An Enhanced Framework for Semantic Web Service Discovery

Nadia Yacoubi Ayadi and Mohamed Ben Ahmed

National School of Computer Sciences
University of Manouba, 2010 Tunisia
nadia.yacoubi@asu.edu, mohamed.benahmed@riadi.rnu.tn

Abstract. The paper describes an ontology-based framework for semantic Web services description and discovery. The proposed framework relies on a lightweight and minimal description model allowing to describe services in terms of a set of functional and non-functional properties. Functional properties describe services capabilities in terms of Inputs, Outputs, Preconditions and Effects (IOPE). Non-functional properties describe services in terms of Quality of Service (QoS) properties. We propose an enhanced discovery approach in which Web services are discovered based on their functional and non functional properties. The enhanced discovery approach proposed is twofold. First, the approach relies on deductive relaxation of discovery queries based on semantic service descriptions and domain knowledge (domain ontology) to select a set of services that match a discovery query with different matching degrees. Second, non-fonctional properties are incorporated into the Web service discovery mechanism to generate a partially ordered list of services that meet user functional and non-functional requirements.

Keywords: Semantic Web, Web service discovery, Ontologies, deductive reasoning, logical relaxation, Quality of Service.

1 Introduction

Due to the proliferation and ubiquity of the Internet network, information systems are becoming increasingly distributed in nature. This growing connectivity allows organisations to share different types of resources, to automate and outsource their business processes in order to offer their services to a worldwide audience [1]. The paradigm of service-oriented computing is gaining popularity as an appropriate system engineering approach for enabling distributed, heterogeneous software components to communicate and interoperate through Web services [2]. Existing Web service technologies such as the Web Service Description Language[1] (WSDL), the Simple Object Access Protocol[2] (SOAP), and the Universal Description, Discovery, and Integration[3] (UDDI) have concentrated

[1] http://www.w3.org/TR/wsdl
[2] http://www.w3.org/TR/soap/
[3] http://www.uddi.org/pubs/uddi_v3.htm

M. Snene, J. Ralyté, and J.-H. Morin (Eds.): IESS 2011, LNBIP 82, pp. 53–67, 2011.

on providing an interoperability infrastructure and syntactic service descriptions. Recently research on semantic Web has explored the use of ontologies to enhance service descriptions with formal semantic annotations, such services are namely known as Semantic Web Services (SWS) [3].

The overall aim of this effort is to enable semantic-based reasoning that exploit rich service descriptions to support smooth automation of service discovery and composition. However, due to the inherent complexity required to fully capture computational functionality, creating SWS descriptions has represented an important knowledge acquisition bottleneck and has required the use of rich languages and complex matchmaking algorithms. Semantic frameworks such as OWL-S [4], WSMO (Web service Modeling Ontology) [5] and SAWSDL (Semantic Annotations for WSDL) [6] propose rich semantic descriptions of Web services. On one side, OWL-S and WSMO frameworks offer domain-independent and structured representation of service semantics by means of upper (top-level) service ontologies and languages with formal logic groundings such as OWL and WSML. On the other side, SAWSDL provides a set of semantic annotations to enrich syntactic WSDL service descriptions and comes without any formal semantics. We may notice that these frameworks offer heterogeneous semantic descriptions of Web services because they support different service semantics facets, they use different languages to express service descriptions, and hence they allow different types of reasoning.

In this paper, in order to deal with semantic heterogeneity of semantic Web services descriptions, we propose an RDF-S *canonical description semantic model* comprised by a set of canonical primitives that captures service capabilities (functional properties) and Quality-Of-Service (QoS)-related properties (non-functional properties) of Web services. The proposed model can be seen as a *lightweight and minimal model* of semantic service descriptions that captures the core semantics of a Web service. Semantic frameworks presented above are linked to our canonical description model by a set of semantic mappings. Therefore, we propose an enhanced Web service discovery approach that relies on both functional and non-functional properties to select the most *appropriate* services. Thus, based on the canonical model, our approach allows to discover semantic Web services created under different frameworks.

The paper is structured as follows. Section 2 reviews existing frameworks for semantic Web service description and main approaches for service discovery. Section 3 presents an enhanced semantic Web service description and discovery framework. In section 4, we illustrate our contributions with a real-world case study from the bioinformatics domain. Finally, section 5 concludes and outlines our future work.

