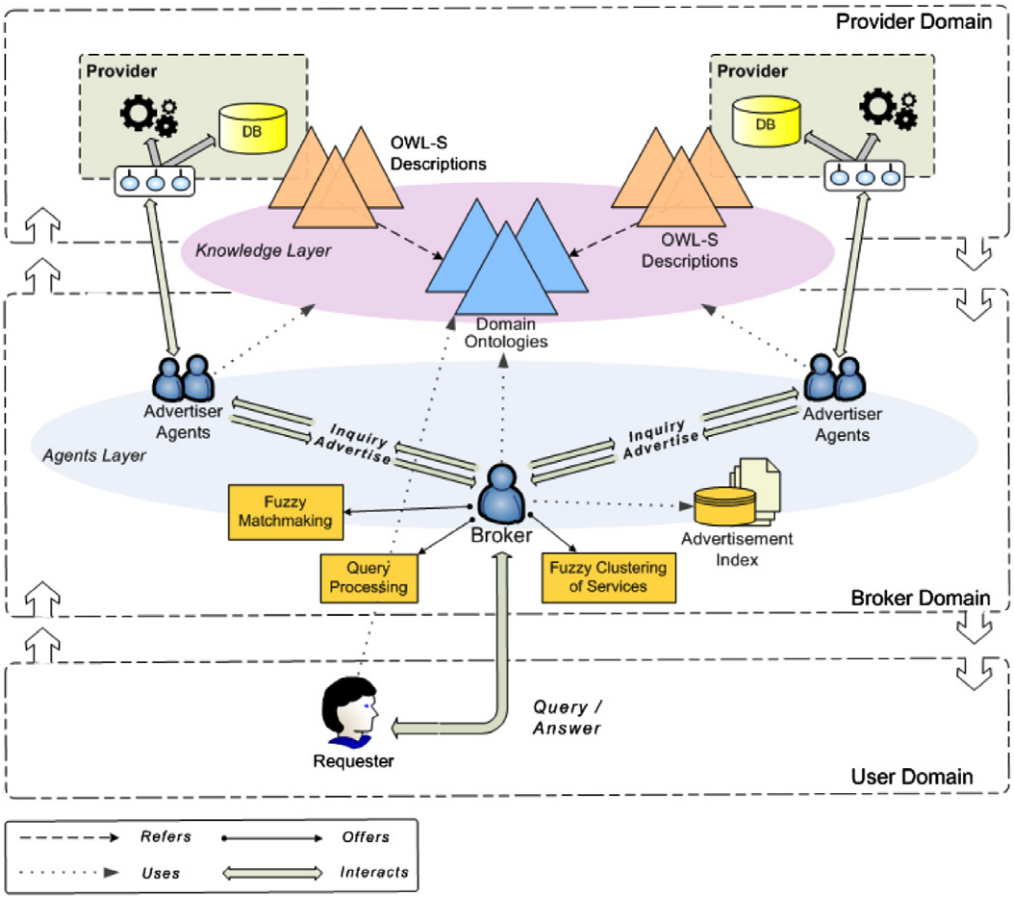

Fig. 3. Multi-tiers architecture of the semantic matchmaking system.

possible (specialization/generalization) relationships with other ontology terms in the domain ontologies. The ensemble of these concepts will be exploited to define the fuzzy multisets, associated to that service. In our approach, the agents refers only the OWLS Profiles (as said, no assumptions have been taken into account on the other OWLS modules).

– The *broker agent* achieves activities of brokerage and matchmaking of semantic web services. A single instance of this agent class exists in the system. The *broker agent* mediates between the service requester and *advertiser agents*. It interprets the requester's query, then interrogates the *advertiser agents* for discovering the web service which better matches with the service request. Moreover, the *broker agent* contributes to augment the knowledge of the system. It acquires the (fuzzy multiset-based) knowledge related to the advertised services, by interaction with the *advertiser agents* and proactively indexes the discovered services. More specifically, it controls a clustering-based knowledge engine, for monitoring a local classification of the discovered services (based on the similarities among the IOPR functionalities). When no discovered service directly matches with service requester, the *broker agent* looks up in the clusters of discovered services, an approximate solution.

4.3. The working flow

The *advertiser agents* and the *broker agent* assure the interaction with the external environment (requesters and providers), hiding all the details of the matchmaking and brokering activities. Initially, the *broker agent* acquires and "interprets" the request, identifying the different query portions (input, output, relative

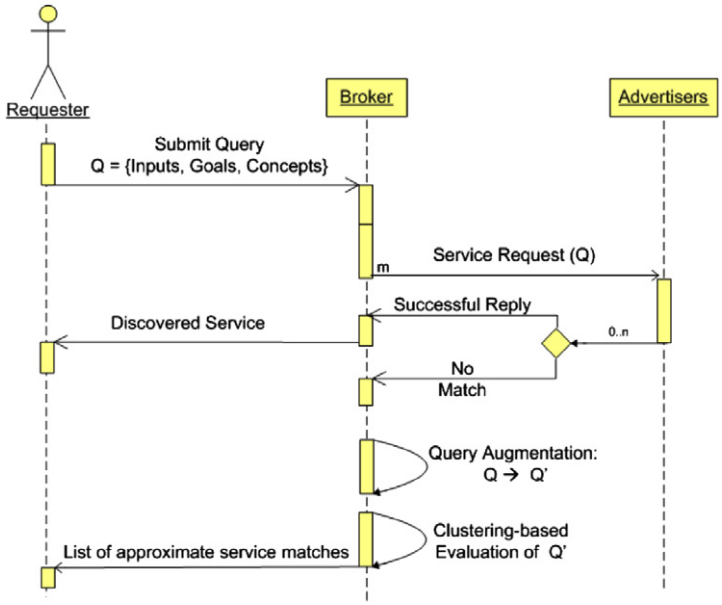

Fig. 4. Broker–advertiser interaction (with requester's query Q).

parameters, etc.). Then, it interacts with the *advertiser agents*, in order to establish if some agents "know" a service whose functionalities match with the arguments of the request. The *broker–advertiser* interaction is based on the FIPA brokerage interaction protocol [9]. Fig. 4 shows a sketched interaction between the broker and an advertiser. The two typologies of agents exchange information messages. The requester starts the communication. The broker interacts with the advertisers in order to discover the appropriate OWLS-based service. If no match is found, the broker tries to provide an approximate reply to the request, exploiting the local knowledge, arranged in a clustering-based collection of services.

