Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S

Georgios Meditskos and Nick Bassiliades, *Member, IEEE,*

Abstract—In this paper, we describe and evaluate a Web service discovery framework using OWL-S advertisements, combined with the distinction between service and Web service of the WSMO Discovery Framework. More specifically, we follow the Web service discovery model, which is based on abstract and lightweight semantic Web service descriptions, using the Service Profile ontology of OWL-S. Our goal is to determine fast an initial set of candidate Web services for a specific request. This set can then be used in more fine-grained discovery approaches, based on richer Web service descriptions. Our Web service matchmaking algorithm extends object-based matching techniques used in Structural Case-based Reasoning, allowing (a) the retrieval of Web services not only based on subsumption relationships, but exploiting also the structural information of OWL ontologies, and (b) the exploitation of Web services classification in Profile taxonomies, performing domain-dependent discovery. Furthermore, we describe how the typical paradigm of Profile input/output annotation with ontology concepts can be extended, allowing ontology roles to be considered as well. We have implemented our framework in the OWLS-SLR system, which we extensively evaluate and compare to the OWLS-MX matchmaker.

Index Terms—Web service discovery, abstract descriptions, OWL-S Profile, structural information, role-oriented matchmaking.

1 Introduction

Web services have brought a communication revolution in heterogeneous domains where the efficient collaboration among different parties is important, such as in e-commerce and e-business. However, the increasing use of Web services has raised new challenges, such as the automated Web service discovery. Web is continuously enriched with Web services and it is transformed from a Web of documents into a Web of documents and services. The problem that arises is how a human or an agent could be assisted during service selection. The XML representation of Web services (WSDL [1]) guarantees syntactic interoperability but it is unable to semantically describe services.

Semantic Web services (SWSs) [2] aim at making Web services machine-understandable and use-apparent, utilizing Semantic Web technologies for Web service annotation and processing. The idea is to provide ontology-based descriptions of Web services that could be processed by ontology reasoning tools. In that way, intelligent agents would be able to automatically understand what a Web service does and what it needs in order to perform a task.

In this paper, we adopt a conceptual model for semantic Web services [3] and we follow the WSMO Discovery Framework [4] (WSMO-DF) for Web service discovery, using descriptions that are expressed as instances of the Profile concept of the Service Profile (SP) of the OWL-S ontology [5]. The rationale is to use lightweight Web service descriptions based on inputs, outputs and non-functional properties, in order to determine fast an initial set of candidate Web services for a request. Our framework can be considered as a prephase of more complex discovery frameworks that make use of richer Web service descriptions, for example, precondition, effects or state transitions, narrowing the space where they should be applied on. Our approach has been realized in the OWLS-SLR system, which we extensively describe and evaluate.

The contributions of our work can be summarized in the following:

- We combine object-based structural matching techniques that are used in the domain of Structural Case-based Reasoning (SCBR) [6], with Description Logic (DL) reasoning [7] over Profile instances, enhancing the discovery with services that cannot be retrieved using only logic-based reasoning.
- We allow the existence of Profile taxonomies, incorporating domain knowledge through Profile instance class membership relationships.
- We enhance the discovery procedure of our framework by considering also ontology roles, exploiting the excellent classification capabilities of DL reasoning. In that way, we combine the strong points of the WSMO-DF and OWL-S SP modeling paradigms.

The rest of the paper is structured as follows: in section 2 we present the basic background and our motivation. In section 3 we present the matching techniques used in SCBR frameworks. In section 4 we extend the SCBR metrics to the SP model for SWS discovery and we describe implementation aspects of OWLS-SLR. In section 5 we analyze experimental results and we compare OWLS-

- *The authors are with the Department of Informatics, Aristotle University of Thessaloniki, 54124, Greece.*
E-mail: {gmeditsk, nbassili}@csd.auth.gr

Manuscript received xx; revised xx; accepted xx; published online xx.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-09-0507. Digital Object Identifier no. xx.

SLR to the OWLS-MX [8] matchmaker. In section 6 we introduce ontology roles as annotation concepts in the SP model. Finally, in sections 7 and 8, we review related work and we conclude, respectively.

2 Background and Motivation

2.1 Semantic Annotation of Web Services

Web service discovery can be defined as the problem of locating suitable Web services to fulfill a given objective. In the SWS paradigm, discovery is performed over semantic descriptions of Web services. WSMO-DF and OWL-S SP are two frameworks that regulate the way descriptions should be defined (see also section 7).

2.1.1 The WSMO Discovery Framework

WSMO-DF is based on the WSMO framework [9] for Web service discovery. In WSMO-DF, a *Web service* is a computational entity which is able, by invocation, to achieve a goal. A *service*, in contrast, is the actual value provided by this invocation [3]. Therefore, there are abstract Web service and concrete service descriptions. The former describe Web services in terms of their abstract functionality, whereas the latter contain a more detailed information about the service. For example, a hotel offers an abstract service for booking rooms, and requesters provide concrete descriptions of their requirements, for example, number of rooms, date, etc.

In accordance to the distinction between Web service and service, WSMO-DF follows two levels of abstraction during Web service discovery, based on Light or Rich semantic descriptions. In the former, Web services are represented as Complex Concepts (CCs), mapping them in domain ontologies *as a whole*, and the matchmaking examines the subsumption relationships of CCs. The latter is the most fine-grained level, where Web services are modeled in terms of state transitions, obligations of requesters, etc.

2.1.2 The OWL-S Service Profile Ontology

OWL-S is an OWL ontology [10] that offers the conceptual model for semantically annotating Web services. The modeling is performed based on four upper ontologies, namely Service, Service Profile (SP), Service Process and Service Grounding. In brief, the SP provides the information needed for an agent to discover a service (*advertisement*). An advertisement contains descriptive information, such as the service name, and information about the provider. It describes also the functional properties of the service, that is, inputs, outputs, preconditions and effects, and non-functional properties, such as quality. The Service Process and Service Grounding provide information for an agent to make use of a service.

Each advertisement can be either a direct instance of the OWL-S Profile concept or it can be defined based on a Profile subclass hierarchy [11]. The Profile-based Web service discovery involves the procedure of matchmaking service requests and advertisements, both represented as Profile instances. Inputs and outputs (I/Os) are annotated with ontology concepts (signature [12]), and preconditions and effects (specification) are described using a rule formalism.

TABLE 1
Web Service Description Examples (DL syntax)

Domain Ontology Axioms
$Order \sqsubseteq Economy \sqcap \exists title.\top \sqcap \exists account.\top \sqcap \exists item.\top \sqcap \exists to.\top$, $Search \sqsubseteq Education \sqcap \exists title.\top \sqcap \exists item.\top$, $Economy \sqsubseteq Profile$, $Education \sqsubseteq Profile$, $Book \sqsubseteq Item$, $Magazine \sqsubseteq Item$, $gr : Country$, $uk : Country$
Complex Concepts
1) $a1 \equiv Order \sqcap \forall title.Title \sqcap \forall item.Book \sqcap \forall account.User \sqcap \exists to.\{gr\}$
2) $a2 \equiv Order \sqcap \forall title.Title \sqcap \forall item.Magazine \sqcap \forall account.User \sqcap \exists to.\{gr\}$
3) $a3 \equiv Order \sqcap \forall title.Title \sqcap \forall item.Magazine \sqcap \forall account.User \sqcap \exists to.\{uk\}$
4) $a4 \equiv Search \sqcap \forall title.Title \sqcap \forall item.Book$
OWL-S Service Profile Instances
1) $a1 : Order$, $\langle a1, Title\rangle : hasInput$, $\langle a1, User\rangle : hasInput$, $\langle a1, Book\rangle : hasOutput$, $\langle a1, gr\rangle : to$
2) $a2 : Order$, $\langle a2, Title\rangle : hasInput$, $\langle a2, User\rangle : hasInput$, $\langle a2, Magazine\rangle : hasOutput$, $\langle a2, gr\rangle : to$
3) $a3 : Order$, $\langle a3, Title\rangle : hasInput$, $\langle a3, User\rangle : hasInput$, $\langle a3, Magazine\rangle : hasOutput$, $\langle a3, uk\rangle : to$
4) $a4 : Search$, $\langle a4, Title\rangle : hasInput$, $\langle a4, Book\rangle : hasOutput$

Example. Table 1 depicts four Web services advertisements using the CC and the OWL-S SP approaches. The *a1* advertisement is classified in the *Order* class and requires a title and an account as inputs in order to return a book that can be sent to Greece. Similarly, the *a2* advertisement is classified in the *Order* class and requires a title and an account in order to return a magazine that can be sent to Greece. The *a3* advertisement is also classified in the *Order* class and requires a title and an account in order to return a magazine that can be sent to UK. Finally, the *a4* advertisement is classified in the *Search* class and returns a book based on the title. The above characteristics are expressed in the CC model by defining appropriate complex classes that describe services as a whole, whereas in the OWL-S SP model each advertisement is expressed as an instance of the appropriate Profile subclass.

2.2 Motivation

We define our motivation in terms of our decision to follow the SP model instead of the CC model, and to incorporate structural ontology information and roles.

2.2.1 Complex Concepts and Profile Instances

Our decision to use the SP model for describing Web services targets at the usability of the framework. We argue that it is more intuitive for providers and for average users to advertise Web services following the SP model, annotating I/O parameters. The CC approach, although it offers more expressive power, it requires more elaborate skill of the people creating the descriptions.

Furthermore, it is difficult to consider CCs in repositories, for example, in UDDI [13], in contrast to OWL-S Profile instances. The difficulty relies on the fact that a CC does not follow a standard description pattern, since all the properties are considered equal during concept definition (see Table 1). On the other hand, in the SP model, the functional properties are distinguished from the non-functional parameters and they can be mapped on UDDI following a standardized approach [14].

The CC model has increased capabilities in describing the *class of objects* where a Web service can be categorized in terms of subsumption relationships, existential and universal quantifiers. The SP model lacks the ability to incorporate such universal and existential quantifiers, since it defines instances and not concepts. Any role information stems only from values for the roles that the instances inherit from the Profile taxonomy. Moreover, in the case where Web services are described as direct instances of the Profile concept, the domain knowledge can only be captured through special roles, since all the instances belong to the same concept. In our framework, we allow advertisements to be defined in terms of Profile taxonomies, such as the Profile instances of Table 1, capturing domain knowledge through instance class memberships (DL ABox reasoning).

2.2.2 Structural Ontological Knowledge

The matchmaking that is based only on logic-based reasoning, such as the CC approach, computes only the subsumption relationships among the annotation concepts. Therefore, any structural information is ignored, for example, sibling relationships that may enhance the discovery, especially in cases where few or no results are initially returned for a request.