2 Related Work

Semantic service discovery is the process of locating existing Web services based on the description of their functional and non-functional semantics. Discovery scenarios typically occur when one is trying to reuse an existing piece of functionality (represented as a Web service) towards building new or enhanced business

processes. In following, we review semantic service description frameworks considered as the most prominent ones in the field of SWS. Also, we briefly describe the two services description languages USDL [7] and BSDL [8] that offer service descriptions from a purely business perspective. We review only functional-aware Web services discovery approaches which consider only functional service profiles in terms of inputs, outputs, preconditions and effects or post-conditions.

2.1 Semantic Web Service (SWS)

Semantic service description refers to an abstract description that makes explicit functional and/or non-functional semantics of a Web service. Functional semantics are captured by a service profile and/or service process model. The service profile describes the service signature in terms of Input (I) and Output (O) parameters, Preconditions (P) and Effects (E) which are conditions supposed to hold before or after executing the service in a given world state. In contrast, non-functional properties allow the description of a Web service in terms of quality attributes, such as performance, reliability, or availability issues. The former parameters are observable at run time, while the latter are embodied in the static structure of the Web service. It is well known that functional and non-functional properties constrain each other, and therefore, they should be treated together.

In order to describe the semantics of a set of available services, different description frameworks have been proposed [6, 4, 5]. Each semantic service description framework can be characterised with respect to the scope and granularity of their conceptual models and also the language used to express service descriptions.

OWL-S is an OWL ontology comprised by three main components: the *service profile* which provide a concise service description required to its publication in a registry; the *service model* which establish how the service works as a set of processes; and the *service grounding* which specify how the service can be accessed. The OWL-S service profile describes a Web service in terms of a set of Inputs, Outputs, Preconditions and Effects (IOPE). Once a service has been selected the profile is useless; rather, the client will use the service model to control the interaction with the service. The OWL-S process model allows to describe services in terms three types of processes: atomic, simple and composite. Composite processes are decomposable into other (non-composite or composite) processes; their decomposition can be specified by using control constructs such as *sequence* and *split*. However, OWL-S does not dictate any constraint between the profile and process models, so the two descriptions may be inconsistent without affecting the validity of the OWL expressions. Moreover, OWL-S does not consider non functional properties of Web services related to the service quality.

On the other side, the Web Service Modeling Ontology (WSMO) [5] describes services from the requester (consumer) and provider point views. A WSMO instance describes a service in terms of the functional description of the service's capabilities, the ontologies used to describe the service, the mediators that can deal with protocol and process related mismatches between web services; and a

set of non-functional properties. In addition, the model supports the definition of client goals that need to be satisfied when a Web Service is invoked and the specification of the mediators that can be used to handle heterogeneity between goals specifications and services descriptions. We notice that goals definition and mediation aspects are not considered in the OWL-S ontology.

Semantic annotations for WSDL (SAWSDL) defines mechanisms to annotate WSDL elements with a set of mappings that establish the meaning of WSDL element via semantic annotation. However, when a WSDL element is annotated with several concepts, it is not possible to differentiate which concept annotates the service category, which one annotates the behavior or which one annotates a precondition or an effect of a service.

The aforementioned frameworks propose rich semantic descriptions of Web services. Table 1 presents a comparison pointing to the syntactic and semantic heterogeneities of these descriptions. It is clear that besides the complexity introduced by these frameworks and the additional effort demanded of users to understand their conceptual foundations, they brought additional semantic heterogeneity to the actual Web.