The *advertiser agents* maintain a fuzzy multiset-based representation of the discovered services (i.e. "know" the service capabilities), acquired by previous interaction with the service providers. This activity is accomplished periodically, when the *advertiser agents* notice a web service is published/updated. In this approach we stress the situation, when no direct match occurs; otherwise the broker can return the required service capabilities [7]. In the literature [15,27,13], one of the most problems is the lack of matching between the concepts used in the given request (that is, the terminology naming, exploited to defined parameters and functions) and the functionalities described in the provided service.

The *broker agent* translates each requester's query in concept-based representation, through fuzzy multiset too, in order to maintain a consistent modeling. In fact, the request can be seen as an effective "web service" where the input, output, etc. are claimed. So, when no direct match occurs, the *broker agent* tries an approximate reply, by comparing the representation of the requested service with the classified ones, maintained as the local knowledge. Homogeneous comparisons between the clustered services and the requester's query are so admitted. The *broker agent* chooses the services (one or more) which are closer to the input query. Specifically, it compares the fuzzy multiset associated to the request with the fuzzy multisets corresponding to the centers of the clusters. The services in the cluster whose center has minimal distance from the service request are candidate to be the approximate reply.

5. A closer look at the broker agent's activities

The matchmaking activity is aimed at getting one or more web services that provide the right or approximate reply to the input request. Periodically, the *broker agent* organizes the local knowledge engine in order to create appropriate clusters of "similar" services. By interacting with the *advertiser agents*, it gets the fuzzy multiset representation of the supervised services. Then it starts up the clustering process, which produces some

grouping (or, in other words, a classification) of the collected web services. Thanks to the *broker agent* interaction, this clustered knowledge is periodically elaborated, updated and maintained in an index of the reference advertisements. The *broker agent* re-starts up the clustering activity, when some actions are triggered: (1) most discovered services disappear or change in the web; (2) new services are discovered and "registered" on the system platform; (3) appearance or disappearance of services can modify the set of concepts, elicited to describe the web services; and (4) the services classification is restricted to a specific context: only a proper selection of common key-concepts is used in the clustering execution.

Further task of the *broker agent* is to supervise this selection of concepts, called *features* set, in order to guarantee a good clustering and consequently a good reply (that is, the returned list of services) when approximate results are required. The following subsections detail the selection of the features set, the building of corresponding data-matrix and its exploitation in the clustering process. All these activities are supervised by the *broker agent*.

5.1. The features selection

The features set is composed of all the "characteristics" of the given collection of OWLS documents, that are representative for the analyzed web services.

In other words, the "complete" feature set is composed of all the distinct ontology terms (concepts or properties identified by the associated URIs) $\overline{T} = \{t_1, t_2, ..t_n\}$ that are related (via variable names) to the IOPRs of each semantic web service. In the following, let us denote $var(t_i)$ the variable name (i.e. the parameter) associated to the ontology term t_i.

Generally, the features set is composed of the following subsets (compare with in Section 2.1,.

- *Preconditions*: the set of all the ontology terms t_i which participate as precondition (whose language specification assume a form such as $\langle$*profile:Precondition rdf:resource=*$var(t_i)\rangle$) in the OWLS Profile. For instance, considering the OWL-S Profile in Section 2.1, $var(t_i)$=COUNTRY_ CODE_VALIDITY.
- *Outputs*: all the ontology terms t_i which are related to the output description of the OWLS Profile: $\langle$*profile:hasOutput rdf:resource="*$var(t_i)$"$\rangle$.
- *Inputs* (t_i): all the ontology terms t_i which play the role of input in the OWLS Profile of a web service: $\langle$*profile:hasInput rdf:resource="*$var(t_i)$"$\rangle$.
- *Text*(t_i): all the words in the textual description of a service, that can be "translated" into ontology terms: $\langle$*profile:textDescription xml:lang="en*"$\rangle$ $t_i, \ldots, t_j$ $\langle$*/profile:textDescription*$\rangle$, where the ontology terms $t_i, \ldots, t_j$ (with $i, j \in [1, n]$ and $i \leqslant j$) come from the parsing of the included text and represent the approximate interpretation of the concepts words as ontology terms, when no right correspondence exists.
- *Results*(t_i): all the ontology terms t_i in the result description (whose language specification assumes a form such as $\langle$*profile:hasResult rdf:resource="*$var(t_i)$"$\rangle$) of the OWLS Profile.

Note an ontology term can play more roles in the same service; the same way, the same ontology term can be exploited in different services.

Because the whole feature set is too big, a reduced set of features is often computed. In this case, a filtering of the relevant ontology concepts or properties is applied, taking into account some factors:

- The relationships between terms emphasize the generalization/specialization structure of ontologies. In literature, most studies aim at analyzing the taxonomic relationship in/among ontologies and then evaluating the consistency of them [10,25]. In this approach, only concepts hierarchical relationships are studied. In fact, parent–son relationship are analyzed then collapsed and opportunely replaced by just one concept that can represent the most general or a most specialized concept of them. This evaluation is taken out by the analysis of the OWLS profile documents, considering their relevance of the involved concepts.
- The choice of a concept is related to the occurrences of the associated ontology term, too. A local evaluation between two linked ontological terms allows the elicitation of the most representative concept as candidate of the feature set. Moreover, as said in the previous point, the initial features set is reduced exploiting some criteria of subsumption, existing among the most recurring ontological concepts.

In summary, the final feature set is composed of all the terms that cover of all the web services at "descriptive" level.

5.2. The building of data matrix

The data matrix constitutes the input of the clustering algorithm: each row is a fuzzy multiset that represents the correspondent web service description. Each multiset is a vector, whose elements are the sequence of values (in the range [0, 1]), associated to each ontology term selected. The columns of the data matrix instead, represent the elements of features set. Thus, the data matrix has m rows, where m is the number of participating web services and n columns, where n is the cardinality of the feature set.

In Section 3, some simple heuristics associate multi-values to each ontology term, and then, build the fuzzy multiset for each service (description). Although these criteria have been reasonably exploited in our previous work [8], herein, more attention is given in the analysis of the hierarchical relationships between ontological terms. We evaluate the nature of the relationship existing between an ontological term (occurring in an OWLS Profile of a service) and an element of features set. More specifically, let us suppose f' is an ontology term that assumes a specific tagged IOPR role in the given OWLS profile; for each ontology term f', a taxonomic comparison $compare(f, f') \in [0, 1]$ with each element $f \in \overline{T}$ of the features set is applied. Thus, supposing the given role-based heuristics in the example of Section 3.1 hold, the following situations can arise:

- the terms f and f' are equal: $f' = f \Rightarrow compare(f, f') = degree(f)$
- f' is a more specific concept than f: $f' \subset f \Rightarrow compare(f, f') = degree(f)/2$
- f' is a more general concept than f: $f' \supset f \Rightarrow compare(f, f') = 1/degree(f)$

