Modeling, Matching and Ranking Services Based on Constraint Hardness

Claudia d'Amato[1] and Steffen Staab[2]

[1]Department of Computer Science, University of Bari - Italy
[2] ISWeb, University of Koblenz-Landau - Germany
[1]claudia.damato@di.uniba.it, [2]staab@uni-koblenz.de

Abstract. A framework for modeling Semantic Web Service is proposed. It is based on Description Logic (DL), hence it is endowed with a formal semantics and, in addition, it allows for expressing constraints in service descriptions of different strengths, i.e. *Hard* and *Soft Constraints*. *Semantic service discovery* can be performed by matching DL descriptions, expressing both Hard and Soft constraints, and exploiting DL inferences. Additionally, a method for solving the problem of *ranking* services is proposed which is based on the use of a semantic similarity measure for DL. This method can rank (matched) service descriptions on the grounds of their semantic similarity w.r.t. the service request, by preferring those that are able to better satisfy both Hard and Soft Constraints.

1 Introduction

In the last few years, the Web had two revolutionary changes, Web Service technology and the Semantic Web technology, that transformed it from a static document collection into an intelligent and dynamically integrated collection of resources. The former has allowed uniform access via Web standards to software components residing on various platforms and written in various programming languages. The latter has enriched existing Web data with their meaning, logically expressed with formal descriptions that are machine processable, thus facilitating access and integration. The major limitation of Web Services is that their retrieval and composition still require manual effort. To solve this problem, researchers have augmented Web Services with a semantic description of their functionality [8, 11]. By reusing a common vocabulary, service modelers can produce semantic service descriptions that can be shared and understood on the Web. Such vocabulary is defined by upper-level ontologies such as OWL-S[1] and WSMO[2] for *Semantic Web Services*.

In this paper we propose a framework for describing services, based on Description Logic (DL) [1]. DL is the theoretical foundation of OWL[3] language and it is endowed by a formal semantics, thus allowing expressive service descriptions. Moreover the service discovery task can be performed by algorithms defined in terms of standard and non-standard DL inferences. The use of DL in service descriptions and discovery task

[1] http://www.daml.org/services/owl-s/1.0/
[2] http://wsmo.org
[3] The ontology language for the Semantic Web, http://www.w3.org/2004/OWL/.

is not new [7, 5, 13, 10, 14, 6, 3]. However in [7] it is showed that primitives, modeled by DL, sometimes produce counterintuitive results. This issue is analyzed in [6], where preliminary guidelines for modeling service descriptions are presented. Moreover, the notion of *variance* is introduced, namely a service description usually represents numerous variants of a concrete service. Exploiting this notion, a service discovery algorithm is proposed. The assumption is that precise control of *variance* in service description is crucial to ensure quality of the discovery process.

The framework that we propose enriches the guidelines of [6]. However, in a real scenario it is important to express another form of variance in service descriptions (and particularly in the service request side), represented by the optional and the mandatory aspects of a service description. Hence we introduce the notion of *Hard* and *Soft Constraints*. Namely, we call *Hard Constraints* (HC) those features of a service description that have to be necessarily satisfied by the target services and we call *Soft Constraints* (SC) those features whose satisfaction is only preferable. To be able to distinguish *HC* and *SC* is important both for *business-to-consumer* interaction and for service discovery task. In fact with respect to business-to-consumer interaction, *HC* and *SC* allow to express the real necessities of the user; with respect to the process discovery task, the distinction between *HC* and *SC* make possible to relax some needs, increasing the possibility of satisfying a request. We propose a way to express these kind of constraints and how to deal with them during the service discovery phase.

Furthermore, we propose a new procedure for ranking the services (discovered in the previous phase), that is able to manage the variance introduced by *HC* and *SC*. The procedure uses a semantic similarity measure for DL concept descriptions that assigns higher ranks to services that are more similar to the requested service and that satisfy both its *HC* and *SC*, while services that are less similar and/or satisfy only *SC* of the request receive a low rank.

The paper is organized as follows. The DL framework for describing services is presented in the next section. In Sect. 3 the discovery and ranking processes are detailed. The conclusions of this work are drawn in Sect. 4.

2 Modelling Service Descriptions

The main reason of the attention to service descriptions is the need of automating processes such as service discovery and composition. A *service description* is expressed as a set of constraints that have to be satisfied by the service providers. It can be thought as an abstract class acting as a template for *service instances*; namely, a service description defines a space of possible service instances (as in [12]), thus introducing *variance*.

Variance is the phenomenon of having more than one instance and/or more than one interpretation for a service description. Following [6], we distinguish between *variance due to intended diversity* and *variance due to incomplete knowledge*. To explain these concepts, the notion of *possible worlds* (borrowed from the first-order logic semantics) is used. Under open-world semantics, a modeler must explicitly state which service instances are not covered by the service description. For each aspect of the service description that is not fully specified there are *several possible worlds*, reflecting a way of resolving incompleteness (*variance due to incomplete knowledge*). Besides, given a

possible world, the lack of constraints possibly allows for many instances satisfying a service description (*variance due to intended diversity*).