We are motivated by the usefulness of the structural information and we introduce in our matchmaking algorithm matching techniques that are used in Structural Case-based Reasoning (SCBR), a specialized approach to Case-based Reasoning. In SCBR, the idea is to represent cases according to a domain model [6] that is structured in an object-oriented manner, including IS-A relationships and inheritance. Each case and query is modeled as an object and the additional knowledge that stems from the model is used during matching. The object relevance is determined using the *interclass* and *intraclass* metrics. The former is defined over the common attributes, whereas the latter is defined upon the class types of two objects.

In our work, we extend the intraclass and interclass notions to the domain of SWS discovery. The idea is to perform matchmaking on Profile instances represented as objects, considering the domain ontologies and any Profile taxonomy as the domain models.

2.2.3 Web Service Descriptions and Ontology Roles

The semantic tagging of I/O parameters with predefined concepts has limited expressive power. For example, consider a Web service advertisement whose one of its inputs is annotated with the concept *Person* that has three roles: *SSN*, *address* and *name*. In this case, we cannot determine what the Web service really requires: *SSN*, *name*, *address* or all of them? On the other hand, the CC approach takes into account roles through restrictions.

In order to overcome this limitation of the SP paradigm, we enhance our framework with the ability of a role-based Web service functional annotation based on the open-world assumption and the classification capabilities of the DL reasoning paradigm. In that way, we are able to extend the annotation and discovery procedures of our SP-oriented framework with ontology roles. Actually, our approach is an effort to leverage the modeling differences between the CC and SP paradigms, where the former is concept-oriented, allowing the full exploitation of the logical formalism that is used to define concepts, whereas the latter is instance-oriented, treating Web service descriptions as Profile instances.

3 SCBR Similarity Metrics

In SCBR, both cases and queries are represented as objects, enhancing the typical attribute-value representation of the traditional CBR with domain knowledge.

Definition 1. *An Object O is a triple $\langle ID, C, P\rangle$, where ID is the unique identifier of the object, C is the object class type, and P is a set that contains attribute-value pairs of the form $\langle p, V\rangle$, where p is an attribute and V a set of values.*

The domain model is represented as a class hierarchy and the objects are initialized with a single class type and property-value definitions. For example, let $A \preceq B$ denote that class A is subclass of class B, $p \in Att(A)$ denote that the attribute p is defined in class A, $o.p$ denote the set of values of object o for property p, that is, the set V, and $o \rightarrowtail A$ denote the class type of object o. Let three classes A, B and D, where $A \preceq B$ and $B \preceq D$. If $o \rightarrowtail A$, then o is also an object of B and D, due to inheritance. Furthermore, let p_A and p_B be two attributes, where $p_A \in Att(A)$ and $p_B \in Att(B)$. If $o \rightarrowtail A$, then both expressions $o.p_A$ and $o.p_B$ are valid, since attributes are inherited to subclasses.

In that way, every object encapsulates domain knowledge, regarding class relationships and property-value definitions, which is used for matching cases and queries through the interclass and intraclass similarity metrics.

3.1 Intraclass Similarity

The intraclass metric defines the similarity of two objects in terms of the values in their common attributes, based on two value matching functions: the V_s function for simple values, for example, integers, strings, etc., and the V_r function for relational values, that is, objects.

Let two objects $O_A = \langle ID_A, C_A, P_A\rangle$ and $O_B = \langle ID_B, C_B, P_B\rangle$ and their common attribute p. The partial intraclass similarity S_p for the property p is defined as

$$S_p(O_A, O_B) = \begin{cases} V_s(ID_A.p, ID_B.p), & \text{if } p \text{ is simple} \\ V_r(ID_A.p, ID_B.p), & \text{if } p \text{ is relational.} \end{cases}$$

The overall intraclass similarity of two objects O_A and O_B is defined by aggregating their partial similarities.

Definition 2. *Let two objects* $O_A = \langle ID_A, C_A, P_A \rangle$ *and* $O_B = \langle ID_B, C_B, P_B \rangle$ *and the set* T *of their common attributes, that is,* $\forall p \in T, \exists \langle p, V \rangle \in P_A \wedge \exists \langle p, V' \rangle \in P_B$. *The intraclass similarity is defined, with respect to an aggregation function* Θ, *as*

$$S_{intra}(O_A, O_B) = \underset{\forall p \in T}{\Theta} S_p(O_A, O_B).$$

3.2 Interclass Similarity

The interclass metric captures the hierarchical relationship of two object class types, based on a *hierarchical matching function* H that denotes the similarity of two objects in terms of their class types.

Definition 3. *Let two objects* $O_A = \langle ID_A, C_A, P_A \rangle$ *and* $O_B = \langle ID_B, C_B, P_B \rangle$. *Their interclass similarity is defined, with respect to a hierarchical matching function* H, *as*

$$S_{inter}(O_A, O_B) = H(C_A, C_B).$$

The overall similarity of two objects is defined by aggregating their intraclass and interclass similarities.

Definition 4. *Let two objects* O_A *and* O_B. *Their similarity* S *is defined, with respect to an aggregation function* Φ, *as*

$$S(O_A, O_B) = \Phi\left[S_{intra}(O_A, O_B), S_{inter}(O_A, O_B)\right].$$

4 OWL-S Profile Metrics

In this section, we describe the $\mathcal{DLH}$ and $\mathcal{DLR}$ Profile-aware similarity metrics, extending the intraclass and interclass SCBR metrics to an ontology environment, and enhancing them with DL reasoning. Firstly, we introduce the notion of the *object specification* for representing advertisement and query instances in our framework.

Definition 5. *An object specification is a quintuple* $\langle \mathcal{ID}, \mathcal{C}, \mathcal{I}, \mathcal{O}, \mathcal{NF} \rangle$, *where* $\mathcal{ID}$ *is the Profile instance identifier,* $\mathcal{C}$ *is the set of the most specific concepts to where* $\mathcal{ID}$ *belongs,* $\mathcal{I}$ *and* $\mathcal{O}$ *are the sets of I/O annotation concepts, respectively, and* $\mathcal{NF}$ *is the set of non-functional property-value pairs.*

We refer to an advertisement instance as an $\mathcal{A}$ specification and to a query instance as a $\mathcal{Q}$ specification. In that way, the Profile instances of Table 1, which are used as examples in the rest of the paper using the *a1* advertisement as a query, are represented as

$\mathcal{Q} = \langle$*a1*, {*Order*}, {*Title*, *User*}, {*Book*}, {⟨*to*, *gr*⟩}⟩
$\mathcal{A}_2 = \langle$*a2*, {*Order*}, {*Title*, *User*}, {*Magazine*}, {⟨*to*, *gr*⟩}⟩
$\mathcal{A}_3 = \langle$*a3*, {*Order*}, {*Title*, *User*}, {*Magazine*}, {⟨*to*, *uk*⟩}⟩
$\mathcal{A}_4 = \langle$*a4*, {*Search*}, {*Title*}, {*Book*}, {}⟩.

We approach the SWS discovery problem as the procedure of determining the similarity of an $\mathcal{A}$ ($\langle \mathcal{ID}_a, \mathcal{C}_a, \mathcal{I}_a, O_a, \mathcal{NF}_a \rangle$) and $\mathcal{Q}$ ($\langle \mathcal{ID}_q, \mathcal{C}_q, \mathcal{I}_q, \mathcal{O}_q, \mathcal{NF}_q \rangle$) specification, based on three levels of similarity:

1) *Taxonomical Similarity (TS)*. It is computed over the $\mathcal{C}_a$ and $\mathcal{C}_q$ sets of an $\mathcal{A}$ and $\mathcal{Q}$ specification and denotes their similarity in terms of their taxonomical categorization in a Profile subclass hierarchy.
2) *Functional Similarity (FS)*. It is computed over the input ($\mathcal{I}_a$ and $\mathcal{I}_q$) and output ($\mathcal{O}_a$ and $\mathcal{O}_q$) sets of an $\mathcal{A}$ and $\mathcal{Q}$ specification (*signature similarity*).
3) *Non-Functional Similarity (NFS)*. It is computed over the values of the common non-functional properties of an $\mathcal{A}$ and $\mathcal{Q}$ specification.

4.1 The $\mathcal{DLH}$ Metric

The $\mathcal{DLH}$ metric represents the similarity of two ontology concepts in terms of their hierarchical position. It depends on a concept similarity function S, and on a set F of hierarchical filters. In the following, we assume that $S(A, B)$ denotes the similarity of two concepts A and B, with respect to the function S, and that $S(A, B) \in [0..1]$, with 1 denoting absolute match.

The $\mathcal{DLH}$ metric incorporates four hierarchical filters between two ontology concepts. We use the notation $A \overset{f}{\sim} B$ to denote that A matches to B, with respect to one of the following hierarchical filters f.

1) *exact (e)*. The two concepts should have either the same URI, or they should be equivalent concepts, that is, $A \overset{e}{\sim} B \Leftrightarrow A = B \vee A \equiv B$.
2) *plugin (p)*. The concept B should subsume concept A, that is, $A \overset{p}{\sim} B \Leftrightarrow A \sqsubseteq B$.
3) *subsume (su)*. The concept A should subsume concept B, that is, $A \overset{su}{\sim} B \Leftrightarrow B \sqsubseteq A$.
4) *sibling (si)*. The concepts should be subsumed by a concept T and they should not be disjoint, that is, $A \overset{si}{\sim} B \Leftrightarrow \exists T : A \sqsubseteq T \wedge B \sqsubseteq T \wedge A \sqcap B \sqsubseteq \top$.

We generalize the $A \overset{f}{\sim} B$ relation to a set of filters F and we define that the concept A matches the concept B, with respect to a filter set F, if and only if there is at least one filter f in F, such that $A \overset{f}{\sim} B$, that is:

$$A \overset{F}{\sim} B \Leftrightarrow \exists f \in F : A \overset{f}{\sim} B.$$

Definition 6. *Let two concepts* X *and* Y. *Their* $\mathcal{DLH}$ *similarity is the normalized value to* $[0..1]$ *that is defined, with respect to a concept similarity function* S *and a hierarchical filter set* F, *as*

$$\mathcal{DLH}(X, Y, F) = \begin{cases} S(X, Y) & \text{if } X \overset{F}{\sim} Y \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We generalize (1) on two sets S_A, S_B of concepts as

$$\mathcal{DLH}_{set}(S_A, S_B, F) = \frac{\sum\limits_{\forall B \in S_B} \max\limits_{\forall A \in S_A} \left[\mathcal{DLH}(B, A, F)\right]}{|S_B|} \quad (2)$$

Intuitively, for each concept $B \in S_B$ there should be at least one concept $A \in S_A$ relevant to B, with respect to the filter set F. Otherwise, $\mathcal{DLH}_{set}$ returns 0 (absolute mismatch). The overall $\mathcal{DLH}_{set}$ similarity is computed as the mean value of the sum of the maximum $\mathcal{DLH}$s for each concept B, since each B may have more than one relevant concepts in S_A.