For instance, let us reconsider the Example 3.1, and let us suppose that $f' = CITY$. This term is an *input* type (and it is in the feature set) and, according to the heuristics defined in Example 3.1, $degree(CITY) = 1$. Now let us suppose that the features set $\overline{T}$ contains the ontology term $f = TOWN$ and in some reference ontologies, a relationship of subsumption exists ($f' \subset f$ i.e. f is a more general concept than f'); so compare(f, f') = 0.5. For terms that play the role of Precondition, Result and textual description, similar rules can be applied. Note that, in order to hold the controvariance criteria of input and output types [13], this condition has to be applied for input and output in inversely way. In other words, if the previous situation holds for an input type term, in the case of term f' of *output type*, the rules can be simply applied as follows:

- the terms f and f' are equal: $f' = f \Rightarrow compare(f, f') = degree(f)$
- f' is a more specific concept than f: $f' \subset f \Rightarrow compare(f, f') = 1/degree(f)$
- f' is a more general concept than f: $f' \supset f \Rightarrow compare(f, f') = degree(f)/2$

Our experimentation is based on the analysis of only input and output types too, holding similar assumptions.

5.3. The clustering

Our approach exploits the well-known Fuzzy C-Means (FCM) algorithm [3]. As said, it takes as input a collection of multisets, in a matrix form. Let us recall the fuzzy multisets represent the semantic description of the web services, by means of all the elicited ontology terms and the role played in the OWLS Profile; thus we exploit on a modified version of FCM algorithm, which takes as input a multiset-valued data matrix [22].

The FCM clustering is an unsupervised process, aimed at minimizing an objective function. The result produces a fuzzy partitioning of the data-matrix.

Definition 3. Let $\mathscr{P} = \{p_1, \ldots, p_m\}$ be a set of web service, described by the specifications of the OWLS Profile documents. Fixed a number K of clusters, the exploited objective function is the following [3,22]:

$$J(M, V) = \sum_{i=1}^{K} \sum_{j=1}^{L} u_{i,j}^2 d(C_i, S(p_j)) \tag{6}$$

where $M = (u_{i,j})$ is a $K \times L$ matrix of cluster memberships satisfying the constraint:

$$M = \left\{ u_{i,j} : \sum_{i=1}^{K} u_{i,j} = 1; u_{i,j} \geqslant 0 \ \forall i, j \right\}, \tag{7}$$

and $V = (C_1, \ldots, C_K)$ is the ordered collections of cluster centers.

The value $d(C_i, S(p_j))$ is the distance measure [22], where the arguments C_i and $S(p_j)$ are two fuzzy multisets.

In general, the distance measure between two fuzzy multisets $A, B \in \mathscr{FM}(T)$ with $A = \{\langle \mu_A^1(x_1), \ldots, \mu_A^{n_{x_1}}(x_1)\rangle / x_1, \ldots\}$ and $B = \{\langle \mu_B^1(x_1), \ldots, \mu_B^{n_{x_1}}(x_1)\rangle / x_1, \ldots\}$ is defined as follows:

$$d(A, B) = \sum_{x \in T} \sum_{j=1}^{L(x)} \left| \mu_A^j(x) - \mu_B^j(x) \right| \tag{8}$$

where [22]

$$L(X) = \max\{L(x : A), L(x : B)\} \tag{9}$$

$$L(x : A) = \max\{j : \mu_A^j(x) \neq 0 \ j = 1 \ldots, n_x\} \tag{10}$$

$L(x : A)$ is the length of the membership sequence $< \mu_A^1(x), \mu_A^2(x), \ldots, \mu_A^{n_x}(x) >$ related to $x \in T$. Let us observe that d satisfies the axioms of metrics [22].

6. Query–answering activity: a sketched scenario

This section gives a global description of the query/answering activities, starting from the query submission to the services replies, by emphasizing the main steps related to the fuzzy matchmaking.

In order to give an idea about the usage application scenario, let us suppose the requester would like to get information about all the accommodation closer to the beach, in a specified city (see Fig. 5). The query Q could have the following form:

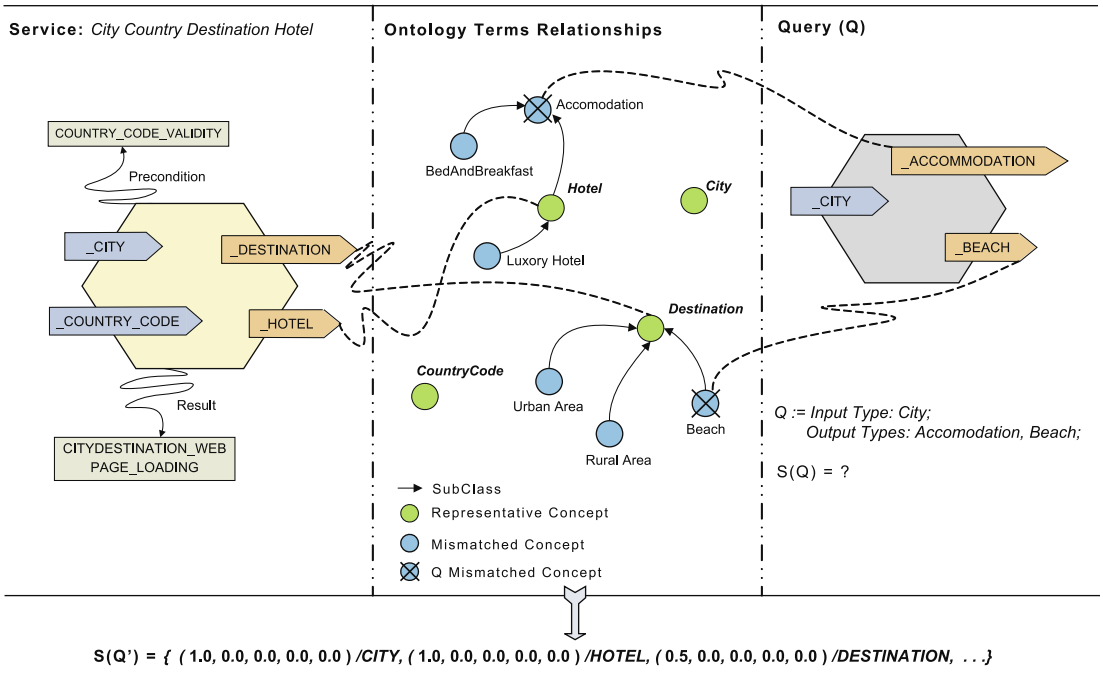

Fig. 5. Approximate fuzzy multiset-based representation of input query.