Table 1. Comparison of OWLS, WSMO and SAWSDL

	OWL-S	**WSMO**	**SAWSDL**
Functionality Description	Service Profile	Capability	Operation
IOPE	hasInput, hasOutput, hasPrecondition, hasResult	Precondition, Postcondition, Assumption, Effect	Input, Output
Service behaviour	Process Model	Interface	Not supported
Non-functional Properties	Profile hierarchy	Quality Of Service	Not supported
(Supported) Ontology Languages	OWL	WSML	Any Ontology Language
Goal Specification	Not Supported	WSMO goal	Not supported

Other languages was proposed to describe services from a purely business perspective. The most prominent ones are the Universal Service Description Language (USDL) [7] and the Business Service Description Language (BSDL) [8]. Cardoso et al. [7] propose USDL as a language for describing business, operational and technical aspects of "*universal*" services, i.e., any type of services independently of their economic area. The business description includes the

formal specification of legal, marketing and bundling aspects. The operational description includes functional and behavioral characteristics, and resource requirements. Finally, the technical description specifies how a service can be invoked and relies on references to Web services protocols. Although, the formal specification of USDL relies on a MOF-based meta-model to describe any type of services, authors do not present any case study to illustrate how USDL services may be discovered.

On the other side, BSDL [8] addresses the description of both functional and non fonctional properties of a business service. BSDL is similar to our canonical model; both of them provide formal semantics based on a logic formalism. Moreover, BSDL proposes to apply an hierarchical decomposition or refinement of business services in order to create a service network. In our opinion, such decomposition does not reflect reality of business, different patterns of service networks exist including hierachical pattern. Even if BSDL covers a wider range of non functional properties than our canonical model, it does not perform any domain based reasoning to exploit domain knowledge, reasoning in BSDL is only restricted to decomposition rules.

2.2 Service Matching and Discovery

Service matching and discovery algorithms can be classified according to their search and reasoning capabilities. On the one hand, Web service search engines such as seekda![4] provide access to a significantly higher number of WSDL files for a given keyword-based user request and relies on either keyword search or category browsing. However, Web service search engines are not well suitable for Web service discovery because services are selected in terms of keyword-based searches.

The majority of service discovery approaches use Request Functional Profiles that represent user requirements in terms of functional properties, and performs deductive-based semantic service matching. Paolucci et al. [9] propose a logic-based service discovery approach that relies on the information published in the OWL-S Service Profile to identify matches between the user's query and a service profile. In [10], authors describe an ontology-based framework for the discovery of semantically heterogeneous Web services. The approach relies on user-supplied, and context-specific mappings from user ontologies to relevant domain ontologies used to specify Web services. User-specified functional requirements are considered during the discovery process and a user-specified criteria is used to rank the discovered services. OWLS-MX [11] is a hybrid approach that complements logic-based reasoning with approximate matching based on syntactic-based similarity computations. OWLS-MX uses the OWL-DL description logic reasoner Pellet for logic based filtering, and the cosine, loss-of-information, extended Jacquard, and Jensen-Shannon information divergence based similarity metrics, for complementary approximate matching. However, existing approaches for semantic Web services discovery do not provide a generic solution; mostly, because

[4] `http://webservices.seekda.com/`

they consider one specific semantic description format. Therefore, user requirement are expressed in terms of functional OR non-functional constraints and not both of them. Moreover, reasoning tasks performed by existing approaches do not include domain constraints reasoning and are only based on hierarchy reasoning.

3 An Enhanced Semantic Service Discovery Approach

In this paper, we propose an enhanced discovery approach that combines deductive-based techniques and ranking techniques to select a set of candidate services with different degrees of matching and classify them in terms of their non-functional properties. The main contributions of this paper are summarized as follows:

- In order to deal with the heterogeneity of semantic Web services descriptions, we propose a canonical semantic description model of Web services as a *core model* for service semantics.
- A Logic-based reasoning approach that enables flexible matchmaking between Web services descriptions and discovery queries considering the background knowledge domain.
- An enhanced service discovery approach comprised by two steps. First, candidate services, those satisfying functional user requirements, are selected based on relaxation techniques. Second, candidate services are ranked with respect to their non-functional properties values in order to generate a partial ordered set of Web services.

3.1 Canonical Semantic Service Description Model

Figure 1 illustrates the canonical semantic service description model which is a simple RDF-S integration model conceived based on the principle of minimal ontological commitment [12] to capture the core semantics of a Web service. We notice that the canonical model does not aim to be yet another service model which bring further heterogeneity to the SWS landscape; rather the different semantic frameworks for Web services are mapped to the proposed model [13]. The semantic model offers a set of lightweight semantic annotations that describe functional and non functional properties.