The *Taxonomical Similarity* denotes the similarity of two specifications in terms of their concept membership sets $\mathcal{C}_i$ and therefore, it is equal to their $\mathcal{DLH}_{set}$ similarity.

Definition 7. *The taxonomical Similarity between $\mathcal{A}$ and $\mathcal{Q}$ specifications is defined as the $\mathcal{DLH}_{set}$ similarity of their $\mathcal{C}_a$ and $\mathcal{C}_q$ sets, that is,*

$$TS(\mathcal{A}, \mathcal{Q}, F_T) = \mathcal{DLH}_{set}(\mathcal{C}_q, \mathcal{C}_a, F_T), \tag{3}$$

where F_T is the set of the hierarchical relationships that we allow to exists among the concepts of the $\mathcal{C}_a$ and $\mathcal{C}_q$ sets.

4.1.1 $\mathcal{DLH}$ Distance Measures

The $\mathcal{DLH}$ implementation incorporates two concept distance measures D, and therefore, we have that $S(A, B) = 1 - D(A, B)$ in (1). More specifically, we have implemented a variation of (a) the *edge-counting distance (EC)*, an intuitive measure that computes the distance of two concepts based on the number of edges found on the shortest path between them, and (b) the *upwards cotopic (UC)* measure [15] that measures the ratio of the common superclasses of two concepts. We have chosen these two measures for their intuitiveness and the simplicity of the implementation.

4.1.1.1 Edge-counting distance: The edge counting distance (EC) is implemented over the subsumption hierarchy that is computed by the Pellet DL reasoner [16]. An edge exists between two concepts A and B if $A \sqsubseteq_d B \vee B \sqsubseteq_d A$, where $A \sqsubseteq_d B$ denotes that A is a direct subclass of B, ignoring the mutual subsumption edges between equivalent concepts. The implementation of the EC distance between two concepts can be summarized in the following five priorities rules r_i, where $r_1 > r_2 > r_3 > r_4 > r_5$.

- r_1: if $A = B \vee A \equiv B$, then $EC(A, B) = 0$.
- r_2: if $A \sqcap B \sqsubseteq \perp$, then $EC(A, B) = 1$.
- r_3: if $A \sqsubseteq B \vee B \sqsubseteq A$, then $EC(A, B) = e/e_{max}$.
- r_4: if $LCA(A, B) \neq \varnothing$, then $EC(A, B) = \min_{\forall T \in LCA(A,B)} [EC(A, T) + EC(B, T)]$.
- r_5: $EC(A, B) = 1$.

More specifically, if there is a hierarchical relationship between two concepts A and B (r_3), then the EC distance is equal to the number of edges that exist in their shortest path (e) normalized to $[0..1]$ using the maximum EC distance (e_{max}) found in the ontology. In order to compute fast the e_{max}, we approximate it as $e_{max} = 2 \cdot h - 1$, where h is the maximum edge distance from a leaf concept to $owl{:}Thing$ ($\top$).

The *LCA* set (r_4) denotes the set of the *Least Common Ancestors* of two hierarchically unrelated concepts, ignoring $owl{:}Thing$. Pellet computes the ontology classification results as a Directed Acyclic Graph, and thus, more than one least common ancestors might exist for

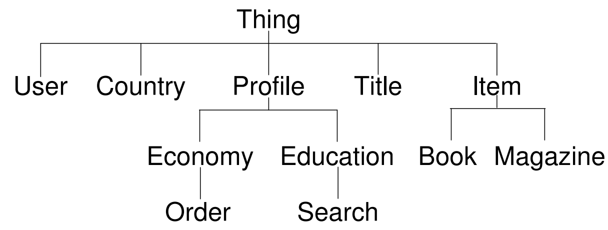

Fig. 1. The hierarchical relationships of Table 1

a concept pair. To this end, the EC distance is determined by the concept T that results in the minimum EC distance. We do not consider the $owl{:}Thing$ concept in sibling relationships, since $\forall A, B : A \sqsubseteq \top \wedge B \sqsubseteq \top$, and thus, no special structural knowledge is provided.

Example. We exemplify on the calculation of the TS for the specifications in Table 1 based on the EC distance. Fig. 1 depicts the hierarchical relationships of the named domain ontology concepts of Table 1, with $e_{max} = 2 \cdot 3 - 1 = 5$. The $\mathcal{A}_2$ and $\mathcal{A}_3$ specifications of Table 1 have the same taxonomical concept to $\mathcal{Q}$, and therefore, $TS(\mathcal{A}_2, \mathcal{Q}, F_T) = TS(\mathcal{A}_3, \mathcal{Q}, F_T)$, with

$$\begin{aligned} TS(\mathcal{A}_3, \mathcal{Q}, F_T) &\overset{(3)}{=} \mathcal{DLH}_{set}(\{\mathit{Order}\}, \{\mathit{Order}\}, F_T) \\ &\overset{(2)}{=} \frac{\mathcal{DLH}(\mathit{Order}, \mathit{Order}, F_T)}{|\{\mathit{Order}\}|} \\ &\underset{e \in F_T}{\overset{(1)}{=}} 1 - EC(\mathit{Order}, \mathit{Order}) \overset{(r_1)}{=} 1, \end{aligned}$$

only if $e \in F_T$, since only the exact filter satisfies the concept relationship in (1). If $e \notin F_T$ then $TS(\mathcal{A}_2, \mathcal{Q}, F_T) = TS(\mathcal{A}_3, \mathcal{Q}, F_T) = 0$.

The $\mathcal{A}_4$ specification has the *Search* taxonomical concept. The EC distance of *Order* and *Search* is $EC(\mathit{Search}, \mathit{Order}) \overset{(r_4)}{=} EC(\mathit{Search}, \mathit{Profile}) + EC(\mathit{Order}, \mathit{Profile}) = \frac{2}{5} + \frac{2}{5} = \frac{4}{5}$, since the minimum path of each concept from the most specific superclass is $e = 2$. In that way,

$$\begin{aligned} TS(\mathcal{A}_4, \mathcal{Q}, F_T) &\overset{(3)}{=} \mathcal{DLH}_{set}(\{\mathit{Order}\}, \{\mathit{Search}\}, F_T) \\ &\overset{(2)}{=} \frac{\mathcal{DLH}(\mathit{Search}, \mathit{Order}, F_T)}{|\{\mathit{Search}\}|} \\ &\underset{si \in F_T}{\overset{(1)}{=}} 1 - EC(\mathit{Search}, \mathit{Order}) \\ &\overset{(r_4)}{=} 1 - \frac{4}{5} = \frac{1}{5}, \end{aligned}$$

only if $si \in F_T$, since the two concepts satisfy only the sibling filter in (1). If $si \notin F_T$ then $TS(\mathcal{A}_4, \mathcal{Q}, F_T) = 0$.

4.1.1.2 Upwards cotopic: The upwards cotopic (UC) measure takes into account the position of a class C in a hierarchy H. It is defined as

$$UC(C, H) = \{A \in H \mid C \sqsubseteq A \vee C = A\},$$

that is, the set of the superclasses of a class C, including C itself. In that way, the distance of two classes A and B in a hierarchy H is defined, in terms of the UC, as

$$\delta(A, B) = 1 - \frac{\left|UC(A, H) \cap UC(B, H)\right|}{\left|UC(A, H) \cup UC(B, H)\right|}.$$

We adjust δ in order to handle ontological concepts and to ignore *owl:Thing*, and we define the distance $\widetilde{\delta}$ of two concepts A and B of an ontology H as

$$\widetilde{\delta}(A,B) = 1 - \frac{\left|UC(A,H) \mathbin{\widetilde{\cap}} UC(B,H)\right| - 1}{\left|UC(A,H) \mathbin{\widetilde{\cup}} UC(B,H)\right| - 1}, \quad (4)$$

where $\widetilde{\cap}$ denotes *semantic concept set intersection* and $\widetilde{\cup}$ *semantic concept set union*, that is,

$$S_A \mathbin{\widetilde{\cap}} S_B = \{x \mid x \mathrel{\widetilde{\in}} S_A \wedge y \mathrel{\widetilde{\in}} S_B\}$$
$$S_A \mathbin{\widetilde{\cup}} S_B = \{x \mid x \mathrel{\widetilde{\in}} S_A \vee x \mathrel{\widetilde{\in}} S_B\}.$$

The $\widetilde{\in}$ notation denotes *semantic set membership*, that is, a concept C semantically belongs to a concept set S, denoted as $C \mathrel{\widetilde{\in}} S$, if $C \in S \vee \exists A \in S : C \equiv A$. Finally, two priority rules are defined in $\widetilde{\delta}$ ($r_1 > r_2$) in order to consider the semantics of class disjointness.

r_1: if $A \sqcap B \sqsubseteq \perp$, then $\widetilde{\delta}(A,B) = 1$.
r_2: $\widetilde{\delta}(A,B) \in [0..1)$.

Example. The TS of the specifications in Table 1 using the UC distance is computed as follows: based on the hierarchy H of Fig. 1 we have that $UC(\mathit{Order}, H) = \{\mathit{Order}, \mathit{Economy}, \mathit{Profile}, \top\}$ and $UC(\mathit{Search}, H) = \{\mathit{Search}, \mathit{Education}, \mathit{Profile}, \top\}$. Therefore, from (4) we have that $\widetilde{\delta}(\mathit{Search}, \mathit{Order}) = 1 - \frac{2-1}{6-1} = \frac{4}{5}$ and finally,

$$\begin{aligned} TS(\mathcal{A}_4, \mathcal{Q}, F_T) &\overset{(3)}{=} \mathcal{DLH}_{set}(\{\mathit{Order}\}, \{\mathit{Search}\}, F_T) \\ &\overset{(2)}{=} \mathcal{DLH}(\mathit{Search}, \mathit{Order}, F_T) \\ &\underset{si \in F_T}{\overset{(1)}{=}} 1 - \widetilde{\delta}(\mathit{Search}, \mathit{Order}) \\ &= 1 - \frac{4}{5} = \frac{1}{5}. \end{aligned}$$

For the other two specifications $\mathcal{A}_2$ and $\mathcal{A}_3$, we have that $TS(\mathcal{A}_2, \mathcal{Q}, F_T) = TS(\mathcal{A}_3, \mathcal{Q}, F_T) = 1$, only if $e \in F_T$, since both specifications have the same taxonomical concepts to the $\mathcal{Q}$ specification.