Q:=INPUT: <City>=CityName; OUTPUT: <Accomodation>=*, <Beach>=*

where the input is a *City* whose name is given (*CityName*), whereas the outputs are unspecified (herein, we use the symbol *) and are relative to the concepts *Accommodation* and *Beach*.

6.1. Query modeling

When the broker acquires this query, sends a message to all the advertiser agents, in order to discover if there is an OWLS service description which meets the service request.

Let us suppose that the query does not directly meet an available service, so no direct match arises. Fig. 5 shows the main activities involved in this phase.

The broker starts up the discovery task for finding an approximate reply. Then, it takes into account the ontology terms of the *features* set. For simplicity, let us assume the features set is composed of the concepts considered in the example in Section 3.1 (i.e. t_1=COUNTRY_CODE_VALIDITY, t_2=COUNTRY_CODE, t_3=CITY, t_4=DESTINATION, t_5=HOTEL, t_6=CITY_DESTINATION_WEB PAGE_LOADING). The ontology term *City* is in the *features* set, but *Accommodation* and *Beach* are not present. At this point, the broker looks for some ontology terms closer to *Accommodation* and *Beach*, in order to adapt the service query to the available web service. It looks up in the reference ontologies, possible relationships with the given concepts.

By the analysis of the ontology *travel.owl*, let us suppose the broker finds the wanted relationships. Fig. 5 shows the discovered relationships between the terms in the domain ontology, with respect to the given service request Q (on the right of the figure). More specifically, the ontological relationships show the term ontology *Accommodation* appears as a direct superclass of the term *HOTEL*, whereas the term *Beach* is a subclass of *DESTINATION*. The general criteria the broker uses is to find the ontology terms of the reference ontologies that are closer to the mismatched concepts in the query. Recall these concepts must belong to the *features* set. In this case, both the ontology terms belong to the complete feature set, thus, they are proper candidates for re-adapting the query Q. According to the function *compare* (f, f'), given in Section 5.2, *HOTEL* and *DESTINATION* are both *output type* ontology terms (see example in Section 3.1), thus the following values are computed:

$$\begin{aligned} compare(HOTEL,\ Accommodation) &= 1/2 = 0.5 \\ &\quad (\text{because } Accommodation \supset HOTEL) \\ compare(DESTINATION, Beach) &= 1/1 = 1 \\ &\quad (\text{because } Beach \subset DESTINATION) \end{aligned}$$

Finally, the broker re-designs the fuzzy multiset relative to the query, exploiting only terms of the feature set, as follows:

$$\begin{aligned} \mathscr{S}(Q') = \{&(1.0, 0.0, 0.0, 0.0, 0.0, 0.0)/CITY, (1.0, 0.0, 0.0, 0.0, 0.0, 0.0)/HOTEL, \\ &(0.5, 0.0, 0.0, 0.0, 0.0, 0.0)/DESTINATION, \ldots\} \end{aligned}$$

Although *COUNTRY_CODE_VALIDITY*, *COUNTRY_CODE* and *DESTINATION_WEB-PAGE_LOADING* are in the *features* set, their membership sequences are empty with respect to the given query, thus they do not appear. The "adapted" query Q' represents a service compliant to the local system knowledge, that is an approximation of the original query the requester looks for.

6.2. Fuzzy matchmaking

The broker evaluates the new query with respect to the clustered web services, stored in its local acquaintance; specifically it compares the fuzzy multiset, associated to the query with the centers of the clusters (i.e. the relative fuzzy multisets), through a distance measure (as defined in Section 5.3). The closest cluster center determines the reference cluster. The broker captures those services whose distance from the query service is not higher than a prefixed threshold (evaluated with respect to the reference cluster). Fig. 6 shows the "action scope" of the query Q', that is the area in which this distance is evaluated. Specifically, the figure presents a sketched clustering of the collected web services (i.e. the collected OWLS Profiles); on the right side, a zoom on the clustering emphasizes

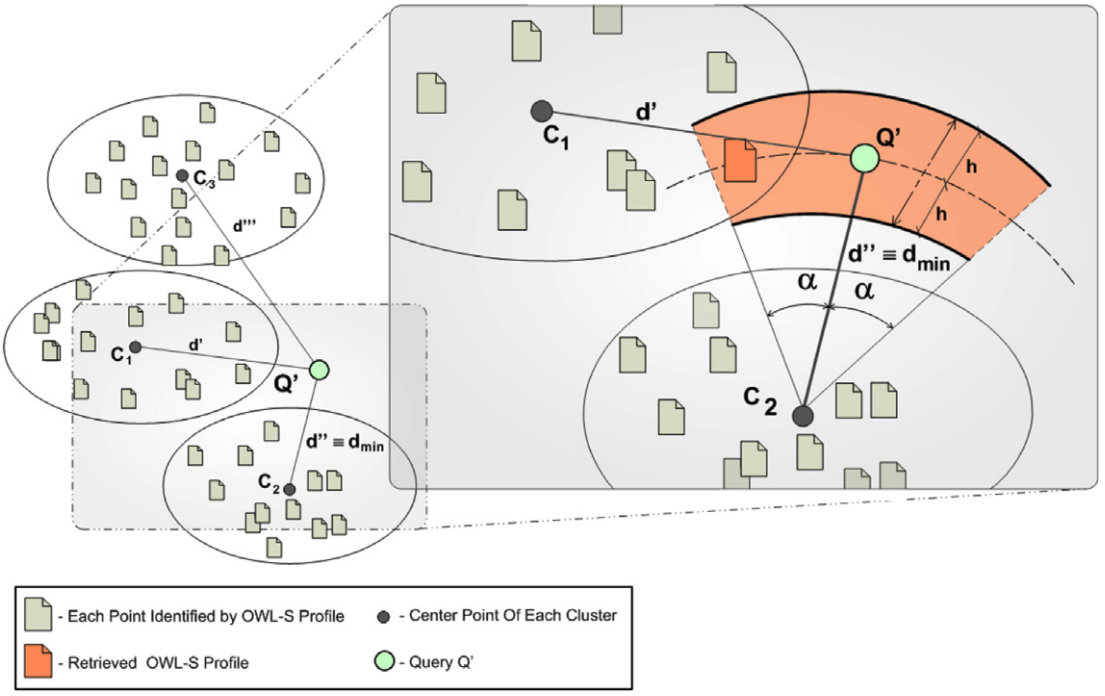

Fig. 6. Cone formatted by the given threshold (angle α and width h).

