Behavioral Matchmaking of Semantic Web Services*

Zijie Cong and Alberto Fernández

CETINIA, Universidad Rey Juan Carlos, Madrid, Spain
zijie@ia.urjc.es, alberto.fernandez@urjc.es

Abstract. Service matchmaking is an integral link of service discovery, composition, invocation and other similar tasks under Service-Oriented Architecture (SOA). Most current approaches measure the degree of match of two services based merely on their I/O pairs which could leads to false results. This paper presents an approach for matchmaking in Semantic Web Services (SWS) that considers each service as a sub-graph of the semantic network of the ontology formed by inputs, outputs, pre- and post-conditions with contribution of syntactical information such as keywords and textual descriptions. The similarity between services is defined as the similarity between these graphs. The aim of this approach is to reveal the internal work flow and intention of service, i.e. behavior, thus it agrees with human intuition to a larger extent than existing approaches.

1 Introduction

The original intention of adding semantic annotations to web services is to improve the automation of service discovery, selection, invocation and interoperation by letting service descriptions to be machine-processable [12]. One integral part of such automation is matchmaking among services.

Various approaches have been proposed in previous studies. Without concerns about semantics of its components, one primitive method to calculate the similarity of services is based on the syntactical information - e.g. keywords, tag-clouds and textual descriptions.

For services with semantic information, inputs/outputs (I/O) matching is a common method for measuring the similarity. Inputs and outputs of a semantic service are instances of ontological concepts. The similarity of two services is determined by the subsumption relation the taxonomy tree between corresponding concepts of I/O pair. The result is a degree of semantic similarity, such as EXACT, PLUG-IN, SUBSUMES and FAIL [11]. Some studies, such as [9], aimed to achieve higher robustness and precision by combining both semantic and syntactical approaches.

* Work partially supported by the Spanish Ministry of Science and Innovation through grants TIN2009-13839-C03-02 and CSD2007-0022(CONSOLIDER-INGENIO 2010)

More recently, various graph based approaches have been proposed. In [7], a service was considered as a composition of processes and thus could be represented as a finite-state machine (FSM), the similarity between services was defined as the similarity between two FSMs. Like other similar graph-based approaches [6,5], it concentrated on structural similarity of services instead of the semantic similarity of atomic units of functionality.

This paper presents a novel but preliminary approach for service matchmaking. The main rational behind this approach is that a service could be considered as a sub-graph (Service Behavioral Graph) of a semantic network which maps input concepts to output concepts via elements specified in conditions, retrieved from textual description, it reveals the behavior of services which could be a more intuitive option for calculating the degree of match of services.

The rest of the paper is organized as follows. Section 2 shows the motivation of this work with an example of 2 service descriptions using a shared ontology with 20 concepts and 10 relations. The concept and main components of Service Behavioral Graph are defined in section 3, algorithms for obtaining those components are also shown. Section 4 describes the calculation of the degree of match between two services and how it compromises with other studies. Finally, in section 5 we conclude our current work with a discussion and future plans.

2 Motivation

Although an appropriate measurement of degree of match is difficult to define, it is consensus that the result of matching should agree with human intuition. Inputs and outputs sometimes may not provide sufficient information about service's behavior, and relying solely on them may lead to false results. An example is presented in the rest of this section.

Figure 1 illustrates an ontology of publication with 20 concepts and 10 relations connecting them, this ontology is adopted from [1].

Every service description used in this paper is a 4-tuple (T, I, O, Q), where:

$S_{(T)}$ is syntactical information of the service which may include keywords, tag-cloud or textual description.
$S_{(I)}$ is a set of input concepts.
$S_{(O)}$ is a set of output concepts.
$S_{(Q)}$ is a set of predicates that must be true *after* the execution of the service, i.e. post-conditions.

Due to the diversity of specifications and implementations of conditions in different service description approaches, in this paper, for the sake of simplicity, we consider these conditions as a conjunction of predicates that are defined in the ontology. A predicate is a binary relation between two concepts, such as *hasBirthday(Novelist, Date)*.

The preconditions are intentionally ignored as these conditions are usually checked before the execution of the service thus they do not concern with the actual behaviors.