Note that both the EC and UC distances result in the same TS values for the example, since they are applied to the simple ontology of Table 1. In the general case, the effectiveness of each measure depends on the characteristics of each ontology.

4.2 The $\mathcal{DLR}$ Metric

The $\mathcal{DLR}$ metric denotes the similarity between $\mathcal{A}$ and $\mathcal{Q}$ specifications in terms of the values in their common properties. It is defined in terms of the *Functional* (*FS*) and *Non-Functional* (*NFS*) similarities and of a Web service filter W_f that we analyze in the following sections.

Definition 8. *Let two specifications $\mathcal{A}$ and $\mathcal{Q}$. Their $\mathcal{DLR}$ similarity is the pair $\langle FS, NFS\rangle$ of their Functional and Non-Functional similarities, with respect to the Web service filter W_f, that is,*

$$\mathcal{DLR}(\mathcal{A}, \mathcal{Q}, W_f) = \langle FS(\mathcal{A}, \mathcal{Q}, W_f), NFS(\mathcal{A}, \mathcal{Q})\rangle.$$

4.2.1 Functional Similarity

The FS is based on the $\mathcal{DLH}_{set}$ similarity of the I/O sets of two specifications, so as to ensure that (a) all the advertisement inputs are satisfied by the query inputs, and (b) all the query outputs are satisfied by the advertisement outputs (signature matching).

Definition 9. *The Functional Similarity between $\mathcal{A}$ and $\mathcal{Q}$ specifications is the normalized value to $[0..1]$ that is defined with respect to the Web service filter W_f, as*

$$FS(\mathcal{A}, \mathcal{Q}, W_f) = \sqrt{\mathcal{DLH}_{set}(\mathcal{I}_q, \mathcal{I}_a, F_I) \cdot \mathcal{DLH}_{set}(\mathcal{O}_a, \mathcal{O}_q, F_o)} \quad (5)$$

We use the geometric mean, instead of the arithmetic mean, because a Web service should be excluded if either of its input or output similarity is zero.

In order to control the different degrees of relaxation during I/O matching, the FS makes use of a Web service filter W_f that defines the values of the hierarchical filter sets F_I and F_O in (5). More specifically, we define the Exact (W_e), Plugin (W_p), Subsume (W_{su}) and Sibling (W_{si}) Web service filters with the following relationships to the F_I and F_O filter sets.

- $W_e \rightarrow F_I = F_O = \{e\}$. This is the strictest filter that allows two specifications to match only if they refer to the same or to equivalent concepts in their I/Os.
- $W_p \rightarrow F_I = \{e, p\} \wedge F_O = \{e, su\}$. This is a more relaxed filter and intuitively denotes an $\mathcal{A}$ specification that could be used instead of a $\mathcal{Q}$ specification. The rationale is that all the inputs of the advertisement should be equivalent or subclasses of the query inputs, and all the outputs of the query should be equivalent or superclasses of the advertisement outputs.
- $W_{su} \rightarrow F_I = F_O = \{e, p, su\}$. This filter relaxes even more the matching criterion and the advertisement is allowed to have (a) more general inputs than the query and (b) more general outputs than the query.
- $W_{si} \rightarrow F_I = F_O = \{e, p, su, si\}$. This is the most relaxed filter, allowing also the existence of sibling relationships among I/O concepts.

The order of Web service filter relaxation is $W_e < W_p < W_{su} < W_{si}$. Moreover, for each W_f filter we define three additional levels of granularity based on the number of the I/Os of the query and an advertisement. More specifically, we define the *exclusive* (x), *exclusive-input* (xi) and *exclusive-output* (xo) *grouping filters*. The x filter is satisfied for the matched advertisements that have the same number of I/O parameters to the query. The xi filter is satisfied for the advertisements that have the same number of input parameters only to the query. Similarly, the xo filter deals with the number of output parameters. For example, the xW_e filter matches advertisements that pass the W_e Web service filter and have the same number of I/O parameters to the query. The xiW_{si} filter matches advertisements that pass the

W_{si} Web service filter and have the same number of input only parameters to the query input parameters.

The grouping filtering is motivated by the fact that an advertisement that satisfies, for example, the W_e filter and has the same number of I/Os to the query, should be considered as a more "exact" result than an advertisement with different number of I/O parameters. Based on this assumption, the matched advertisements are returned in groups (see Algorithm 1 in section 4.3), according to the W_f and grouping filter that satisfy. For each W_f, the order of relaxation is $xW_f < xiW_f < xoW_f < W_f$. Therefore, $xW_e < xiW_e < xoW_e < W_e < xW_p < xiW_p < \cdots < xoW_{si} < W_{si}$ (16 grouping filters in total). The decision to define $xiW_f < xoW_f$ is arbitrary.

Example. We exemplify on the computation of the FS of our example, using the EC distance. Both $\mathcal{Q}$ and $\mathcal{A}_2$ specifications have the same input sets $\mathcal{I}_q = \mathcal{I}_a = \{Title, USER\}$ with $\mathcal{DLH}_{set}(\mathcal{I}_q, \mathcal{I}_a, F_I) = 1$, only if $e \in F_I$. For the output sets, we have that $\mathcal{O}_q = \{Book\}$ and $\mathcal{O}_a = \{Magazine\}$ with

$$\begin{aligned}\mathcal{DLH}_{set}(\mathcal{O}_a, \mathcal{O}_q, F_O) &\overset{(2)}{=} \mathcal{DLH}(Book, Magazine, F_O)\\ &\underset{si\in F_O}{\overset{(1)}{=}} 1 - [EC(Book, Item) +\\ &\qquad EC(Magazine, Item)]\\ &= 1 - \left(\frac{1}{5} + \frac{1}{5}\right) = \frac{3}{5},\end{aligned}$$

only if $si \in F_O$, since the two concepts have a sibling relationship and each concept is a direct subclass $(e = 1)$ of the *Item* class. Therefore, from (5), we have that $FS(\mathcal{A}_2, \mathcal{Q}, W_f) = \sqrt{1 \cdot \frac{3}{5}} = 0.774$, only if we select the W_{si} Web service filter that satisfies the hierarchical filter requirements we have mentioned, that is, $e \in F_I$ and $si \in F_O$. Otherwise, $FS(\mathcal{A}_2, \mathcal{Q}, W_f) = 0$. The same holds for the $\mathcal{A}_3$ specification. Furthermore, the $\mathcal{A}_2$ and $\mathcal{A}_3$ specifications satisfy the xW_{si} grouping filter, since they both have the same number of input and output parameters to the $\mathcal{Q}$ specification.

The $\mathcal{A}_4$ specification has the $\{Title\}$ input set and therefore, its input $\mathcal{DLH}$ similarity to $\mathcal{Q}$ is $\mathcal{DLH}_{set}(\{Title, User\}, \{Title\}, F_I) = 1$, only if $e \in F_I$. Furthermore, both $\mathcal{Q}$ and $\mathcal{A}_4$ have the same output sets ($\{Book\}$) and their output $\mathcal{DLH}$ similarity is $\mathcal{DLH}_{set}(\{Book\}, \{Book\}, F_O) = 1$, only if $e \in F_O$. Therefore, from (5), we have that $FS(\mathcal{A}_4, \mathcal{Q}, W_f) = \sqrt{1 \cdot 1} = 1$ that holds for each Web service filter W_f, since $\forall W_f : e \in F_I \wedge e \in F_O$. Moreover, the $\mathcal{A}_4$ specification satisfies the xoW_e grouping filter, since it has only the same number of output parameters to the $\mathcal{Q}$ specification.

4.2.2 Non-Functional Similarity

The NFS is defined in terms of two functions; the dt function for computing the similarity of two datatype values and the ob function for computing the similarity of two object values, where $dt(a, b)$ and $ob(a, b) \in [0..1]$. The overall NFS similarity of two specifications is the mean value of the dt and ob functions over the common properties of the two specifications.

Definition 10. *Let the sets T_d and T_o of the common datatype and object properties, respectively, of $\mathcal{A}$ and $\mathcal{Q}$ specifications with $T_d \cup T_o \neq \varnothing$. The Non-Functional Similarity is the normalized value to $[0..1]$ that is defined, with respect to the functions dt and ob, as*

$$NFS(\mathcal{A}, \mathcal{Q}) = \frac{\displaystyle\sum_{\substack{\forall d\in T_d\\ \forall o\in T_o}} \left[dt\left(\mathcal{ID}_a.d, \mathcal{ID}_q.d\right) + ob\left(\mathcal{ID}_a.o, \mathcal{ID}_q.o\right)\right]}{|T_d \cup T_o|} \quad (6)$$

If $T_d \cup T_o = \varnothing$, then we define that $NFS(\mathcal{A}, \mathcal{Q}) = 1$. The T_d and T_o sets ignore the properties that have a value in an $\mathcal{A}$ but not in a $\mathcal{Q}$ specification, assuming that requesters are not interested in properties that do not annotate in queries.

4.2.2.1 NFS functions: The dt function computes the similarity of two datatype value sets V_A and V_Q by comparing straightforwardly the values. In the case of $xsd{:}string$ ranges we use the Jaro-Winkler similarity measure [17] (*str* function), a normalized to $[0..1]$ metric that calculates the similarity of two strings as

$$dt(V_A, V_Q) = \max_{\substack{\forall v_a \in V_A\\ \forall v_q \in V_Q}} \left[str(v_a, v_q)\right],$$

obtaining the most similar matching combination. In any other case, we compare directly the two value sets, returning a value between 0 and 1 that denotes the similarity of the sets in terms of the ratio of their common values, that is:

$$dt(V_A, V_Q) = \frac{|V_A \cap V_Q|}{|V_A \cup V_Q|}. \quad (7)$$

The ob function computes the similarity of two sets V_A and V_Q of instances in the same way to (7).