the "action scope" of the query Q'. The "action scope" of the query depends on an angle α and a width h (see the zoom on the right side of Fig. 6); varying these two parameters, the action scope can be bigger or smaller, allowing the retrieval of a number of services more or less close. The expansion of this area can be set, by acting on an input variable which represents the *precision degree of services retrieval*; it is connected to the parameters α and h, thus modifying it, these parameters vary and, consequently, the percentage of the retrieved answer set varies too. Briefly given a service request, the matchmaking activity (driven by the broker agent) harvests all the services descriptions encompassed in the zone of its "action scope", computed by the input value of the *precision degree of services retrieval*. The following two main steps are performed in this phase:

- *Evaluation of the nearest cluster*: the system computes, in terms of distance, the cluster, whose center is closer to the given request. Let us note this step takes constant time. The only system's cost is the start-up activities. In Fig. 6, the cluster is C_2 is the selected one (the distance "d" is the minimal).
- *Harvesting of all the services enclosed in the selected zone*: acting on the *precision degree of services retrieval*, the angle α and the width h can be modified. The angle is measured starting by the imaginary line that links the center of the cluster to the query (i.e. the distance) and moving in clockwise and counterclockwise of α degrees; similarly the width is measured starting to the query position and advancing of $+h$ and $-h$ in the same direction of the imaginary line center-query. In particular, the angle can assume values in the range 0–180, whereas the width "h" (see Fig. 6) varies in the range of distance 0 to a maximal distance, computed between the center of the cluster and the services further to this cluster. Fig. 6 shows the sketched area for the given query. All the services placed in this area are retrieved.

Varying the *precision degree of service retrieval*, a different number of services can be collected; then the retrieved answer set can contain services that are in the right cluster with high membership, but, at the same time, services that are in other clusters can be gathered too (if the angle is big). These results do not necessarily represent a weakness of our system; rather they can evidence the fact that some characteristics (features) are common to services in different services groups.

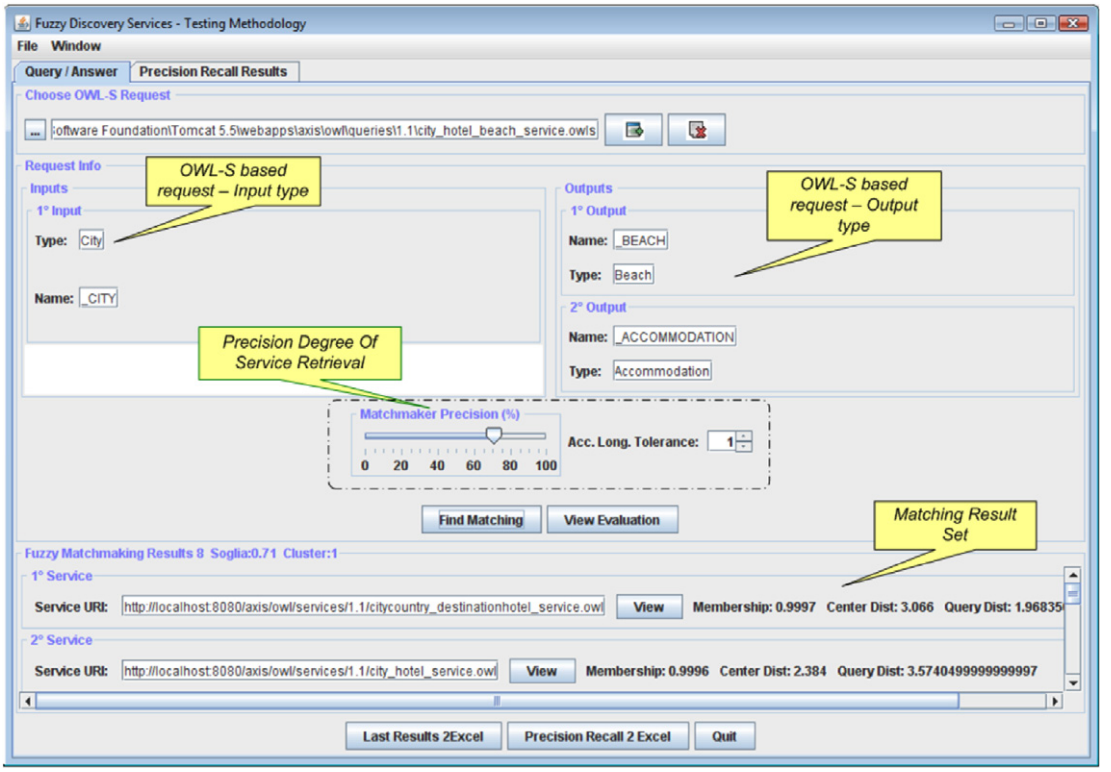

Fig. 7. Interface view.

Fig. 7 presents the application screenshot, split in some parts: on the top, the OWL-S input/output service request is shown, whereas on the bottom the ranked set of the related matching results (i.e. URI services) is listed. Moreover, the precision degree of services retrieval is controlled by the sliding bar, in the center of the panel, that acts on the angle and width, as previously mentioned.

Let us note, during these activities, the broker stores references to all that queries that are unsolved: indeed, it collects all the requested but unfound concepts, given in the query. Periodically, when it is required (i.e. when the number of mismatching concepts becomes big), the clustering algorithm is recomputed, eventually adding the "saved" concepts in the features set.

7. Experimental results

Our approach has been evaluated on the test collections created for OWLS-MX [17]. The authors in [17] define a hybrid approach which exploits both logic-based reasoning and content-based information retrieval techniques: when the logic-based filters do not return the expected semantic web services, textual analysis methods based on similarity metrics refine and improve the results. Our approach instead, considers the evaluation of the retrieval performance, through the use of techniques of Soft Computing. Indeed, we exploit the clustering for getting grouping of web services joint by similar characteristics, whereas the fuzzy component allows us to model the web services and describe how a web service is well represented in a certain cluster. Further details about the implementation of the system and some experimental results are given in the following section.