Let us consider the following service description (for a request):
Flight(flight) *and* operatedBy(flight,company)*and* departureTime(flight,time) *and* arrivalTime(flight,time) *and* from(flight,Germany) *and* to(flight,Italy)

and the *Service instances:*

- Flight(0542) *and* operatedBy(0542,ryanair) *and* departureTime(0542,8:00) *and* arrivalTime(0542,9:40) *and* from(0542,Hahn) *and* to(0542,Bari)
- Flight(0721) *and* operatedBy(0721,hlx) *and* departureTime(0721,12:00) *and* arrivalTime(0721,13:10) *and* from(0721,Cologne) *and* to(0721,Milan)

This description represents the request of flights from Germany to Italy, independently from departure and arrival time, company and cities involved. This lack of constraints allows many possible instances (as above), inducing *variance due to intended diversity*.

Now, let us consider the service instance below:
Flight(512) *and* operatedBy(512,airBerlin)*and* departureTime(512,18:00) *and* arrivalTime(512,19:30) *and* from(512,Berlin) *and* to(512,London)

This is also a correct instance of the service request reported above, because the fact that London is not an Italian city is left unspecified in the KB. So there can be a possible world in which London is an Italian city. Here the absence of constraints induces *variance due to incomplete knowledge*.

In order to cope with the effects of the *variance* on the semantics of a service description, it is necessary to adopt a language for service representation characterized by well-defined semantics. This is one of the peculiarities of the DL family. We intend to enrich the framework in [6] for describing services using DL, by showing how to deal with *HC* and *SC* expressed in service descriptions. The framework is reported below.

- A service description is expressed by a set of DL-axioms $D = \{S, \phi_1, \phi_2, ..., \phi_n\}$, where the axioms ϕ_i impose restrictions on an atomic concept S, which represents the service to be performed.
- Domain-specific background knowledge is represented by a *knowledge base* (KB) that contains all relevant domain-level facts.
- A *possible world*, resolving incomplete knowledge issues, is represented by a single DL model (interpretation) I of $KB \sqcup D$.
- The service instances that are acceptable w.r.t. a service description D, are the individuals in the extension S^I of the concept S representing the service.
- Variance due to intended diversity is given by S^I containing different individuals.
- Variance due to incomplete knowledge is reflected by $KB \sqcup D$ having several models $I_1, I_2, ...$

The axioms in a service description D constrain the set of acceptable service instances in S^I. These constraints are generally referred to the properties used in a description. Here, various ways for constraining a property using DL are reported.

Variety: a property can either be restricted to a fixed value or it can range over instances of a certain class. This is expressed by $\forall r.i$ (or $\exists r.i$) and $\forall r.C$ (or $\exists r.C$),

respectively. For any acceptable service instance, the value of such a property must either be an individual or a member of a class.

Multiplicity: a property can either be multi-valued, allowing service instances with several property values, or single-valued, requiring service instances to have at most one value for the property. By the number restriction $\leq 1\ r$, a property is marked as single-valued. Using the restrictions $\leq m\ r$ (with $m \geq 2$) $\geq n\ r$, $\exists r.\top$, $\exists r.C$, and $\forall r.C$ a property is marked as multi-valued.

Coverage: a property can be explicitly known to cover a range. If a property is *range-covering*, the service description enforces that in every possible world, there is an acceptable service instance with this property value. This introduces variance due to intended diversity. This kind of constraint is expressed by an axiom of the form $C \sqsubseteq \exists r^{-}.S$ in D, where the concept C is the range of the property r to be covered. A non-range-covering property induces variance due to incomplete knowledge, as in distinct possible worlds different subsets of the range will be covered.

Example 2.1. Let us consider the following requested service description

$D_r = \{\ S_r \equiv \mathsf{Company} \sqcap \exists \mathsf{payment}.\mathsf{EPayment} \sqcap \exists \mathsf{to}.\{\mathsf{bari}\} \sqcap$
$\sqcap \exists \mathsf{from}.\{\mathsf{cologne},\mathsf{hahn}\} \sqcap \leq 1\ \mathsf{hasAlliance} \sqcap$
$\sqcap \forall \mathsf{hasFidelityCard}.\{\mathsf{milesAndMore}\};$
$\{\mathsf{cologne},\mathsf{hahn}\} \sqsubseteq \exists\, \mathsf{from}^{-}.S_r \quad \}$

$KB = \{\mathsf{cologne}{:}\mathsf{Germany},\ \mathsf{hahn}{:}\mathsf{Germany},\ \mathsf{bari}{:}\mathsf{Italy},\ \mathsf{milesAndMore}{:}\mathsf{Card}\}$