A *Service* is declared as an RDF triple as follows (s, `rdf:type`, isService). Each service s has a possibly empty set of *inputs* declared by a set of RDF triples, (c, `hasInput`, $inputName$) and a non empty set of *outputs* defined by a set of RDF triples (c, `hasOutput`, $outputName$). Also, a service s has a possibly empty set of *preconditions* and *effects*, and an optional *Category*. The inputs and outputs of a service describe respectively the parameters required for the execution of the service and those generated after the execution of the service. A service is also characterized by its category, which is the description of the functionality provided. Each input, output and category is defined with a name (label) and is annotated in terms of a set of domain ontologies concepts.

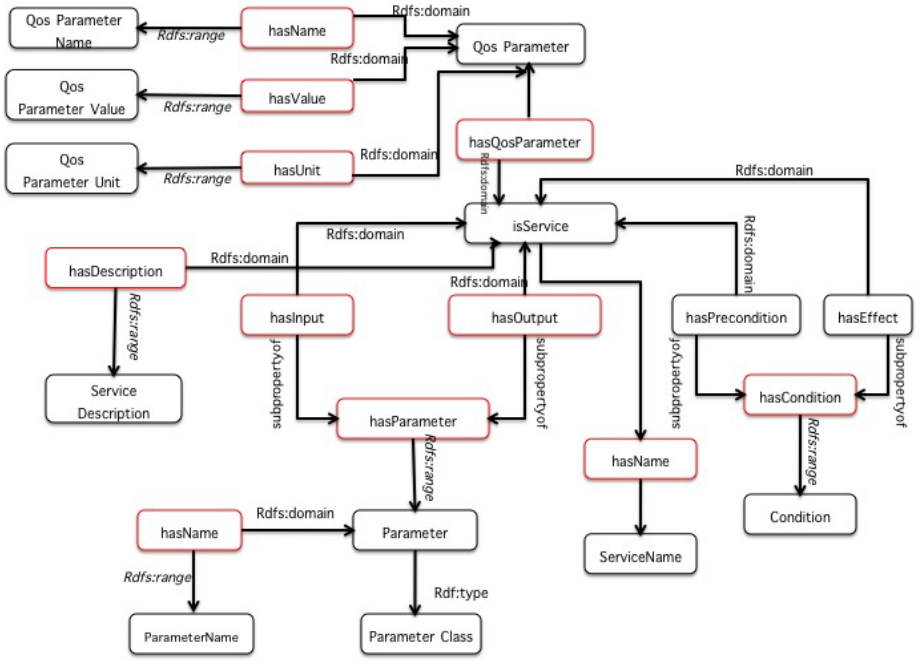

Fig. 1. Canonical Semantic Service Description Model

From the non-functional view, the concept of *service* is also described by a possibly empty set of non-functional properties, namely Quality-of-Service (QoS) properties. In our model, each QoS property is defined by a triple (p, v, u) where p is QoS parameter name, v is a value's parameter and u is the unit in which a value v is expressed. QoS properties are defined by a set of RDF triples as follows (s, `hasQoSproperty`, p). We adopt the non-functional properties set, proposed by [14], that includes metrics to evaluate the Web service *performance*, *dependability* and *reputation* reflecting the consumer perception towards the service. The performance of the Web service is measured by metrics such as *throughtput*, *response time*, *execution time* and *resource utilization*. The dependability of a web service integrates different attributes including *reliability*, *availibitity* and *security*.

Given a set of available services described as above, a discovery query is defined as a set of functional and non-functional requirements expressing constraints in terms capability descriptions and QoS parameters values. We formalize discovery queries as follows.

Definition 1. *An Extended Discovery Query Q is a tuple (F, NF), where F represents a set of functional requirements and NF represents a set of non-functional requirements.*

- *Functional requirements F are represented by a pair (I,O), where I is a set of inputs and O is a set of outputs that need to be produced as the result of the query Q.*
- *Non-functional requirements NF are represented by a set of 4-tuples (P, C, V, U), where, P is a QoS parameter, C is a comparator; V is a value and U is an unit; the evaluation of the query Q has to respect the conjunction of the conditions expressed in the set NF. If (P, C_i, V_i, U_i) and (P, C_j, V_j, U_j) $\in$ NF, then $C_i = C_j$, $V_i = V_j$ and $U_i = U_j$.*

We adopt SPARQL query language [15] to express extended discovery queries as illustrated by the following example.