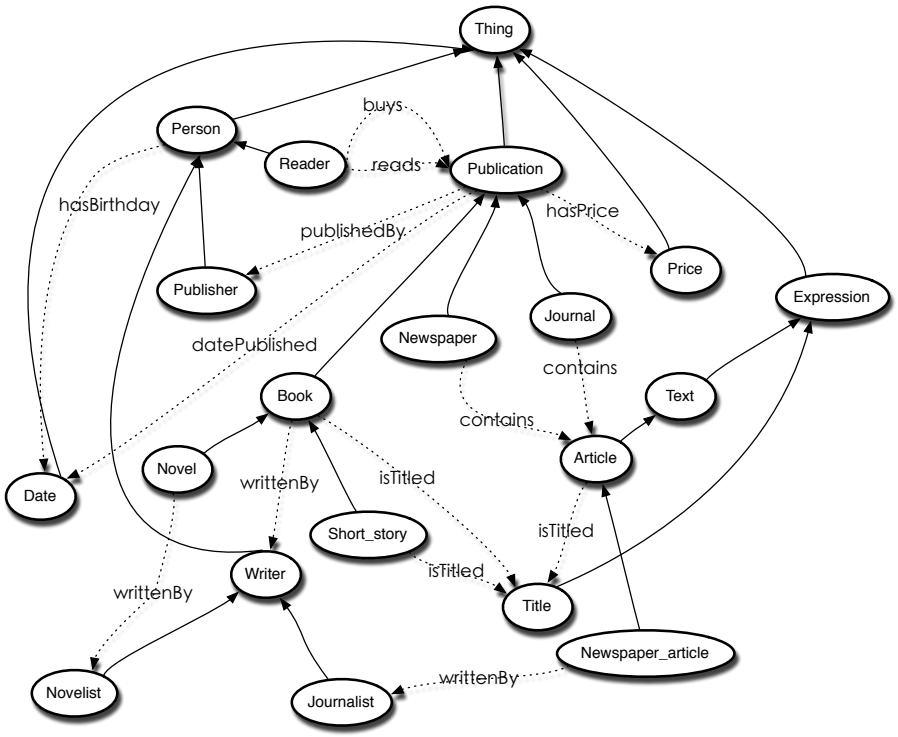

Fig. 1. An ontology of publications with 20 concepts and 10 relations, solid lines represent *subClassOf* relations.

To illustrate the problem with I/O matching approaches, we define two services in figure 2. By using I/O matching approaches such as [11], the matchmaker will not be able to distinguish between S_1 and S_2 as their inputs and outputs are identical, thus these two services matches exactly, even though the functionality of those two services is different.

Therefore the aim of our approach is to overcome the above limitations by exploiting the behavioral information of services.

3 Service Behavioral Graph (SBG)

To exploit the behavioral information of a service, we consider a service as a function that maps its inputs to its outputs. In Semantic Web Services, where inputs and outputs are ontological concepts, this mapping is usually defined by relations in the same domain ontology. As an ontology can be represented by a *multi-relational graph* where each vertex denotes a concept and each edge denotes a relation between concepts, a service thus can be further considered as a sub-graph of an ontology. More formally,

$$S_1 = \begin{cases} T = & returns\ the\ birthday\ of\ a\ given\ novelist \\ I = & \{Novelist\} \\ O = & \{Date\} \\ Q = & hasBirthday(Novelist,\ Date) \end{cases}$$

$$S_2 = \begin{cases} T = & published\ date\ of\ a\ novelist's\ earliest\ book \\ I = & \{Novelist\} \\ O = & \{Date\} \\ Q = & \emptyset \end{cases}$$

Fig. 2. Services using the ontology of publication

Definition 1. *(Service Behavioral Graph)* Let G be an ontology in its graph representation, $G = (V, \mathbb{E})$ where V is the set of concepts and $\mathbb{E}$ is the set of relations of heterogeneous types, where each relation is represented using a pair $< L, (V \times V) >$, where L is the label of the relation (e.g. *hasBirthday*). A service S is denoted as $G_S = \left(V^{'}, \mathbb{E}^{'}\right)$ where $V^{'} \subseteq V$ and $\mathbb{E}^{'} \subseteq \mathbb{E}$. Elements of $V^{'}$and $\mathbb{E}^{'}$ are identified using the service description. This sub-graph of the ontology is referred as *Service Behavioral Graph* (SBG).

Figure 3 shows the SBGs of S_1 and S_2. These graphs can be discovered from the ontology graph using *critical elements* and *behaviorally correct paths*, which are defined in the following sections. Note that those graphs are different each other despite the fact that their I/O descriptions coincide.