As defined with framework, D_r is described as a set of axioms that impose restrictions on S_r which is the service that has to be performed. The requester asks for flight companies that fly from Cologne and Hahn to Bari and accept electronic payment when selling tickets. Then it is required that a company has at most one alliance with another flight company and, if it has a fidelity program, it is "Miles and More". In the description, different several kinds of constraints are reported. *Variety* constraints are used with the properties to, from and hasFidelityCard, indeed these properties are restricted to a fixed value. The *at-most* number restriction (≤ 1) for the property hasAlliance is a *Multiplicity* constraint with which the property hasAlliance is declared to be single-value. A *Coverage* constraint is expressed by the last axiom in D_r which makes explicit the range covered by the property from. Namely, this axiom asserts that $\{\mathsf{cologne},\mathsf{hahn}\}$ is the range coverage of the property from.

If we require that, for all air companies, the payment method is specified and that the unique method allowed is electronic payment, the service description has to be:

$D_r = \{\ S_r \equiv \mathsf{Company} \sqcap \exists \mathsf{payment}.\mathsf{EPayment} \sqcap \forall \mathsf{payment}.\mathsf{EPayment} \sqcap$
$\sqcap \exists \mathsf{from}.\{\mathsf{cologne},\mathsf{hahn}\} \sqcap \exists \mathsf{to}.\{\mathsf{bari}\} \sqcap \leq 1\ \mathsf{hasAlliance} \sqcap$
$\sqcap \forall \mathsf{hasFidelityCard}.\{\mathsf{milesAndMore}\};$
$\{\mathsf{cologne},\mathsf{hahn}\} \sqsubseteq \exists\, \mathsf{from}^{-}.S_r \quad \}$

In this way we force the existence of a payment method and oblige that all payment methods have to be electronic payments. □

The services presented in the example represent simple descriptions. In real scenarios a service request is typically characterized by some needs that *must* be necessarily satisfied and others that *should* be satisfied (expressing a preference). Then, the former will be considered as *Hard Constraints (HC)* and the latter as *Soft Constraints (SC)*.

Taking this difference into account makes the service description and management more complex. The new descriptions have to be defined by the user/requester through an *HC* set and an *SC* set, whose elements are expressed in DL as seen above.

More formally, let $D_r^{HC} = \{S_r^{HC}, \sigma_1^{HC}, ..., \sigma_n^{HC}\}$ be the set of *HC* for a requested service description D_r and let $D_r^{SC} = \{S_r^{SC}, \sigma_1^{SC}, ..., \sigma_m^{SC}\}$ be the set of *SC* for D_r. Every element in D_r^{HC} and in D_r^{SC} is expressed as previously seen. The complete description of D_r is given by $D_r = \{S_r \equiv S_r^{HC} \sqcup S_r^{SC}, \sigma_1^{HC}, ..., \sigma_n^{HC}, \sigma_1^{SC}, ..., \sigma_m^{SC}\}$. Note that, in this description, new information on constraint hardness has been added.

Example 2.2. Let us consider a slightly modified version of the previous example distinguishing between *HC* and *SC*:

$$D_r = \{\ S_r \equiv \mathsf{Flight} \sqcap\ \exists \mathsf{from}.\{\mathsf{Cologne},\ \mathsf{Hahn},\ \mathsf{Frankfurt}\} \sqcap \exists \mathsf{to}.\{\mathsf{Bari}\} \sqcap \\ \sqcap \forall \mathsf{hasFidelityCard}.\{\mathsf{MilesAndMore}\}; \\ \{\mathsf{Cologne},\ \mathsf{Hahn},\ \mathsf{Frankfurt}\}\ \sqsubseteq\ \exists\, \mathsf{from}^{-}.S_r; \quad \{\mathsf{Bari}\}\ \sqsubseteq\ \exists\, \mathsf{to}^{-}.S_r \quad \}$$

where

$$HC_r = \{\ \mathsf{Flight} \sqcap \exists \mathsf{to}.\{\mathsf{Bari}\}\ \sqcap \exists \mathsf{from}.\{\mathsf{Cologne},\ \mathsf{Hahn},\ \mathsf{Frankfurt}\}; \\ \{\mathsf{Cologne},\ \mathsf{Hahn},\ \mathsf{Frankfurt}\}\ \sqsubseteq\ \exists\, \mathsf{from}^{-}.S_r; \quad \{\mathsf{Bari}\}\ \sqsubseteq\ \exists\, \mathsf{to}^{-}.S_r\ \}$$

$$SC_r = \{\ \mathsf{Flight} \sqcap \forall \mathsf{hasFidelityCard}.\{\mathsf{MilesAndMore}\}\};$$

$$KB = \{\ \mathsf{cologne,hahn,cologne{:}Germany},\ \mathsf{bari{:}Italy},\ \mathsf{MilesAndMore{:}Card}\}$$