Example 1. Suppose that a client is looking for a shipping service to deliver goods from Geneve in Switzerland to Hannover in Germany. The following SPARQL discovery query specifies the properties of the package to deliver: package weight, expedition and delivery dates, expedition and delivery cities and a set of non-functional constraints as follows:

```
SELECT      ?service, ?capability
{
 WHERE
?service          hasCapability          ?capability;
                  hasQoSProperty         QoS1;
                  hasQoSProperty         QoS2.
?capability       hasInput               I1.
I1                rdfs:label             "PackageName1";
                  _:hasPackageWeight     30;
                  _:hasExpeditionCity    "Geneve";
                  _:hasDeliveryCity      "Hannover";
                  _:hasDeliveryDate      '02-02-2011'.
QoS1              rdfs:label             "Reputation";
QoS2              rdfs:label             "Price";
FILTER
(           QoS1 > 90   &&        QoS2 < 15)
}
```

In this paper, a two-fold approach is proposed to select the services satisfying an extended discovery query. The first phase allows to determine the most appropriate services that meet functional and non functional requirements of a discovery query. In this phase, a deductive-based approach is used; the search space of a discovery query is enlarged by relaxing the functional constraints of the query. In the second phase, a ranking process is performed to generate a partial ordered set of Web services based on QoS properties and user preferences. In the next two sections, we present relaxation techniques used in the selection phase and the ranking process of our enhanced discovery approach.

3.2 Discovery Query Relaxation

The query relaxation step corresponds to the rewriting process of a given discovery query Q based on a set of logical axioms. We define different types of relaxation depending on the types of knowledge used. Thus, we distinguish *type relaxation* which is an instance based reasoning, *hierarchy relaxation* that exploits concept taxonomy and *predicate relaxation* which exploits semantic relations between concepts in a domain ontology. We explain in following the different types of query relaxation.

– **Type relaxation** allows to infer a new triple (a, *rdfType*, c) from the triple (a, *rdfCollection*, c). Based on type relaxation, we define the following two axioms in order to relax some discovery query conditions:

$$\frac{(S,hasInput,I)(I,rdfCollection,C_1)}{(S,hasInput,C_1)} \quad (A_1)$$

$$\frac{(S,hasOutput,O)(O,rdfCollection,C_1)}{(S,hasOutput,C_1)} \quad (A_2)$$

– **Hierarchy relaxation** considers all classes belonging to the tansitive closure of a concept C in a domain ontology $\mathcal{O}$. We define two hierarchy relaxation axioms:

- **Input specialisation** asserts that if a concept A is declared as an input concept of a requested capability C then all sub-classes of A are asserted as relaxed functional constraints of the query Q. For example, given a query Q containing the triples (?c, *hasInput*, I_1) and (I_1, *rdftype*, `Book`); if the concept `Novel` is defined as a sub-concept of the concept `Book` in a domain ontology, then we infer a new triple (?c, *hasInput*, `Novel`).

$$\frac{(S,hasInput,I)(I,rdfType,A)(B,subClassOf,A)}{(S,hasInput,B)} \quad (A_3)$$

- **Output generalisation** asserts that if a concept A is declared as an output of a requested capability c then all super-classes of A may also asserted as relaxed functional constraints of the query Q. For example, given a query Q contains the triples (?c, *hasOutput*, O_1) and (O_1, *rdftype*, $ConferenceProceedings$) and the `ConferenceProceedings` is a sub-concept of the concept `Proceedings`, then we infer a new triple (?c, *hasOutput*, `Proceedings`).

$$\frac{(S,hasInput,I)(I,rdfType,A)(A,subClassOf,B)}{(S,hasInput,B)} \quad (A_4)$$

– **Predicate relaxation** relies on semantic relations defined between concepts in a domain ontology $\mathcal{O}$ defined by a triple (C_1, P, C_2) where C_1 and C_2 are concepts and P is the predicate that relates C_1 and C_2. We distinguish two types of axioms: the predicate to domain axiom and predicate to range relaxation axiom defined as follows:

  - **Predicate to domain** axiom allows to infer a new triple (c, *hasInput*, C_1) from the triples (c, *hasInput*, P), (C_1, P, C_2) and (P, *dom*, C_1).

$$\frac{(S,hasInput,I)(I,rdfType,P)(C_1,P,C_2)(P,dom,C_1)}{(S,hasInput,C_1)} \quad (A_5)$$

  - **Predicate to range** axiom allows to infer a new triple (c, *hasOutput*, C_2) from the triples (c, *hasOutput*, P), (C_1, P, C_2) and (P, *range*, C_2).

$$\frac{(S,hasOutput,O)(O,rdfType,P)(C_1,P,C_2)(P,range,C_2)}{(S,hasOutput,C_2)} \quad (A_6)$$

Example 2. Suppose that a scientist is looking for a bioinformatic service that takes as input a *nucleotide sequence*, an *organism identifier* and a *database name* and returns as output a collection of *nucleotide sequences* that are similar to a given input sequence. Sequence similarity is computed based on a given primary sequence structure in order to deduce homology between DNA or proteins sequences. Figure 3.3 illustrates the SPARQL query graph and the corresponding relaxed query based on the hierarchy relaxation. The query answers may include services that match exactly the query, if it exists, and relaxed matchs based on Input specialisation and output generalisation relaxation. Discovery query's answers will include services that take as input any type of sequences, i.e., DNA, RNA and mRNA and that return as output all types of genomic sequences.

3.3 Non-functional Matchmaking

Relaxation techniques enlarge the search space of a discovery query allowing to select new services with various matching degrees. Let $S = \{S_1, ..., S_n\}$ be a set of candidate services that meet functional and non-functional constraints of a discovery query Q. The QoS of a candidate service S_i is represented as a vector QoS_i=<$q_{i,1}$,...,$q_{i,n}$>, where $q_{i,j}$ is the value of QoS attribute j ($1 \leq j \leq n$). The quality vectors of a set of candidate services S are used to derive a quality matrix Q.

Definition 2. *A quality matrix* $Q = \{QoS_{ij}\}$ *refers to a collection of quality attribute values of a set of candidate services, such that, each row of the matrix corresponds to the values of the* j *QoS attributes of a service* S_i