7.1. Implementation

Our prototype has been implemented in Java and achieves a multi-agent system, based on the platform Jade [12]. It exploits the OWLS API 1.1, which provides a programmatic access to OWLS service descriptions and

OWL-DL reasoner, based on the tableaux algorithms Pellet [2]. The OWLS API is tightly coupled with the Jena Semantic Web Framework [1]. We have extended the Axis OWLS plugin [24] in order to allow the advertiser agents to query the list of available services, offered by the providers and then to retrieve each OWLS description. The approach considers a preliminary phase of the system configuration which includes the following activities:

- *Indexing of OWLS description of the available web services*. The agents consult indexes in order to get urls, which are the references to the locations of the OWLS descriptions (or eventually the urls of the Axis provider where our plug-in is installed). This information is necessary to retrieve the OWLS profiles of the services. The OWLS API parses and loads the service description and then, elicits the ontology terms, by exploring their "definition" in the reference ontologies. The collected information represents a knowledge base on which the Pellet reasoner can infer and answer to the service requests.
- *Feature set selection*. As said, the features set can be composed of all the discovered ontology terms; sometimes a restricted subset can be exploited. The filtering of the terms as well as the discovery of relationships between them are activities entrusted to the reasoner Pellet that intervenes in the analysis of the hierarchical structure of the involved ontologies.
- *Data matrix building*. The data matrix is composed of all the indexed service, represented as fuzzy multisets (see Section 5.2). The Pellet reasoner plays a crucial role for deducting the correlations between ontology terms in service profiles and the terms in the features set. In fact, the discovered correlations are generically relationships of specialization and generalization (i.e. not exclusively direct relationships, such as parent–child). The evaluation of these relationship (as given in Section 5.2) allows the building of data matrix.
- *Clustering execution*. The clustering activity produces a fuzzy partition of the OWLS description of the semantic web services (see Section 5.3). In other words, the clustering achieves grouping of web services that evidence similar characteristics. The membership distribution for each service among the clusters, describes how that service belongs to each cluster. The service is assigned to the cluster in which it has highest membership value. On the other hand, the service which presents membership equally distributed among the clusters, emphasizes two aspects: 1) the features set does not contain representative features (i.e. ontology terms) for the service; 2) because the FCM algorithm uses a prefixed number of clusters, this number could be not appropriately chosen.

Although these startup activities are time consuming, some of them (i.e. the building of features set and data matrix and the clustering) are executed only periodically, for instance when new services must be added to the service index.

7.2. Test case

Our experimentation exploits the relevance set defined in OWLS-MX approach [17], consisting of 503 indexed OWLS-based services. This collection is a subset of the OWLS-MX test case, which is composed of the OWLS 1.1 service advertisements and queries descriptions (and the relative referenced OWL ontologies repository).

Table 1 shows in details the involved services, the reference categories and the relative queries. As evinced by the Table 1, the medical domain has not been considered, because some ontology-based inconsistency have been revealed, during the inference activity of Pellet reasoner. In this test case, all the ontology terms returned by the Pellet's process are in the feature set. On the basis of the system setting, the computed feature set consists of 242 ontology terms.

Furthermore, in order to get an adequate comparison with the OWLS-MX experimentation, we have tried to set up a similar starting configuration. Thus, we restrict to the ontology terms in the Input and Output specification only. This test case has been executed on the OWLS Profile description of semantic web services, by considering only the terms declared in the *hasInput* and *hasOutput* specifications of OWLS IOPRs. Consequently, the fuzzy multisets are composed of couple of membership values associated to the input and output types of the ontology terms collected during the parsing of OWLS profiles. Recalling the OWLS Profile described in Section 2.1, the selected terms are _COUNTRY_CODE and _CITY as input, whereas _DESTINATION and _HOTEL as output. The heuristic-based rules assume the following form:

Table 1
Services and queries for each category of semantic web services

Domain	Services	Query
Education	135	6
Food	25	1
Medical	//	//
Travel	106	5
Communication	24	2
Economy	189	6
Weapon	24	1

If an ontology term is in the Input, the relevance degree is 0.3.
If an ontology term is in the Output, the relevance degree is 0.7.

The choice of these relevance values aims at underlining the different roles of the input and output types of the ontology terms: goal is to assign the two types value quite distant (i.e. different). We have noted empirically that this choice assures a more accurate partitioning when the clustering is performed.

As known, the FCM is a clustering method which needs predefined cluster number. The choice of the number of cluster is immediately related to the matchmaking performance. By the empirical analysis of data matrix and the partition matrices, obtained considering different settings of clustering executions, the number of clusters has been set to 8.

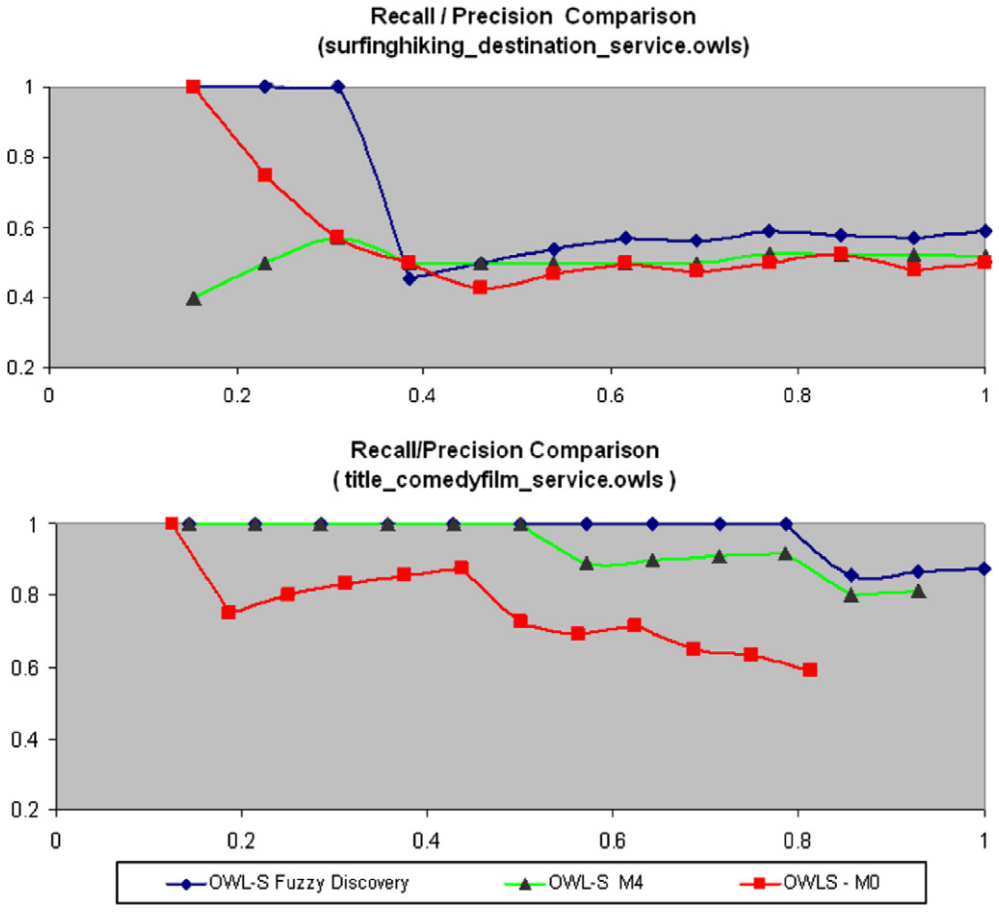

Fig. 8. Compared evaluation of precision/recall on two queries.