With this service description a requester asks for flights starting from Frankfurt or Cologne or Hahn and arriving at Bari. The use of "Miles And More" card would be preferred. Departure and arrival places are expressed as *HC*. This means that provided services must fulfil these constraints. This is understandable thinking, for instance, to a requester that want to go from Koblenz to Bari. He/she is interested in Cologne, Hahn and Frankfurt airports because they have the same distance from Koblenz, while he/she is not interested in other airports because much more distance. Instead the use of "Miles And More" card is expressed as *SC*, namely flights that allow the use of this card are preferred, but the requester accepts also flights that do not allow the use of this card. This is because the use of *Miles and More* card is advantageous for the requester but it is not his primary need; his/her primary need is to have a flight for reaching Bari. □

This new representation can better model the requester's needs, allowing to express real-life preferences, feature not considered in the original framework [6]. Moreover expressing *SC* allows to have service instances satisfying a request even if part of it is ignored, thus augmenting the possibility of having response for a request.

3 Service Discovery and Ranking

Service Discovery is the task of locating service providers that can satisfy the requester's needs. In this scenario, semantic service descriptions can be used to automate the task. Discovery is performed by matching a requested service description to the service descriptions of potential providers, in order to detect relevant ones. Two service descriptions match if there is an acceptable instance for both descriptions [14, 12, 6, 7].

Considering the framework presented in Sect. 2, let D_r and D_p respectively a requested service description and a provided service description, expressed as a set of axioms imposing restrictions on the services that have to be performed that are called S_r and S_p respectively. The matching process (w.r.t. a KB) can be defined as a boolean function *match*(KB, D_r, D_p) which specifies how to apply DL inferences to perform matching. Various matching procedure, based on DL inferences, have been proposed [7, 14, 12]. We fix our attention to those proposed in [6]. Differently from the others, this procedure is able to treat *variance* (particularly variance due to incomplete knowledge) without being too weak or too strong. Indeed, the other matching procedures [13, 14, 5] consider a match valid if there exists a common instance service at least in *one possible world*. This match, called *Satisfiability of Concept Conjunction*, is the weakest check w.r.t. both kinds of variance. Indeed, along the dimension of intended diversity, it is sufficient to find one common service instance. Along the dimension of incomplete knowledge, it is sufficient to find one possible world in which such a service instance exists, regardless of all other possible worlds.

Another type of matching procedure [7, 10, 9] executes match by checking for subsumption, either of the requestor's description by the provider's or vice versa. It is called *Entailment of Concept Subsumption*. This check is very strong, since it requires one of the service descriptions to be more specific than the other, for all service instances in all possible worlds. Conversely, a valid match for the procedure in [6] occurs when there exists a common instance service between a provider's service description D_p and a requestor's service description D_r w.r.t. KB, in *every possible world*. It can be formalized as:

$$KB \cup D_r \cup D_p \models \exists x S_r(x) \wedge S_p(x) \Leftrightarrow KB \cup D_r \cup D_p \cup \{S_r \sqcap S_p \sqsubseteq \bot\}\text{unsatisfiable}$$

This check is called *Entailment of Concept Non-Disjointness*. It is stronger than *Satisfiability of Concept Conjunction* because checks for an intersection in every possible world, but it is not as strong as *Entailment of Concept Subsumption*, because it does not require one of the sets of acceptable service instances to be fully contained in the other set. This match increases (w.r.t *Entailment of Concept Subsumption*) the possibility to find interesting provided services, decreasing the error due to variety (more present in *Satisfiability of Concept Conjunction*).

The provided services, selected by the matching process, have to be ranked w.r.t. certain criteria and then returned to the requestor, in order to start the negotiation process between the requested service and provided services. We focus our attention on the ranking services process and propose a ranking procedure based on the use of a semantic similarity measure for DL. This procedure ranks in higher positions, provided services that are most similar to the requested service and that satisfy both *HC* and *SC* of the requested service. Instead, services that are less similar and/or satisfy only *HC* are ranked in lower positions. This is because services that satisfy both *HC* and *SC* can satisfy more needs of the requested service than services that satisfy only *HC*.

In the following, the measure used for determining the similarity value between service descriptions is presented, then the ranking services process is explained.