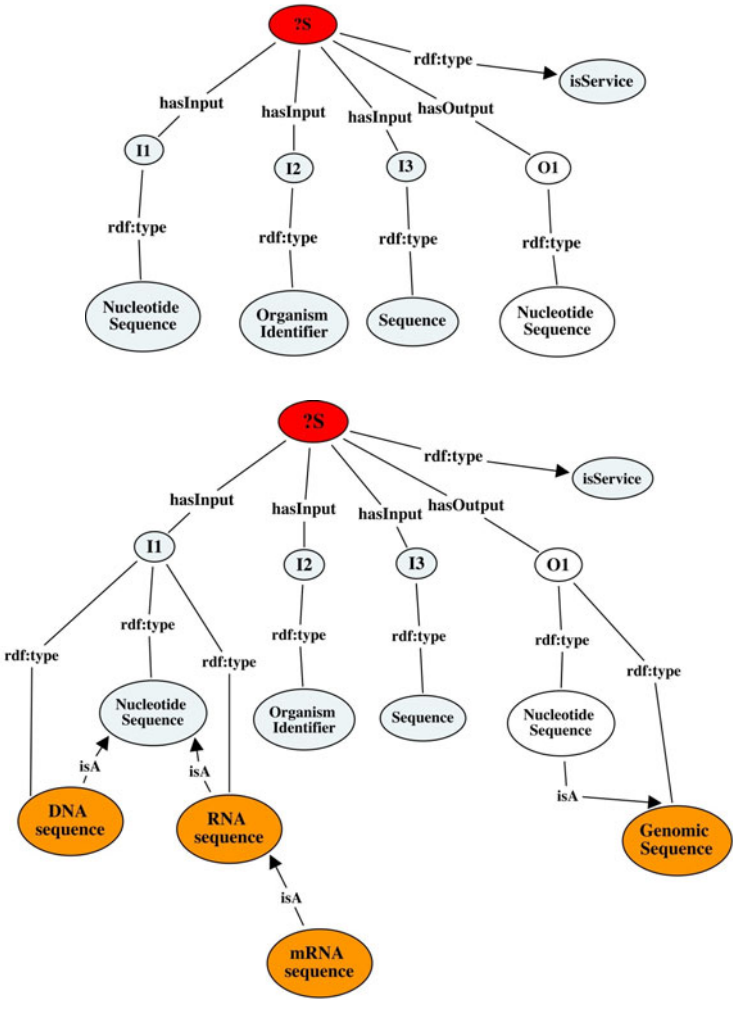

Fig. 2. Example of a Discovery Query Relaxation based on Hierarchy relaxation

QoS attribute values can be either negative or positive, thus some QoS values need to be minimized whereas other values have to be maximized. For example, availibity and reliability should be maximized while price should be minimized. To cope with this issue, the scaling phase normalizes every QoS attribute value by transforming it into a value between 0 and 1 with respect to the formulas below [16]:

– Negative attributes :

$$q'_{i,j} = \begin{cases} \frac{q_j^{max} - q_{i,j}}{q_j^{max} - q_j^{min}} & \text{if} \quad q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{else} \end{cases}$$

– Positive attributes :

$$q'_{i,j} = \begin{cases} \frac{q_{i,j} - q_j^{min}}{q_j^{max} - q_j^{min}} & \text{if} \quad q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{else} \end{cases}$$

where $q'_{i,j}$ denotes the normalized value of QoS attribute j associated with service candidate S_i; It is computed using the current value $q_{i,j}$ and also q_j^{max} and q_j^{min}, which refer respectively to the maximum and minimum values of QoS attribute j among all services candidates.

Example 3. Given the discovery query illustrated in example 3, we suppose that the quality matrix corresponding to the candidate services set is the following:

$$A = \begin{pmatrix} & Availibility & Throughput & Reputation \\ S_1 & 0.94 & 0.93 & 0.96 \\ S_2 & 0.92 & 0.97 & 0.93 \\ S_3 & 0.93 & 0.94 & 0.97 \end{pmatrix}$$

In order to rank a set of candidate services, we propose a user preference-based approach in which a user assigns a weight value for each QoS attribute. An additive value function is defined as follows:

$$f_{QoS}(service_i) = \sum_{ij}^{m} (QoS_{i,j} \times Weight_j)$$

For instance, a user may specify the following weights: $Weight_{Availibility} = 0.8$, $Weight_{throughput} = 0.6$ and $Weight_{reputation}$=0.9. A ranking function is defined as follows.

Definition 3. *Let S be the set of candidate services, R_a is a set of ranking attributes, and $R_O \in \{ascending, descending\}$ is the ranking order, then $f_{rank}(S,x,R_O) = S'$ is called the ranking function, which produces S', the ordered set of candidate services.*

4 Case Study

In this section, we describe a real-world case study conducted in the bioinformatic domain in order to illustrate the enhanced discovery approach proposed in this paper and the performance of relaxation techniques. As examples, we formulate the following queries that reprensent a set of functional requirements of a scientist:

(Q_1) *Which Web services return a set of nucleotides sequences corresponding to a gene identifier?*

(Q_2) *Which Web services return a set of protein sequences similar to a given protein sequence?*

(Q_3) *Which Web services predict the 3D-structure of a given protein sequence?*
(Q_4) *Is there a ressource that estimates phylogenies from protein sequences?*

We have generated a Web service catalog based on the BioMoby Web service catalog which is a bioinformatics Web service registry [17]. Services was described in terms of a set of inputs, outputs and a service category according to the canonical description model. The BioMoby catalog classifies Web services with respect to two hierarchies: a *service type ontology* that organizes services with respect to the task they implement and a *data type ontology* which focuses on the input/output classification. However, BioMoby does not integrates any reasoning tasks and service discovery is only performed through category navigation.

The discovery queries described above were evaluated against the Web service catalog without initiating any reasoning tasks and later with reasoning capabilities. The results obtained from this evaluation are summarized in Table 2. While, the services `runPhylipProtpars` and `runPhylipProtdist` match exactly the discovery query Q_4, no service matches exactly the discovery queries Q_1, Q_2 and Q_3.