The evaluation of the answer set depends on the variation of the *precision degree of service retrieval*: we consider the minimal value this variable must assume because the whole relevance set defined in the OWLS-MX experimentation can be retrieved.

7.3. Results evaluation

The retrieval performance of the system is assessed in terms of precision and recall measures. Like in OWLS-MX approach we consider the evaluation of micro-average of the individual precision-recall curves, exploiting a number $\lambda = 20$ of steps up to reach the highest recall value. Thus, let Q be the set of service requests, D all the relevant service OWLS descriptions of all requests in Q. According to [29] the micro-averaging of recall and precision (at step λ) overall requests, is defined as follows:

$$Rec_\lambda = \sum_R \frac{|D_R \cap B_{\lambda,R}|}{|D|}, \quad Prec_\lambda = \sum_R \frac{|D_R \cap B_{\lambda,R}|}{|B_\lambda|} \tag{11}$$

where D_R is the answer set of relevant services (service advertisements) for given request R, B_λ the set of retrieved OWLS descriptions at the step λ and $B_{\lambda,R}$ is the set of all relevant OWLS descriptions, retrieved at the step λ. Fig. 8 shows the precision/recall curves computed on two service requests *surfinghiking_destination_service.owls* and *title_comedyfilm_service.owls* individually (in this case, the sums in Eq. (11) are irrelevant, because they are computed on a single service request R). The curve obtained by our approach (*OWLS Fuzzy Discovery* line in Fig. 8) is compared with the results computed on the same queries, by exploiting the OWLS-MX algorithm [17]. In fact, the experimentation performed on the OWLS-MX approach is based on different variants of their matchmaker: they range over a purely logic-based approach, named OWLS-M0, to (four) more hybrid variants, called OWLS-M1 to OWLS-M4. Conversely, our matchmaker exploits only the ontological relationships discovered by Pellet reasoner and evaluates the service request, considering the minimal distance to the computed clusters. In Fig. 8 we compare our results with the OWLS-M0 and OWLS-M4 variants that represent the two extremities: the first one exploits only logic-based filters (EXACT, PLUG-IN, etc.), whereas the second one is a hybrid approach based on Information Retrieval techniques, using syntactic similarities. As shown in Fig. 8, our curves tendency shows comparable results to the OWLS MX matchmaker for both the queries.

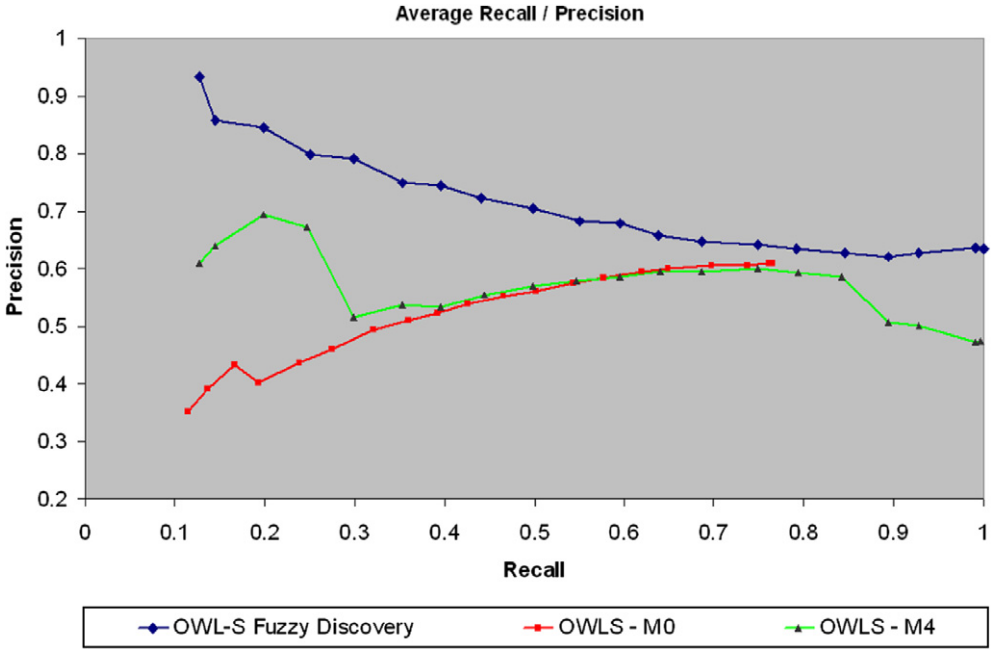

Fig. 9. Comparison of precision/recall measurement with the OWLS-MX approach.

In particular, Fig. 9 shows the tendency of the micro-average of recall/precision curve evaluated on the whole collection set. Let us observe that the performances are very close to each others. Rather, our performance curve seems to show best results, overcoming all OWLS MX variants, even it outruns the OWLS-M4 which applies the nearest-neighbour filter. The results of this experimentation can be downloaded to http://www.lasa.dmi.unisa.it. Let us highlight the query/answering time of our system is insignificant, and no optimization techniques have been adopted. In this sense our system presents performance that are very similar to a traditional web search engine.

8. Conclusions

This approach provides a hybrid service-oriented architecture, that solves dynamic requirements through an agent-based interaction. The framework has been training for discovering and matchmaking of OWLS-based web services, through fuzzy techniques. The matchmaking activity is based on the use of both the fuzzy multiset for representing granular information and the clustering algorithm for grouping the OWLS documents.

The experimentation reflects the setting configuration of OWLS-MX approach: in order to get comparable results, indeed, only the ontology terms appearing as input and output in the IOPRs of the OWLS Profile document have been considered. Additional experimentations based on many IOPRs have been executed and others are still in progress; preliminary considerations reveal reliable results; the role of the value degrees computed by the heuristics-based rules is relevant too, in order to appropriately build the fuzzy multiset and then the data matrix (see Sections 3 and 5.2).

In conclusion, this approach represents a hybrid system for semantic web service matchmaking that synergically combines different technologies and methodologies, listed as follows:

- The modeling of a flexible agent-based approach, to deal with the stringent requirement of the semantic web services environments: discovery, retrieval and matchmaking.
- A fuzzy discovery mechanism deals with the semantics of concepts: the fuzzy multiset model provides a flexible representation of the OWLS-based services, through the identification of the involved terms, and the IOPR roles played.
- A collection of information, through a periodical classification of the updated service capabilities and functionalities. It is important to accurately select the features for the clustering: a good features set often corresponds clusters which appropriately reflect the categories of services description and provide good replies to the service request.
  Furthermore, the query–answering time of the system is relatively short. The main cost in terms of time is due to the re-execution of the clustering, when it is required.
- The data matrix maintains the taxonomic relationships of generalization/specialization between the ontology terms discovered during the analysis of the OWL-S Profiles. In addition, its columns are the terms of the features set. Consequently, an ontology term which does not appear in the features set, is represented in the data matrix through some existing relationships with neighbour terms, declared in the reference ontology.