Example. For the non-functional object property *to* of Table 1, the $\mathcal{Q}$ specification has the *a1.to* $= \{gr\}$ value set and the $\mathcal{A}_3$ specification the *a3.to* $= \{uk\}$ value set. Therefore, we have that $T_o = \{to\}$, $T_d = \varnothing$ and

$$NFS(\mathcal{A}_3, Q) = \frac{ob(\{uk\}, \{gr\})}{1} = \frac{\left|\{uk\} \cap \{gr\}\right|}{\left|\{uk\} \cup \{gr\}\right|} = 0.$$

For the $\mathcal{A}_2$ specification, we have that $NFS(\mathcal{A}_2, Q) = 1$, since they have the same value set for the object property *to*, that is, *a1.to* $=$ *a2.to* $= \{gr\}$. Finally, for the $\mathcal{A}_4$ specification, we have that *a1.to* $= \{gr\} \neq$ *a4.to* $= \varnothing$. In that way, $NFS(\mathcal{A}_4, Q) = 0$. Note that if we use the $\mathcal{A}_4$ specification as a query, then all the NFS similarities would be equal to 1, since $\mathcal{A}_4$ does not define any non-functional property, and therefore, $T_d \cup T_o = \varnothing$ for each $\mathcal{A}$ specification in (6).

4.3 Overall Specification Similarity

The overall similarity sim of $\mathcal{A}$ and $\mathcal{Q}$ specifications is defined in terms of their TS, FS and NFS similarities.

Definition 11. *Let two specifications $\mathcal{A}$ and $\mathcal{Q}$. Their similarity sim is the triple $\langle TS, FS, NFS \rangle$ of their Taxonomical, Functional and Non-Functional similarities, that is,*

$$sim(\mathcal{A}, \mathcal{Q}, F_T, W_f) = \langle TS(\mathcal{A}, \mathcal{Q}, F_T), FS(\mathcal{A}, \mathcal{Q}, W_f), NFS(\mathcal{A}, \mathcal{Q}) \rangle.$$

The aggregation of the triple similarity into a single value is computed as the weighted mean $\overline{sim}$ of the three similarities according to user requirements, that is,

$$\overline{sim} = \frac{a \cdot TS + b \cdot FS + c \cdot NFS}{a + b + c}$$

where a, b and c are normalized weights in $[0..1]$. The overall matchmaking algorithm of a $\mathcal{Q}$ specification with a set of $\mathcal{A}$ specifications is depicted in Algorithm 1. The algorithm examines the complete set of the advertisements, applying a two-phase filtering based on the taxonomical and functional requirements. The rationale is to prune firstly the advertisements that do not taxonomically match with the query, in order for the more costly functional similarity procedure to be applied on a smaller set of advertisements.

Algorithm 1 The specification matchmaking algorithm

Require: a query advertisement $\mathcal{Q}$, the set $S_{\mathcal{A}}$ of the advertised services, the taxonomical filter set F_T, the Web service filter W_f, the weights $a, b, c \in [0..1]$ and two threshold filters $l_t, l_f \in [0..1]$

Ensure: an array of sets that contains the matched services grouped in each of the 16 grouping filters

1: $matches \leftarrow \varnothing$
2: **for all** $\mathcal{A}_i \in S_{\mathcal{A}}$ **do**
3: $\quad ts \leftarrow TS(\mathcal{A}_i, \mathcal{Q}, F_T)$
4: $\quad$ **if** $ts = 0 \vee ts < l_t$ **then**
5: $\quad\quad$ **continue**
6: $\quad$ **else**
7: $\quad\quad fs \leftarrow FS(\mathcal{A}_i, \mathcal{Q}, W_f)$
8: $\quad\quad$ **if** $fs = 0 \vee fs < l_f$ **then**
9: $\quad\quad\quad$ **continue**
10: $\quad\quad$ **else**
11: $\quad\quad\quad w_f \leftarrow getW_f(\mathcal{A}_i, \mathcal{Q})$
12: $\quad\quad\quad nfs \leftarrow NFS(\mathcal{A}_i, \mathcal{Q})$
13: $\quad\quad$ **end if**
14: $\quad$ **end if**
15: $\quad sim \leftarrow \dfrac{a \cdot ts + b \cdot fs + c \cdot nfs}{a + b + c}$
16: $\quad matches \leftarrow matches \cup \{\langle \mathcal{A}_i, sim, w_f \rangle\}$
17: **end for**
18: $G[16] \leftarrow [\varnothing, \cdots, \varnothing]$ $\triangleright$ An array of 16 sets
19: **for all** $t \leftarrow \langle \mathcal{A}_k, sim, w_f \rangle \in matches$ **do**
20: $\quad y\_w_f \leftarrow getGroupingFilter(\mathcal{Q}, \mathcal{A}_k)$
21: $\quad G[y\_w_f] \leftarrow G[y\_w_f] \cup \{t\}$
22: **end for**
23: **for** $i \leftarrow 1, 16$ **do**
24: $\quad G[i] \leftarrow binarySort_{sim,desc}(G[i])$
25: **end for**
26: **return** G

More specifically, for each advertisement $\mathcal{A}_i$ of the $S_{\mathcal{A}}$ set (line 2), the algorithm computes firstly the TS (line 3). If the TS equals to 0 or it is less than the threshold l_t (line 4), then the $\mathcal{A}_i$ is ignored and the algorithm continues with the next specification. The l_t defines the minimum acceptable similarity between the taxonomical concepts, allowing to incorporate different degrees of relaxation. The algorithm continues by computing the FS (line 7) using the Web service filter W_f that is given as input to the algorithm. Similarly to the TS, if the functional similarity equals to 0 or it is less than the threshold l_f (line 8), the algorithm continues with the next specification. The l_f threshold has similar role to the l_t threshold and defines the minimum acceptable functional similarity. If the computed FS value is acceptable then the algorithm retrieves the Web service filter w_f that the $\mathcal{Q}$ and $\mathcal{A}$ specifications satisfy (line 11). The algorithm continues with the computation of the NFS (line 12) and finally, the $\overline{sim}$ value is computed (line 15) and the result is added to the set $matches$ of the matched specifications as a triple of the matched specification, the sim value and the Web service filter w_f (line 16).

The matched advertisements are returned according to the grouping filter that satisfy (line 20). More specifically, each triple of the $matches$ set is added to the G array (lines 20 and 21) that contains 16 sets, one set for each of the 16 grouping filters we have described in section 4.2.1 (xW_e, xiW_e, etc.). Finally, each set is ordered by the sim value of the triples (line 24) and G is returned.

Example. We exemplify on Algorithm 1, using the EC distance, setting $a = 0.8$, $b = 1$ and $c = 0.1$, and ignoring the l_t and l_f parameters for simplicity. If we select $F_T = \{e\}$ and $W_f = W_e$, then none of the advertisements will be matched, since the $\mathcal{A}_2$ and $\mathcal{A}_3$ specifications will be pruned during the computation of the FS (*Book* $\overset{si}{\sim}$ *Magazine*) and the $\mathcal{A}_4$ specification will be pruned during the computation of the TS (*Search* $\overset{si}{\sim}$ *Order*). By setting $W_f = W_{si}$, we are able to retrieve the $\mathcal{A}_2$ and $\mathcal{A}_3$ specifications with $sim_{\mathcal{A}_2} = \frac{a \cdot 1 + b \cdot 0.774 + c \cdot 1}{a+b+c} = 0.881$ and $sim_{\mathcal{A}_3} = \frac{a \cdot 1 + b \cdot 0.774 + c \cdot 0}{a+b+c} = 0.828$ (see the examples of sections 4.1.1, 4.2.1 and 4.2.2 for the computation of the similarity values). Furthermore, both specifications satisfy the xW_{si} grouping filter (same number of I/Os to $\mathcal{Q}$), and therefore, $G[xW_{si}] = \{\langle \mathcal{A}_2, 0.881, xW_{si} \rangle, \langle \mathcal{A}_3, 0.828, xW_{si} \rangle\}$.

Moreover, by setting $F_T = \{e, si\}$, we are able to retrieve also the $\mathcal{A}_4$ specification with $sim_{\mathcal{A}_4} = \frac{a \cdot \frac{1}{5} + b \cdot 1 + c \cdot 0}{a+b+c} = 0.61$ and $G[xoW_e] = \{\langle \mathcal{A}_4, 0.61, xoW_e \rangle\}$. Note that, even if the sim value of $\mathcal{A}_4$ is less than the sim values of the other two specifications, $\mathcal{A}_4$ will be returned as a more relevant match, followed by $\mathcal{A}_2$ and $\mathcal{A}_3$, since the order of relaxation defines that $xoW_e < xW_{si}$ (see section 4.2.1). In other words, the sim

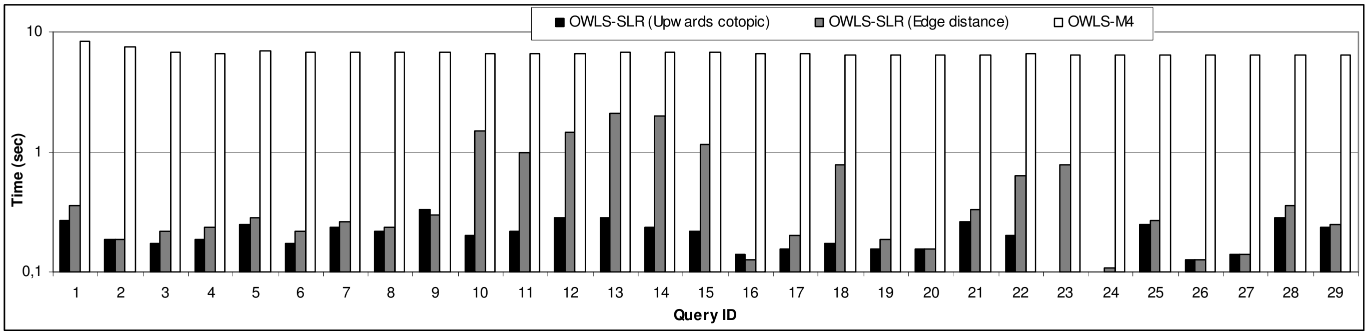

Fig. 2. Query response times

values are used to order the triples of the same grouping set $G[i]$, whereas the total ordering of the matches are defined based on the relaxation of the 16 grouping filters. We argue that such an approach results in a better Top-k precision (section 5.2), than of following a total ordering based on the sim values.

5 Experimental Results

We tested OWLS-SLR and compared it to the OWLS-MX [8] matchmaker (v1.1c), using the OWLS-TC version 2.2 revision 2 collection [18] with 1007 OWL-S advertisements and 29 queries. We have chosen OWLS-MX since it is a well-known matchmaker, having been extensively tested on the OWLS-TC collection. Furthermore, it uses lightweight OWL-S SP descriptions like OWLS-SLR, it is able to incorporate the structural information of the domain ontologies through concept unfolding (*nearest neighbors - NN*), and it is defined upon Pellet, the same reasoner we also use. We used the M4 configuration of OWLS-MX as the best configuration according to [8]. The OWLS-SLR configuration involved $F_T = \{e\}$, $a = 0.8$, $b = 1$, $c = 0.1$ and $l_t = l_f = 0.5$. We have chosen this configuration as the best one, after a number of experiments on the collection. The experiments ran on a Windows XP PC with 3.2 GHz processor, setting maximum JAVA heap size of 800 Mbytes.