3.1 The semantic similarity measure

The semantic similarity measure used for ranking services is presented in [4], in which a semantic similarity measure for DL concept definitions, asserted in the same ontology, is defined. Considering Sect. 2, service descriptions can be viewed as DL concept descriptions asserted in a *T-Box* and their instances can be regarded as concept assertions in an *A-Box* and the *Canonical Interpretation* can be considered (see [1] for details about *T-Box, A-Box* and *Canonical Interpretation*). So the following similarity measure can be applied to the service descriptions

Definition 3.1 (Semantic Similarity Measure). *Let $\mathcal{L}$ be the set of all service descriptions and let $\mathcal{I}$ be the canonical interpretation which maps every service description to its instances. The Semantic Similarity Measure* ***s*** *is a function $s : \mathcal{L} \times \mathcal{L} \mapsto [0, 1]$ defined as follows:*

$$s(S_r, S_p) = \frac{|(S_r \sqcap S_p)^{\mathcal{I}}|}{|(S_r \sqcup S_p)^{\mathcal{I}}|} \cdot \max\left(\frac{|(S_r \sqcap S_p)^{\mathcal{I}}|}{|S_r^{\mathcal{I}}|}, \frac{|(S_r \sqcap S_p)^{\mathcal{I}}|}{|S_p^{\mathcal{I}}|}\right)$$

where $(\cdot)^{\mathcal{I}}$ computes the concept extension w.r.t. $\mathcal{I}$, $|\cdot|$ returns the cardinality of a set.

This function assigns the maximum value in case of semantic equivalence of the service descriptions. Otherwise it assigns a value in the range $[0, 1[$. This value grows with the increasing size of the set of service instances in common (given by the first factor) and it is weighted by a factor which represents the incidence of the intersection with respect to either concept. Particularly, the increase of this factor implies that the concepts are closer to subsume one the other (or even to be equivalent). This means to consider similarity not as an absolute value, but weighted with respect to the degree of non-similarity. The function s is really a similarity measure, (according to the formal definition [2]) and its complexity mainly depends on the complexity of the instance checking operator (for the chosen DL) used to define the extension of concept descriptions (see [4] and Sect.3.3 for details). Similarity value is computed between a requested service description S_r and a provided service description S_p, so the instance checking operator has to define the set of instances for them. For every provided service, the set of instances is known. We need to define the extension of S_r. Note that the measure is applied after the matching process and that the chosen matching procedure (see Sect.3) selects all provided services S_p that have at least one instance satisfying S_r. So it is straightforward to understand that the set of instances for S_r is given by the union of the provided service instances that satisfy S_r. Namely:

$$S_r^I = \bigcup_{j=1}^{n} \{x | S_r(x) \wedge S_p^j(x)\}$$

where n is the number of provided services selected by the matching process.

3.2 Ranking Procedure

The rationale of the procedure consists in measuring the similarity between the requested and the provided services, selected by the matching phase: a higher similarity will result in higher rankings.

The presented measure assigns highest values to services that share most of the instances with S_r so, as in [6], the criterion used is based on *variance*, namely, a provider

is better than another if the variance it provides is greater than the other. However, differently from [6], we are able to supply a total order of the provided services (rather than a partial order). Anyway this is not enough for ensuring that provided services satisfying both *HC* and *SC* of S_r will be in the higher positions, while services satisfying only *HC* will be in the lower positions. Let us consider the following scenario: let S_r be the requested service and let S_p^l and S_p^k two provided services, selected by the matching procedure. As said in Sect.2, a service is mainly described by the set of *HC* and *SC*. Particularly, a service can also be described only by *HC* [4]. Let us suppose that S_r is described by both *HC* and *SC*, that S_p^l is a provided service whose instances all satisfy only the *HC* of S_r and that S_p^k is a provided service whose instances all satisfy both *HC* and *SC* of S_r. So let us consider the canonical interpretation, it is straightforward to see that $\forall x : S_p^k(x) \rightarrow S_p^l(x) \Leftrightarrow (S_p^k)^I \subseteq (S_p^l)^I \Rightarrow |(S_p^k)^I| \leq |(S_p^l)^I| \Rightarrow s(S_r, S_p^k) \leq s(S_r, S_p^l)$. This is the opposite result w.r.t. our criterion. Indeed we want that provided services satisfying both *HC* and *SC* of S_r and more similar to S_r are on top of the ranking. For achieving this goal, the ranking procedure is:

given $S_r = \{S_r^{HC}, S_r^{SC}\}$ service request; $S_p^i (i = 1, .., n)$ provided services selected by *match*(KB, D_r, D_p^i);
for $i = 1, \ldots, n$ **do** compute $\bar{s}_i := s(S_r^{HC}, S_p^i)$
let be $S_r^{new} \equiv S_r^{HC} \sqcap S_r^{SC}$
for $i = 1, \ldots, n$ **do**
    compute $\overline{\bar{s}_i} := s(S_r^{new}, S_p^i)$
    $s_i := (\bar{s}_i + \overline{\bar{s}_i})/2$

Let us call S_r^{HC} the requested service description relative to *HC* and S_r^{SC} those relative to *SC*. For all S_p^i, the similarity values $\bar{s} := s(S_r^{HC}, S_p^i)$ are computed. Hence, let us consider a new service description $S_r^{new} \equiv S_r^{HC} \sqcap S_r^{SC}$ defined as the conjunction of *HC* and *SC* of S_r. The instances of S_r^{new} satisfy both *HC* and *SC* of S_r. So, for all S_p^i the similarity values $\overline{\bar{s}} := s(S_r^{new}, S_p^i)$ are computed. It is straightforward to understand that a S_p^i satisfying only *HC* of S_r will has $\overline{\bar{s}} = 0$. For all S_p^i, the final similarity value s_i is given by the average between $\bar{s}$ and $\overline{\bar{s}}$. This last value s_i is used for setting the rank of the services.