A feasible extension of this framework is the ranking and filtering of the retrieved services: a deeper investigation on the relationships between the ontology terms in the services request and in the services deployment could provide further information about the input/output roles played by terms, for instance, exploiting the controvariance criteria defined in [13]. Moreover, additional ontology terms correlation like disjointWith, intersectionOf, etc. have been taken into account during the building of fuzzy multisets.

Further tests have been executed and some ones are still in progress; in particular, we are working to improve the "replacing" of approximate service requests, when no service advertisement matches the request. This way, our work aims at improving the agent behavior during the matchmaking activity, in order to enable an ontological reconciliation when different ontology definitions appear, for the same concept.

References

[1] Jena – A Semantic Web Framework for Java, <http://jena.sourceforge.net/>.
[2] Mindswap Maryland Information and Network Dynamics Lab Semantic Web Agents Project, <http://www.mindswap.org/>.

[3] J.C. Bezdek, Pattern Recognition and Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.
[4] C. Rey, F. Toumani, B. Benatallah, M. Hacid, Request rewriting-based web service discovery, Lecture Notes in Computer Science 2870 (2003) 242–257.
[5] H. Lausen, D. Roman, U. Keller, Web Service Modeling Ontology WSMO – Standard, <http://www.w3.org/Submission/WSMO/>.
[6] DAML + OIL Reference Description, W3C Note, <http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218>, 2001.
[7] K. Decker, M. Williamson, K. Sycara, Matchmaking and brokering, in: Proceedings of the Second International Conference in Multi-Agent Systems (ICMAS'96), Kyoto, Japan, 1996.
[8] G. Fenza, V. Loia, S. Senatore, L. Scarpa, Flexible geospatial web services through fuzzy multisets, in: Janis Grundspenkis, Marite Kirikova (Eds.), 2006 IEEE International Conference on Granular Computing, GSU, Atlanta, USA, May 10–12, 2006, pp. 514–517.
[9] FIPA, Foundation for intelligent physical agents, URL: <http://www.fipa.org/>.
[10] N. Guarino, C. Welty, An overview of OntoClean, in: S. Staab, R. Studer (Eds.), Handbook on Ontologies, Springer-Verlag, Heidelberg and Berlin, 2004.
[11] H. Wang, Y.Q. Zhang, R. Sunderraman, Soft Semantic Web services agent, in: IEEE Annual Meeting of the Fuzzy Information Processing NAFIPS'04, vol. 1, 27–30 June, 2004, pp. 126–129.
[12] JADE, Java Agent DEvelopment Framework, URL: <http://jade.tilab.com/>.
[13] M.C. Jaeger, G. Rojec-Goldmann, C. Liebetruth, G. Mühl, K. Geihs, Ranked Matching for Service Descriptions Using OWL-S, KiVS, 2005, pp. 91–102.
[14] M.C. Jaeger, S. Tang, Ranked Matching for Service Descriptions using DAML-S, in: Janis Grundspenkis, Marite Kirikova (Eds.), Proceedings of CAiSE'04 Workshops, Riga, Latvia, June 2004, pp. 217–228.
[15] C. Lo, T. Tan, K. Chao, M. Younas, Fuzzy Matchmaking for Web Services, in: 19th International Conference on Advanced Information Networking and Applications, vol. 2, Taiwan, 2005, pp. 721–726.
[16] A. Sheth, A. Patil, S. Oundhakar, K. Verma, K. Sivashanmugam, J. Miller, METEOR-S WSDI: a scalable infrastructure of registries for semantic publication and discovery of web services, Journal of Information Technology and Management 6 (1) (2005) 17–39.
[17] M. Khalid, K. Sycara, M. Klusch, B. Fries, OWLS-MX: hybrid semantic web service retrieval, in: Proceedings of the 1st International AAAI Fall Symposium on Agents and the Semantic Web, AAAI Press, Arlington, VA, USA, 2005.
[18] L. Li, I. Horrocks, A software framework for matchmaking based on semantic Web technology, in: WWW, 2003, pp. 331–339.
[19] V. Loia, W. Pedrycz, S. Senatore, Semantic web content analysis: a study in proximity-based collaborative clustering, IEEE Transaction on Fuzzy Systems 15 (6) (2007) 1294–1312.
[20] V. Loia, W. Pedrycz, S. Senatore, M.I. Sessa, Web navigation support by means of cognitive proximity-driven assistant agents, Journal of the American Society for Information Science and Technology (JASIST) 57 (4) (2006) 515–527.
[21] L.D. Martin, M. Paolucci, S.A. McIlraith, M.H. Burstein, D.V. McDermott, D.L. McGuinness, B. Parsia, T.R. Payne, M. Sabou, M. Solanki, N. Srinivasan, K.P. Sycara, Bringing semantics to web services: the OWL-S approach, in: SWSWPC, 2004, pp. 26–42.
[22] S. Miyamoto, Information clustering based on fuzzy multisets, Information Processing and Management 39 (2003) 197–213.
[23] OWL Services Coalition, OWL-S: Semantic Markup for Web Services, <http://www.daml.org/services/owl-s/>, 2004.
[24] OWL-S Plugin, <http://www.mcj.de/mirrors/owlsplugin/>.
[25] V.C. Storey, Comparing relationships in conceptual modeling: mapping to semantic classifications, IEEE Transactions on Knowledge and Data Engineering 17 (11) (2005) 1478–1489.
[26] K.P. Sycara, M. Paolucci, A. Ankolekar, N. Srinivasan, Automated discovery, interaction and composition of Semantic Web services, J. Web Sem. 1 (1) (2003) 27–46.
[27] K.P. Sycara, M. Paolucci, J. Soudry, N. Srinivasan, Dynamic discovery and coordination of agent-based semantic web services, IEEE Internet Computing 8 (3) (2004) 66–73.
[28] UDDI, Universal Description, Discovery, and Integration, URL: <http://uddi.org/>.
[29] C.J. Van Rijsbergen, Information Retrieval, second ed., Dept. of Computer Science, University of Glasgow, 1979.
[30] WSDL, Web Services Description Language, URL: <http://www.w3.org/TR/wsdl>.
[31] J. Wu, Z. Wu, Similarity-based web service matchmaking, in: IEEE International Conference on Services Computing, 2005, pp. 287–294.
[32] R.R. Yager, On the theory of bags, International Journal of General Systems 13 (1986) 23–37.