5.1 Loading and Query Response Time

The loading time involves the time needed to parse and process the advertisements and queries, whereas the query time involves the time needed to apply the matchmaking algorithm. OWLS-SLR depicts a considerably better loading and query response performance compared to OWLS-M4. OWLS-SLR loaded the dataset in about 30 seconds, whereas OWLS-M4 needed more than 30 minutes. Fig. 2 depicts the query response times of OWLS-SLR and OWLS-M4. In OWLS-SLR, the UC-related distance is computed faster than the EC, since the latter requires the traverse of all the paths between the concepts. However, both configurations perform faster than OWLS-M4 that depicts a constant query response performance.

5.2 Precision and Recall

We used the relevance sets of the collection in order to perform precision and recall tests. Due to the fact that the collection defines only direct Profile instances, we created also a taxonomy-based collection in order to test the performance taking into account the TS. Note that OWLS-MX can handle only direct Profile instances. Fig. 3 depicts the average precision and Fig. 4 the average recall of all queries for OWLS-M4 and OWLS-SLR, according to the Web service filter that was used. We have omitted the *subsumed-by* filter of OWLS-MX for presentation purposes.

OWLS-M4 has in general better precision than OWLS-

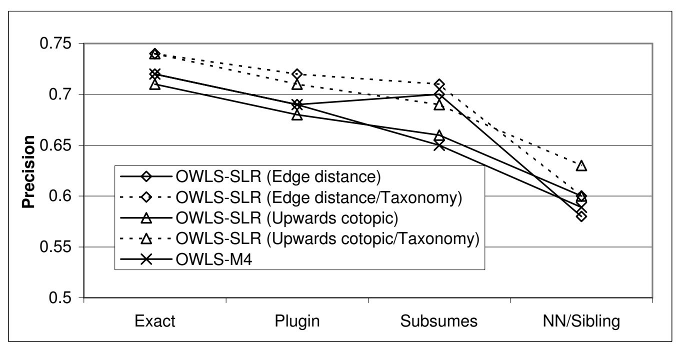

Fig. 3. Average precision.

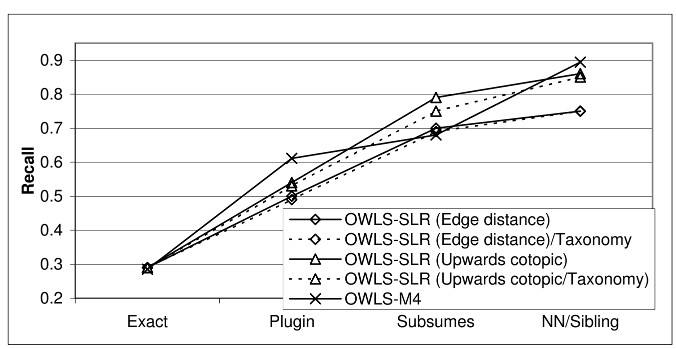

Fig. 4. Average recall.

SLR in the collection without a Profile taxonomy, showing that the filter definitions that it follows, which are based on [12], fit better to the specific collection. However, by performing domain-oriented discovery, OWLS-SLR outperforms OWLS-M4, justifying the advantage of a domain-oriented approach to SWS discovery. The recall of OWLS-SLR using the Profile taxonomy is a little bit lower than the one without taxonomy, since some results do not pass the $F_T = \{e\}$ filter set.

In most cases, we are interested in the first k results of a query. Fig. 5 depicts the average precision of all queries for OWLS-SLR and OWLS-M4 at the Top-k places. The precision at Top-k for a query q is computed as

$$\frac{Relevant_q \cap Returned_{q,k}}{Returned_{q,k}}$$

where $Relevant_q$ is the relevance set of q, and $Returned_{q,k}$ is the Top-k results of the returned advertisements. The experiments have shown that OWLS-SLR has a considerably better precision than OWLS-M4 on the results that are returned first.

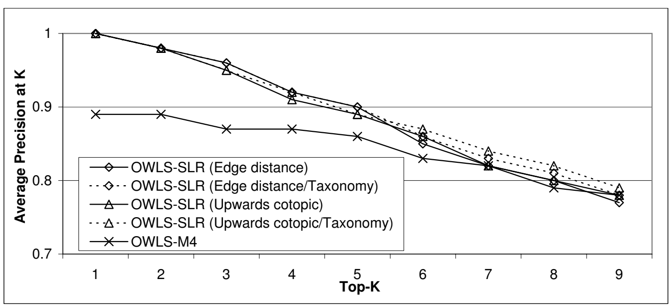

Fig. 5. The average precision at Top-k places.

5.2.1 Scalability

In order to test the scalability of OWLS-SLR, we generated synthetic datasets by altering the base URI of advertisements. Fig. 6 depicts the scalability of OWLS-SLR in terms of loading and Fig. 7 depicts the scalability in terms of query response time. For the legibility of the presentation, Fig. 7 depicts only the query response times up to 4028 advertisements, using the UC-related distance. OWLS-SLR scales very well (almost linearly) both on loading and query response time.

5.2.2 Discussion

We believe that the approach of OWLS-MX to maintain and modify a local ontology imposes an extra overhead on the loading performance. OWLS-SLR loads directly the Profile instances as well as the domain ontologies into the reasoner and therefore, any loading overhead is only related to the capabilities of the underlying reasoner. Furthermore, OWLS-MX performs a concept unfolding for determining concept similarities in the case of nearest neighbor (sibling) matches, generating vectors on which the IR techniques are applied. The determination of the sibling relationships in OWLS-SLR is performed directly on the reasoner's graph that seems to be a more efficient and scalable approach.

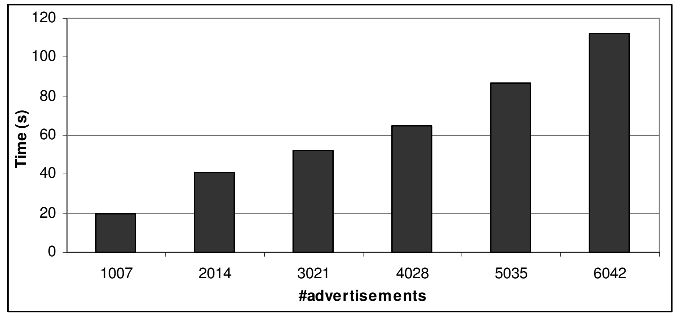

Fig. 6. OWLS-SLR scalability in terms of loading time.

Regarding precision and recall, the strong point of OWLS-SLR is that it allows the existence of a Profile taxonomy in contrast to OWLS-MX that handles only direct Profile instances. To demonstrate the effectiveness of the TS in matchmaking, we present an example taken from OWLS-TC. *EBookOrder1* is a query of the collection with the object specification $\langle q$, {*Economy*}, {*Title*, *User*}, {*Book*},{}$\rangle$. Without taking into account the Profile taxonomy, the *EBookOrder1* query matches the *BookFinder* advertisement, which is defined as $\langle$*adv*, {*Education*}, {*Title*}, {*Book*},{}$\rangle$, in both OWLS-SLR and OWLS-MX. However, these two specifications belong to different domains (*Economy* and *Education*) and the relevance set of *EBookOrder1* does not contain the *BookFinder* advertisement. By considering the Profile taxonomy in OWLS-SLR, the *BookFinder* advertisement is not returned in our experiments, since $F_T = \{e\}$.

Furthermore, OWLS-SLR orders the results based on grouping filters (section 4.2.1). OWLS-MX does not perform grouping, returning the results based on a total similarity ordering. In that way, OWLS-SLR has better Top-k precision since some matches are considered more relevant to some others, even if they satisfy the same Web service filter.

6 Role-driven Web Service Discovery

The OWL-S SP paradigm uses predefined ontology concepts to annotate the Web service I/O parameters. However, it is impractical and not realistic to assume that there will always be an ontology concept suitable for our needs. The ontology roles are also important modeling constructs that encapsulate domain knowledge. Bear in mind that the CC approach makes extensive use of ontology roles. We exploit the classification capabilities of DL reasoning, and we enhance our framework with the ability to perform SWS discovery using also ontology roles as annotation constructs.

6.1 Role-oriented Annotation Concepts

We introduce the notion of the *Role-oriented Annotation Concept (RAC)*, a specially defined concept that derives from cardinality restrictions on ontology roles.

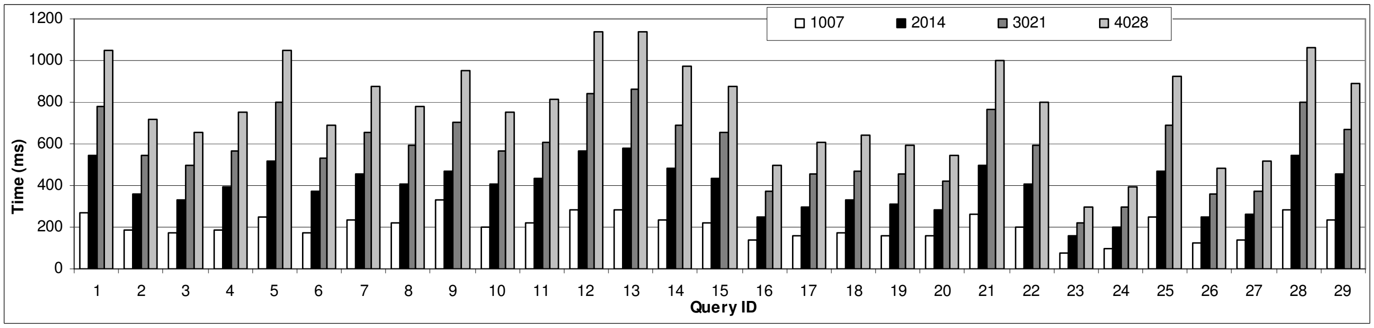

Fig. 7. Query response times according to the number of the loaded advertisements.