Let clarify this process considering the following example. D_r is a requested service description, D_p^l and D_p^k are two provided service description, selected by the matching process. *KB* is the used knowledge base. We rank D_p^l and D_p^k. Let note that here D_p^l and D_p^k are described specifying their *HC* and *SC*. This is in order to show how the ranking process work in this case. However, it is straightforward to see that the procedure can rank provided services even if they are described without any specification about their constraint hardness.

$$D_r = \{\ S_r \equiv \mathsf{Flight} \sqcap \forall \mathsf{operatedBy.LowCostCompany} \sqcap \exists \mathsf{to.\{bari\}} \sqcap \\ \sqcap\ \exists\ \mathsf{from.\{cologne,hahn\}} \sqcap \forall \mathsf{applicableToFlight.Card}; \\ \{\mathsf{cologne,hahn}\}\ \sqsubseteq\ \exists\ \mathsf{from}^{-}.S_r \qquad\qquad \}$$

[4] A service can not be described only by *SC* because it means ask for a service that contains only optional constraints and this does not make sense.

where

$HC_r = \{$ Flight $\sqcap \exists$to.{bari} $\sqcap \exists$ from.{cologne,hahn}
{cologne,hahn} $\sqsubseteq \exists$ from$^-$.S_r $\}$
$SC_r = \{$ Flight $\sqcap \forall$operatedBy.LowCostCompany $\sqcap \forall$applicableToFlight.Card$\}$;

$D_p^l = \{$ $S_p^l \equiv$ Flight $\sqcap \exists$to.Italy $\sqcap \exists$from.Germany;
Germany $\sqsubseteq \exists$ from$^-$.S_p^l; Italy $\sqsubseteq \exists$ to$^-$.S_p^l $\}$
where

$HC_p^l = \{$ Flight $\sqcap \exists$to.Italy $\sqcap \exists$from.Germany;
Germany $\sqsubseteq \exists$ from$^-$.S_p^l; Italy $\sqsubseteq \exists$ to$^-$.S_p^l $\}$
$SC_p^l = \{\}$

$D_p^k = \{$ $S_p^k \equiv$ Flight $\sqcap \forall$operatedBy.LowCostCompany $\sqcap$ $\exists$to.Italy $\sqcap$
$\sqcap \exists$from.Germany;
Germany $\sqsubseteq \exists$ from$^-$.S_p^k; Italy $\sqsubseteq \exists$ to$^-$.S_p^k $\}$
where

$HC_p^k = \{$ Flight $\sqcap \exists$to.Italy $\sqcap \exists$from.Germany;
Germany $\sqsubseteq \exists$ from$^-$.S_p^k; Italy $\sqsubseteq \exists$ to$^-$.S_p^k $\}$
$SC_p^k = \{$ Flight $\sqcap \forall$operatedBy.LowCostCompany$\}$;

$KB = \{$ cologne,hahn:Germany, bari:Italy, LowCostCompany $\sqsubseteq$ Company $\}$

Let us consider S_r, S_p^l and S_p^k. Let note that S_p^l satisfies only *HC* of S_r while S_p^k satisfies both *HC* and *SC* of S_r. Let us suppose that the extensions of S_p^l and S_p^k are: $|(S_p^l)^{\mathcal{I}}| = 8$ and $|(S_p^k)^{\mathcal{I}}| = 5$ and all instances satisfy S_r. Note that $S_p^k \sqsubseteq S_p^l$ then $(S_p^k)^{\mathcal{I}} \subseteq (S_p^l)^{\mathcal{I}}$. So $|(S_r)^{\mathcal{I}}| = 8$. Furthermore, $S_r \not\equiv S_p^l$ and $S_r \not\equiv S_p^k$. Let us consider S_r^{HC} given by: $S_r^{HC} \equiv$ Flight$\sqcap \exists$from.{cologne,hahn}$\sqcap \exists$to.{bari} and S_r^{SC} given by $S_r^{SC} \equiv$ Flight $\sqcap \forall$operatedBy.LowCostCompany $\sqcap \forall$applicableToFlight.Card. Known that all the instances of S_p^l and S_p^k satisfy S_r and particularly that S_p^l satisfies only *HC* of S_r while S_p^k satisfies both *HC* and *SC* of S_r, it is straightforward to understand that $|(S_r^{HC} \sqcap S_p^l)^I| = 8$ and that $|((S_r^{HC} \sqcap S_r^{SC}) \sqcap S_p^l)^I| = |(S_r^{new} \sqcap S_p^l)^I| = 0$, consequently $\overline{\overline{s_l}} = 0$. In the other case, we know that $|(S_p^k)^{\mathcal{I}}| = 5$ and that S_p^k satisfies both *HC* and *SC* of S_r. So, some instances of S_p^k can satisfy only *HC* of S_r and others satisfy both *HC* and *SC* (in the better case we could have that all instances of S_p^k satisfy both *HC* and *SC*). Let us suppose that instances of S_p^k that satisfy both *HC* and *SC* of S_r, namely that satisfy $S_r^{new} \equiv S_r^{HC} \sqcap S_r^{SC}$ are 3. Let applying the procedure:

$\bar{s_l} := s(S_r^{HC}, S_p^l) = \frac{|(S_r^{HC} \sqcap S_p^l)^{\mathcal{I}}|}{|(S_r^{HC} \sqcup S_p^l)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r^{HC} \sqcap S_p^l)^{\mathcal{I}}|}{|(S_r^{HC})^{\mathcal{I}}|}, \frac{|(S_r^{HC} \sqcap S_p^l)^{\mathcal{I}}|}{|(S_p^l)^{\mathcal{I}}|}) = \frac{8}{8} \cdot \max(\frac{8}{8}, \frac{8}{8}) = 1$

$\bar{s_k} := s(S_r^{HC}, S_p^k) = \frac{|(S_r \sqcap S_p^k)^{\mathcal{I}}|}{|(S_r \sqcup S_p^k)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r \sqcap S_p^k)^{\mathcal{I}}|}{|S_r^{\mathcal{I}}|}, \frac{|(S_r \sqcap S_p^k)^{\mathcal{I}}|}{|(S_p^k)^{\mathcal{I}}|}) = \frac{5}{8} \cdot \max(\frac{5}{8}, \frac{5}{5}) = 0.625$

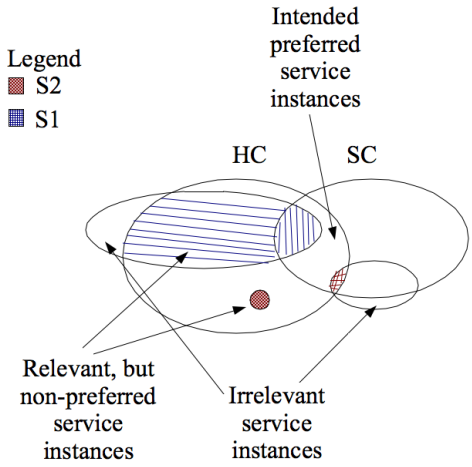

Fig. 1. Common instances between requested service and provided services for their ranking

The next step is computing $\overline{\overline{s_l}}$ and $\overline{\overline{s_k}}$, that, considering the observation above are given by:

$\overline{\overline{s_l}} := s(S_r^{new}, S_p^l) = 0$

$\overline{\overline{s_k}} := \frac{|(S_r^{new} \sqcap S_p^k)^{\mathcal{I}}|}{|(S_r^{new} \sqcup S_p^k)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r^{SC} \sqcap S_p^k)^{\mathcal{I}}|}{|(S_r^{new})^{\mathcal{I}}|}, \frac{|(S_r^{new} \sqcap S_p^k)^{\mathcal{I}}|}{|(S_p^k)^{\mathcal{I}}|}) = \frac{3}{5} \cdot \max(\frac{3}{3}, \frac{3}{5}) = \frac{3}{5} = 0.6$

Hence the final similarity values are: $s_l = 0.5, s_k = 0.6125$ and so the ranking of the provided services is:

1. S_p^k *Similarity Value* 0.6125
2. S_p^l *Similarity Value* 0.5

This result is consistent with the goal. Namely, using this procedure, provided services are ranked w.r.t. both variance and satisfaction of S_r's *SC*.

More in general, the rational of the ranking procedure is showed in Fig. 1. As seen in Sec. 3.1, due to the chosen matching procedure, all the services that have to be ranked have at least one instance satisfying S_r. In the figure, *HC* and *SC* represent the Hard and Soft Constraints of the requested service and S_1 and S_2 represent services to rank. All the instances of S_1 or S_2 that are in *HC* are relevant instance service for S_r, because they satisfy its *HC*. However they are not the preferred instance services for S_r because they do not satisfy also *SC* of S_r. For example if the *HC* of S_r ask for flights from Cologne to Bari and the *SC* of S_r ask for flights that allow the use of Miles and More card then all the instances of S_1 and S_2 that are in *HC* are all flights from Cologne to Bari held by two different company. This instances are relevant because they satisfy the main need, however flights from Cologne to Bari that allow the use of Miles and More card will be preferred w.r.t. flights that do not allow the use of this card. Thus the preferred instance services for S_r are all the instances of S_1 and S_2 that are in the intersection between *HC* and *SC*. These instances are all the flights from Cologne to Bari of S_1 and S_2 that allow the use of Miles and More Card.