Table 2. Discovery Query answers with and without relaxation techniques

Queries	Answers Set without Reasoning	Answers Set with Reasoning

Q_1	`No service`	`DDBJGetEntry`
		`eFetchSequence`
		`DBFetch`
		`retreive_ensembl_sequence`

Q_2	`No service`	`runNCBIblastp`
Q_3	`No Service`	`ShowPDBfromFASTA` `PDBWebService`
Q_4	`runPhylipProtPars` `runPhylipProtdist`	`runClustalWService` `PhylipService`

The first query Q_1 requires a Web service that takes as input a gene identifier and a scientific organism name (i.e., homosapiens) and returns as output a nucleotide sequence. Output generalisation axiom is applied to infer that services returning sequences as output are included in the query answers set. As illustrated by the case study, reasoning tasks enable to enlarge the search space of a discovery query.

5 Conclusion and Future Work

In this paper, we propose a twofold enhanced semantic Web service discovery approach that not only select services based on the functional requirement of

the user but also based on the non-functional requirement. First, in order to cope with semantic service description frameworks heterogeneity, we propose a canonical Web service model. In the proposed approach, flexible Matchmaking of services is based on deductive-based techniques that relax discovery queries constraints to enlarge the search space of candidate services to include services that meet functional and non functional requirement with different degrees of matching. A ranking process is performed in which users express their preferences by assigning weights to a set of ranking non-functional attributes. As future work, we aim to conduct a set of experiments to evaluate the efficiency of the discovery approach. Also, we intend to use clustering techniques to reduce the search space of a discovery query and generate services clusters based on QoS values. Clustering techniques allow to group services with respect to their QoS into a set of clusters which may improve the ranking process of services.

References

1. Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A.H.H., Elmagarmid, A.K.: Business-to-business interactions: issues and enabling technologies. The VLDB Journal 12(1), 59–85 (2003)
2. Singh, M.P., Huhns, M.N.: Service-Oriented Computing Semantics, Processes, Agents. John Wiley & Sons, Ltd., Chichester (2005)
3. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems 16, 46–53 (2001)
4. Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D.L., Sirin, E., Srinivasan, N.: Bringing semantics to web services with OWL-S. World Wide Web 10(3), 243–277 (2007)
5. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, F.C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. Applied Ontology 1(1), 77–106 (2005)
6. Kopecky, J., Vitvar, T., Bournez, C., Farrell, J.: SAWSDL: Semantic annotations for WSDL and XML schemas. IEEE Internet Computing 11(6), 60–67 (2007)
7. Cardoso, J., Barros, A., May, N., Kylau, U.: Towards a unified service description language for the internet of services: Requirements and first developments. In: Proceedings of IEEE International Conference on Services Computing, pp. 602–609 (2010)
8. Lê, L.-S., Ghose, A., Morrison, E.: Definition of a description language for business service decomposition. In: Morin, J.-H., Ralyté, J., Snene, M. (eds.) IESS 2010. LNBIP, vol. 53, pp. 96–110. Springer, Heidelberg (2010)
9. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
10. Pathak, J., Koul, N., Caragea, D., Honavar, V.G.: A framework for semantic web services discovery. In: Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management (WIDM 2005), pp. 45–50 (2005)
11. Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with OWLS-MX. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 915–922 (2006)

12. Guarino, N., Carrara, M., Giaretta, P.: Formalizing ontological commitment. In: AAAI, pp. 560–567 (1994)
13. Ayadi, N.Y.: Une Nouvelle Approche de Découverte et de Composition de Services Web á base de Médiation Sémantique et de Raisonnement déductif. PhD thesis, National School of Computer Sciences (2010)
14. Lee, K., Jeon, J., Lee, W., Jeong, S.H., Park, S.W.: QoS for web services: Requirements and possible approaches. Technical report, W3C Working Group Note (November 25, 2003)
15. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Technical report, W3C RDF Data Access Working Group (January 15, 2008)
16. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
17. Wilkinson, M.D., Links, M.: BioMOBY: An Open Source Biological Web Services Proposal. Briefings in Bioinformatics 3(4), 331–341 (2002)