Definition 12. *Let R be a set of ontology roles. The Role-oriented Annotation Concept for the set R is the equivalent concept to the intersection of minimum cardinality restrictions of the form*

$$RAC_R \equiv\ \geq 1 r_i.\top \sqcap \cdots \sqcap \geq 1 r_n.\top, \forall r_i \in R. \quad (9)$$

We extend (9) in order to incorporate named classes as

$$RAC_R \equiv\ C_1 \sqcap \cdots \sqcap C_m \sqcap \geq 1 r_i.\top \sqcap \cdots \sqcap \geq 1 r_n.\top, \quad (10)$$

where C_m are domain concepts. Practically, a RAC is an ontology concept that is defined using roles and concepts. In fact, our RAC-based approach tries to incorporate the CC modeling capabilities into the SP paradigm. However, instead of considering a Web service as a whole, we give the opportunity to treat a particular input or output parameter as a whole.

6.2 Example and Experiments

The RAC-oriented discovery requires the runtime classification of RACs in domain ontologies. To this end, we enhanced OWLS-SLR with the ability of altering at runtime the domain ontologies. However, all the query RACs are removed from the domain ontologies after the application of the matchmaking algorithm, preserving the initial ontology structure.

We extended the OWL-S Process concept so as to allow OWLS-SLR to handle Web service Profile instances that contain either RACs or ordinary ontology concepts. More specifically, we defined the $rac{:}Input$ and $rac{:}Output$ concepts as specializations to the $process{:}Input$ and $process{:}Output$ concepts. We defined also the roles $rac{:}concept$ and $rac{:}minCardinalityRestriction$ in order to enable the definition of C_m's and role restrictions in (10).

We exemplify on the way ontology roles can be used as annotation constructs, using the (input) RAC-oriented advertisements of Table 2 that describe Web services that return the price of books. For simplicity, all the outputs have been annotated with the *Price* concept. In $\mathcal{A}_1$, the input is annotated using the equivalent RAC to the *Book* concept. This is similar to the SP paradigm, using directly an ontology concept. In $\mathcal{A}_2$, the input is annotated with a RAC based on the role *title*, and $\mathcal{A}_3$ defines a RAC using the role *isbn*. Finally, $\mathcal{A}_4$ defines an input RAC using both the *title* and the *isbn* roles.

TABLE 2
RAC-oriented Specifications

Domain Ontology Axioms
$Book \sqsubseteq title.\top \sqcap isbn.\top, BookService \sqsubseteq \top, Price \sqsubseteq \top$
$\mathcal{A}$ and $\mathcal{Q}$ Specifications
$\mathcal{A}_1 = \langle adv_1, \{BookService\}, \{RAC_A\}, \{Price\}, \{\}\rangle$ $\mathcal{A}_2 = \langle adv_2, \{BookService\}, \{RAC_B\}, \{Price\}, \{\}\rangle$ $\mathcal{A}_3 = \langle adv_3, \{BookService\}, \{RAC_C\}, \{Price\}, \{\}\rangle$ $\mathcal{A}_4 = \langle adv_4, \{BookService\}, \{RAC_D\}, \{Price\}, \{\}\rangle$ $\mathcal{Q} = \langle q, \{BookService\}, \{RAC_E\}, \{Price\}, \{\}\rangle$
RAC Definitions
$RAC_A \equiv Book, RAC_B \equiv \geq 1title.\top, RAC_C \equiv \geq 1isbn.\top,$ $RAC_D \equiv \geq 1title.\top \sqcap \geq 1isbn.\top,\ RAC_E \equiv RAC_B$

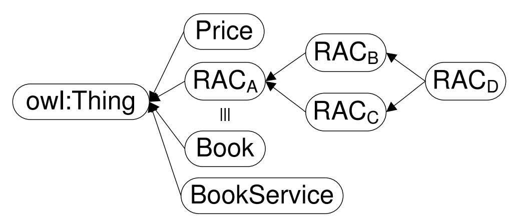

Fig. 9. The TBox relationships among the RACs.

By classifying the RACs into the domain ontology of Table 2, we obtain the TBox relationships of Fig. 9.

Let the $\mathcal{Q}$ specification of Table 2. By classifying RAC_E in the domain ontology, we get that $RAC_E \equiv RAC_B$. Therefore, if we use the W_e filter during matchmaking, only the $\mathcal{A}_2$ specification will be returned, since it matches exactly all the I/Os of $\mathcal{Q}$. By relaxing the Web service filter, we retrieve the other three $\mathcal{A}$ specifications, as well. More specifically, the W_p filter returns $\mathcal{A}_4$, since $RAC_D \overset{p}{\sim} RAC_E$, the W_{su} filter returns $\mathcal{A}_1$, since $RAC_A \overset{su}{\sim} RAC_E$, and the W_{si} filter returns $\mathcal{A}_3$, since $RAC_A \overset{si}{\sim} RAC_E$. In that way, ontology roles can be used as annotation concepts in matchmaking.

In order to test the performance of the RAC-based SWS matchmaking, we used the OWLS-TC collection

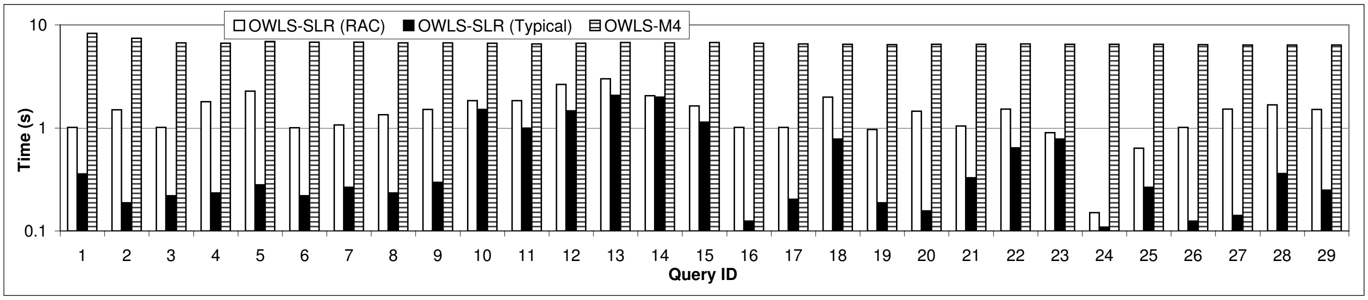

Fig. 8. The RAC-based against the typical query response time of OWLS-SLR and OWLS-MX.

and for each I/O query concept C we defined the equivalent RAC, that is, $RAC \equiv C$, in a similar way to the $\mathcal{A}_1$ RAC-oriented advertisement of Table 2. In that way, we were able to use the same relevance set to the initial collection, as well as to test the performance of the runtime concept classification.

As far as precision and recall are concerned, the experiments resulted in the same performance that we achieved in OWLS-SLR using the typical query collection (Fig. 3 and 4). This fact justifies the soundness of the RAC-based implementation of OWLS-SLR.

Regarding query response time, OWLS-SLR requires more time to answer the RAC-based queries than in the typical collection, as it is depicted in Fig. 8. This happens since the computation of the FS similarity involves also the time Pellet requires to classify and delete the RAC concepts, in contrast to the typical OWLS-SLR functionality, where the TBox reasoning is performed in advance. However, we believe that the role-driven SWS discovery offers more capabilities than the typical OWL-S SP-oriented paradigm, allowing both concepts and roles to be considered in the annotation process. It is worth mentioning that the RAC-based query answering performance of OWLS-SLR is better than of OWLS-MX.

7 Related Work

OWL-S [5], SAWSDL [19], WSDL-S [20] and WSMO [9] constitute the major standards for semantic Web service annotation. Apart from the Light approach we described in section 2.1.1, WSMO-DF allows the definition of descriptions with higher level of detail. The WSMO *capability* element is able to encapsulate goals, mediators, preconditions and assumptions. It is argued that a similar level of detail can also be used in OWL-S [21], [22] through, for example, the Process model [23] or preconditions and effects [24], [25]. A lightweight approach, such as ours, can be used as an initial step to retrieve a set of candidate Web services on where more complex and sophisticated algorithms can be applied.

In [26], Web services are annotated in terms of state transitions. Actually, the Rich Web service representation of WSMO-DF is followed, using an environmental ontology. In contrast to [26], we follow the SP model, exploiting the structural ontology information.

IRS-III [27] extends the WSMO conceptual model and uses the OCML language for internal representation and an OCML reasoner. In contrast to our framework, IRS-III follows the Rich WSMO model and a frame-based rule language for representing ontologies.

A search engine for grid service discovery is presented in [28], using the Rough sets theory. The novelty is in its capability to deal with uncertain properties, that is, properties that are explicitly used by one advertisement but do not appear in another service of the same category. In our framework, we examine only the common properties of an advertisement and query.

In [29], an approach based on context-aware ratings and context-aware experiences is proposed in order to select services. Consumers use ontologies to express the context of their interactions with service providers as a whole, instead of using the SP. It targets mainly at rating environments without exploiting structural knowledge, but it may be used in Web service discovery.

In [30], Web service descriptions are defined as CCs in OWL and the matchmaking procedure examines the subsumption relationships. FC-MATCH [31] follows the same approach, performing also text similarity matching using WordNet. In [32], a framework for annotating Web services using DLs is presented. Similar to ours, it follows the abstract Web service model. However, it treats Web services as a whole.

In the DAML-S/UDDI Matchmaker [33], OWL-S SP advertisements and requests refer to DAML concepts and the matching process performs inferences on the subsumption hierarchy. It uses a different definition of Web service filters from ours and it does not consider Profile taxonomies, roles or grouping filtering.

LARKS [34] uses both syntactic and semantic matching. It uses five matchmaking filters, namely context matching, profile comparison, similarity matching, signature matching and constant matching. LARKS uses its own capability description and DL language in contrast to our OWL-based approach.

OWLS-MX [8] utilizes both logic-based reasoning and content-based IR techniques for Web services in OWL-S. As we have already mentioned, it cannot handle Profile taxonomies and it follows the static SP paradigm, unable to use dynamically ontology roles. iMatcher2 [35] follows the OWLS-MX approach, applying also learning algorithms in order to predict similarities. Like OWLS-MX, it uses a DL reasoner in order to unfold the annotation concepts, creating a vector on which the IR techniques are applied. iMatcher2 does not follow a standard matchmaking algorithm, which is defined through an iSPARQL strategy. WSMO-MX [36] is a hybrid approach based on Rich WSMO service descriptions.

There are plenty of other approaches that are based on I/Os, for example [37], [38], [25]. To the best of our knowledge, these approaches do not perform ABox reasoning on Profile instances in order to exploit the domain knowledge of Profile instances. Instead, they retrieve directly the I/O annotations and any taxonomical knowledge stems from special properties, such as service categorization. Furthermore, they do not consider roles, using static annotation concepts, and do not apply further filtering on results (grouping filtering). METEOR-S [39] follows the WSDL-S approach, where WSDL constructs point to ontology concepts.