The parts of S_1 and S_2 outside *HC* represents all the instances that do not satisfy *HC* and thus irrelevant service instances for S_r; for example flights having a departure and/or arrival place different from those requested. In the same way the part of S_2 outside *HC* but in *SC* represents irrelevant service instances for S_r because these instances satisfy *SC* without satisfying *HC*; for example represents flights that allow the use of Miles and More card but that do not arrive in Bari and so these are not interesting for the request.

At the first time, the procedure ranks provided services that satisfy *HC* w.r.t. variance criteria, indeed provided services that share most of service instances with S_r have higher similarity value. Hence *SC* are considered. The procedure assigns an additional similarity value to provided services that satisfy also *SC*. This similarity value is assigned, again using the variance criteria. Let note that in computing the additional similarity value are not considered all the service instances satisfying *SC* of S_r but only the service instances satisfying both *HC* and *SC* of S_r. This avoid to have in higher ranking position provided services that are very similar to *SC* but dissimilar from *HC*, whose instances are obviously not preferred w.r.t. to services mostly similar to *HC*. Indeed the latter can have a lot of instances satisfying *SC* but that are not relevant at all for the main request.

3.3 Discussion

In this section the complexity of the proposed algorithms is analyzed. Both matching and ranking procedure use reasoning services. Indeed for the matching process, two service descriptions match if their conjunction is not subsumed by the *bottom* concept. So the complexity of the matching procedure depends from the complexity of the subsumption operator for the chosen DL. For the ranking process, the dominant operation is the computation of the similarity value, for which the s measure is called twice for every matched provided service that has to be ranked. The complexity of s mainly depends from the complexity of the *instance checking* operator (for the chosen DL), used for computing the extensions of the service descriptions and the extension of their conjunction and disjunction. However the complexity of the ranking procedure could be decreased by reducing the number of calls to the instance checking operator. Indeed, the extensions of all available services can be computed beforehand, so at request-time, only the extension of the requested service description has to be computed. The extensions of the conjunctive and disjunctive service descriptions can be computed by the use of set theory applied to the extensions already determined.

4 Conclusion and Future Work

This paper proposes a framework based on DL for describing services. Differently from [6], to which it is inspired, our framework allows to express hard and soft constraints in a service description, thus obtaining a more flexible framework for service modeling.

Moreover, these new kind of constraints can be useful for supplying to the requester the most appropriate provided services among those selected from the matching phase. Indeed a new ranking procedure was presented in order to rank services selected by the

discovery process. The aim of this ranking procedure was to help during the choice in the list of eligible provided services discovered in the previous phase. To this purpose, the procedure can take into account the presence of *HC* and *SC*, the *variance* exploiting a measure that can assess the semantic similarity between service descriptions. This yields a total order among the selected services, differently from [6] where the ranking procedure provides only a partial order and is not able to manage *HC* and *SC*.

For the future, an experimentation involving the framework, the matching and ranking procedures is necessary, in order to show the improvement of the quality of the results supplied to the requester. Moreover, a new matching process could be useful for further increasing the quality of the discovery process and reduce the noise in the selection of services.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] H.H. Bock. *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*. Springer-Verlag, 1999.

[3] Andrea Calì, Diego Calvanese, Simona Colucci, Tommaso Di Noia, and Francesco M. Donini. A description logic based approach for matching user profiles. In *Description Logics*, 2004.

[4] C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In A. Pettorossi, editor, *Proceedings of Convegno Italiano di Logica Computazionale, CILC05*, Rome, Italy, 2005.

[5] J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *Proc. of KI-2001 Workshop on Applications of Description Logics*, page vol. 44, 2001.

[6] S. Grimm, B. Motik, and C. Preist. Variance in e-business service discovery. In *Proceedings of the ISWC Workshop on Semantic Web Services*, 2004.

[7] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 331–339, New York, NY, USA, 2003. ACM Press.

[8] Sheila A. McIlraith and David L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.

[9] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In *WWW*, pages 321–330, 2003.

[10] M. Paolucci, T. Kawamura, T.R. Payne, and K.P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347, 2002.

[11] Massimo Paolucci and Katia P. Sycara. Autonomous semantic web services. *IEEE Internet Computing*, 7(5):34–41, 2003.

[12] Chris Preist. A conceptual architecture for semantic web services. In *Proceeding of International Semantic Web Conference*, pages 395–409, 2004.

[13] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *SWWS*, pages 447–461, 2001.

[14] D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 89–98, New York, NY, USA, 2002. ACM Press.