Our work has been motivated by a previous work of ours [40] that implements Object-Oriented similarity measures for SWS discovery, using a production rule engine [41]. In the present work, (a) we use a DL reasoner to handle SWS descriptions and to apply an extended matchmaking algorithm, (b) we define two new OWL-S Profile-aware and DL-based similarity measures and (c) we propose a framework for the incorporation of ontology roles in the SP SWS discovery paradigm.

8 Conclusions and Future Work

In this paper, we presented an OWL-S SP-aware framework for SWS discovery, using abstract and lightweight Web service descriptions. Our intention is to define a framework that can be used as a prephase in more fine-grained approaches that incorporate rich Web service descriptions, such as preconditions, effects or state transitions. In that way, the complex and sophisticated algorithms would be applied on a smaller set of candidate Web service descriptions than the complete initial set.

In an effort to enhance the instance-oriented SP discovery paradigm with the domain modeling capabilities found on CC approaches, (a) we allow the existence of Profile taxonomies, and (b) we enable the annotation of I/Os with ontology roles. Moreover, we defined a matchmaking algorithm that exploits the structural knowledge of ontologies, for example, sibling concept relationships, by considering advertisements and requests as objects and by implementing concept (dis-) similarity measures.

We presented also a comparison of OWLS-SLR to OWLS-MX. The experiments have shown a considerably better performance on loading and querying of OWLS-SLR than of OWLS-MX. Furthermore, we were able to increase precision and recall using grouping filters (Top-k experiments) and performing taxonomy-based discovery. The results seem very promising, as far as the requirement for a fast filtering of Web service advertisements is concerned.

OWLS-SLR is available at [42], together with the experimental collections we have used. For the future, we plan to enhance our framework with more structural [43] and information-content similarity measures [44], [45], [46]. Currently, we are working on enhancing our framework with composition capabilities in order to return not only single Web services, but also Web service compositions [47] based on abstract descriptions.

Acknowledgments

This work was partially supported by a PENED program (EPAN M.8.3.1, No. 03Δ73), jointly funded by EU and the General Secretariat of Research and Technology.

References

[1] *WSDL 1.1*, http://www.w3.org/TR/wsdl, 2001.
[2] M. Burstein, C. Bussler, M. Zaremba, T. Finin, M. N. Huhns, M. Paolucci, A. P. Sheth, and S. Williams, "A Semantic Web Services Architecture," *IEEE Internet Comput.*, vol. 9, no. 5, pp. 72–81, 2005.
[3] C. Preist, "A Conceptual Architecture for Semantic Web Services," in *Int'l Semantic Web Conf.*, 2004, pp. 395–409.
[4] U. Keller, R. Lara, H. Lausen, and D. Fensel, *Semantic Web Service Discovery in the WSMO Framework*. Idea Publishing Group, 2006.
[5] D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D. L. Mcguinness, E. Sirin, and N. Srinivasan, "Bringing Semantics to Web Services with OWL-S," *World Wide Web*, vol. 10, no. 3, pp. 243–277, September 2007.
[6] R. Bergmann and M. Schaaf, "Structural Case-Based Reasoning and Ontology-Based Knowledge Management: A Perfect Match?" *J. Universal Computer Science (UCS)*, vol. 9, no. 7, pp. 608–626, 2003.
[7] F. Baader, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.
[8] M. Klusch, B. Fries, and K. Sycara, ""OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services"," *J. Web Sem.*, vol. In Press, Corrected Proof, 2008.
[9] D. Fensel, H. Lausen, A. Polleres, J. D. Bruijn, M. Stollberg, D. Roman, and J. Domingue, Eds., *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag, 2006.
[10] *OWL*, http://www.w3.org/2004/OWL/, 2004.
[11] *OWL-S 1.1 Release: Examples*, http://www.daml.org/services/owl-s/1.1/examples.html, 2004.
[12] A. M. Zaremski and J. M. Wing, "Specification Matching of Software Components," in *3rd ACM SIGSOFT Symposium on Foundations of Softw. Eng.*, 1995, pp. 6–17.
[13] *UDDI*, www.oasis-open.org/committees/uddi-spec, 2005.
[14] N. Srinivasan, M. Paolucci, and K. P. Sycara, "An Efficient Algorithm for OWL-S Based Semantic Search in UDDI," in *Semantic Web Services and Web Process Composition*, 2004, pp. 96–110.
[15] A. Maedche and V. Zacharias, "Clustering Ontology-Based Metadata in the Semantic Web," in *European Conf. Principles of Data Mining and Knowledge Discovery*, London, 2002, pp. 348–360.
[16] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A Practical OWL-DL Reasoner," *J. Web Sem.*, vol. 5, no. 2, pp. 51–53, 2007.
[17] W. Winkler, "The State of Record Linkage and Current Research Problems," in *Survey Methods Section, Statistical Society of Canada*, 1999, pp. 73–80.
[18] *OWLS-TC version 2.2 revision 2*, http://projects.semwebcentral.org/projects/owls-tc/, 2008.
[19] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Comput.*, vol. 11, no. 6, pp. 60–67, 2007.
[20] *WSDL-S*, http://www.w3.org/Submission/WSDL-S/, 2005.
[21] M. Paolucci, N. Srinivasan, and K. Sycara, "Expressing WSMO Mediators in OWL-S," in *Semantic Web Services: Preparing to Meet the World of Business Applications, 3rd Int'l Semantic Web Conf.*, 2004.
[22] R. Lara, D. Roman, A. Polleres, and D. Fensel, "A Conceptual Comparison of WSMO and OWL-S," in *European Conf. Web Services*, 2004, pp. 254–269.

[23] M. Klein and A. Bernstein, "Toward High-Precision Service Retrieval," *IEEE Internet Comput.*, vol. 8, no. 1, pp. 30–36, 2004.

[24] M. Klein and B. König-Ries, "Coupled Signature and Specification Matching for Automatic Service Binding," in *European Conf. Web Servcices*, 2004, pp. 183–197.

[25] D. Skoutas, A. Simitsis, and T. Sellis, "A Ranking Mechanism for Semantic Web Service Discovery," in *IEEE Congress on Services*, 2007, pp. 41–48.

[26] P. Wang, Z. Jin, L. Liu, and G. Cai, "Building Toward Capability Specifications of Web Services Based on an Environment Ontology," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 4, pp. 547–561, 2008.

[27] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci, "IRS-III: A Broker-based Approach to Semantic Web Services," *J. Web Sem.*, vol. 6, no. 2, pp. 109–132, 2008.

[28] M. Li, B. Yu, O. F. Rana, and Z. Wang, "Grid Service Discovery with Rough Sets," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 851–862, 2008.

[29] M. Şensoy and P. Yolum, "Ontology-Based Service Representation and Selection," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1102–1115, 2007.

[30] L. Li and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology," in *Int'l Conf. World Wide Web*, 2003, pp. 331–339.

[31] D. Bianchini, V. D. Antonellis, M. Melchiori, and D. Salvi, "Semantic-Enriched Service Discovery," in *Int'l Conf. Data Eng. Workshops*, 2006, p. 38.

[32] S. Grimm, B. Motik, and C. Preist, "Matching Semantic Service Descriptions with Local Closed-World Reasoning," in *European Semantic Web Conf.*, 2006, pp. 575–589.

[33] K. P. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated Discovery, Interaction and Composition of Semantic Web Services," *J. Web Sem.*, vol. 1, no. 1, pp. 27–46, 2003.

[34] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 2, pp. 173–203, 2002.

[35] C. Kiefer and A. Bernstein, "The Creation and Evaluation of iSPARQL Strategies for Matchmaking," in *European Semantic Web Conf.*, 2008, pp. 463–477.

[36] F. Kaufer and M. Klusch, "WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker," in *European Conf. Web Servcices*, 2006, pp. 161–170.

[37] J. Cardoso, "Discovering Semantic Web Services with and without a Common Ontology Commitment," in *IEEE Services Comput. Workshops*, 2006, pp. 183–190.

[38] J. Pathak, N. Koul, D. Caragea, and V. G. Honavar, "A Framework for Semantic Web Services Discovery," in *ACM Int'l Workshop on Web Information and Data Management*, 2005, pp. 45–50.

[39] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Inf. Technol. and Management*, vol. 6, no. 1, pp. 17–39, 2005.

[40] G. Meditskos and N. Bassiliades, "Object-Oriented Similarity Measures for Semantic Web Service Matchmaking," in *European Conf. Web Services*, 2007, pp. 57–66.

[41] G. Meditskos and N. Bassiliades, "A Rule-Based Object-Oriented OWL Reasoner," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 3, pp. 397–410, 2008.

[42] *OWLS-SLR*, http://lpis.csd.auth.gr/systems/OWLS-SLR, 2008.

[43] V. Schickel-Zuber and B. Faltings, "OSS: A Semantic Similarity Function based on Hierarchical Ontologies," in *Int'l Joint Conf. of Artificial Intell.*, 2007, pp. 551–556.

[44] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," in *Int'l Joint Conf. of Artificial Intell.*, 1995, pp. 448–453.

[45] D. Lin, "An Information-theoretic Definition of Similarity," in *Int'l Conf. Machine Learning*, 1998, pp. 296–304.

[46] P. M. Schwarz, Y. Deng, and J. E. Rice, "Finding Similar Objects Using a Taxonomy: A Pragmatic Approach," in *Int'l Conf. Databases and Applications of Semantics*, 2006, pp. 1039–1057.

[47] U. Küster, B. König-Ries, M. Stern, and M. Klein, "DIANE: An Integrated Approach to Automated Service Discovery, Matchmaking and Composition," in *Int'l Conf. WWW*, 2007, pp. 1033–1042.

George Meditskos received the BSc degree in computer science in 2004 and the MSc degree in computer science in 2007 from the Aristotle University of Thessaloniki, Greece. Since 2004, he has been a PhD student in computer science at the Aristotle University of Thessaloniki. His research interests include semantic Web, semantic Web services, and artificial intelligence.

Nick Bassiliades received the PhD degree in parallel knowledge base systems in 1998 from the Department of Informatics, Aristotle University of Thessaloniki, Greece. He is currently an assistant professor in the Department of Informatics, Aristotle University of Thessaloniki. His research interests include knowledge-based systems, rule systems, and the semantic Web. He has published more than 100 papers in journals, conferences, and books and coauthored an international book on parallel, object-oriented, and active knowledge-based systems and a Greek book on artificial intelligence. He is a member of the Board of the Greek Artificial Intelligence Society, a director of RuleML Inc., and also a member of the Greek Computer Society, the IEEE, and the ACM.