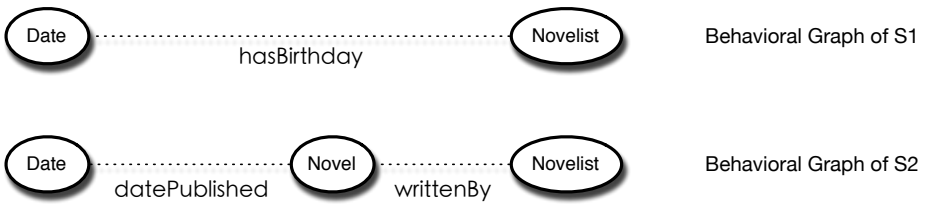

Fig. 3. SBGs of S_1 and S_2

3.1 Critical Elements

As we have mentioned in the beginning of this section the mapping from inputs to outputs is defined by the relations in the domain ontology. This mapping is, in fact, a set of paths from input concepts to output concepts, consisting of one or more relations. There may exist multiple paths between a pair of I/O concepts, therefore, finding proper paths is critical for describing the service's behavior correctly.

Such paths are determined by several components in the ontology which can be concepts or relations, and are referred as *critical elements* in this paper. *Q(Post-condition)* and *T(Syntactical information)* of service descriptions may offer some clues to determine these critical elements.

Syntactical Information Syntactical information is valuable for revealing service's behaviors. For example, even though $S_{1(I,O)} = S_{2(I,O)}$, the textual descriptions *(T)* differ these two services at human-readable level. To find the critical elements, syntactical information and ontological components' identifiers (ID or labels) need to be processed using information retrieval techniques [3] to transform them into a set of keywords with irrelevant words and morphological variants removed. Then components with keywords appeared in the syntactical information of the service are considered to be a critical element. For example, in S_1, relation *hasBirthday,* concepts *Novelist* are identified as critical elements because the words "birthday" and "novelist" have appeared in $S_{1(T)}$.

Post-conditions The post-conditions is a set of predicates that must be true after the execution of the service, for example, conditions that are specified in the *ConditionalOutput* or *ConditionalEffect* part of an OWL-S service description. These predicates often connect input elements with output elements, hence they reveal important information about service's behavior.

Figure 4 shows how *critical elements* can be determined and weighted. This function takes two arguments: O is the domain ontology used by service and S is the service description 4-tuple. Any postcondition which its domain and range are from inputs and outputs separately is considered to be critical elements with weight 1. Syntactical information such as textual descriptions are tokenized and stemmed. A normalized weight computed using TF-IDF [8] technique is assigned to each token. The TF-IDF weight measures the importance of certain words and their corresponding ontological elements in service, the calculation can be done with information of other services in the registry where service advertisements are registered, most commonly a UDDI registry [4]. Ontological elements corresponding to these tokens are considered as critical elements and assigned with weight of its token.

```
1: function CriticalElements(O, S)
2:     for all o ∈ O do
3:         o.weight = 0                                   ▷ Initialize weights to 0
4:     end for
5:     for all q ∈ S_(Q) do
6:         if range, domain of q are in S_(I) and S_(O) separately then
7:             q.weight = 1
8:         end if
9:     end for
10:    T ← TOKENIZE(S_(T))
11:    T ← STEM(T)
12:    T ← S_(T)\Stoplist                                 ▷ Remove common words
13:    for all t ∈ T do
14:        w ← TFIDF(t)                                   ▷ Calculate normalized tf-idf weight
15:        E ← ONTOLOGYELEMENTS(O, t) ∩ S_(I,O)
16:        for all e ∈ E do
17:            e.weight = w                               ▷ Assign weights to the elements
18:        end for
19:    end for
20:    return E
21: end function
```

Fig. 4. Algorithm for determining and weighting the critical elements

3.2 Behaviorally Correct Path (BCP)

To connect inputs with outputs, a path containing critical elements defined in the previous section needs to be found, we refer this path as a *behaviorally correct path (BCP)*.

In semantic networks, concepts are usually connected by heterogeneous links, including hierarchical relations as well as other relations. For similarity measuring purpose, it is necessary to have a unique path between two elements, and such path should not only contain the critical elements, but also be behaviorally correct.

In [2], Aleksovski et al. considered a path to be semantically correct if and only if no hierarchical links appear after a non-hierarchical one. For example, in figure 1, a path *{ShortStory, is_a, Book, writtenBy, Writer}* is semantically correct, while *{ShortStory, is_a, Book, writtenBy, Writer, is_a, Person}* is not.

In practice, however, there is a high possibility that no semantically correct path exists between two concept using Aleksovki's definition. Therefore, for the purpose of this paper, we define a behaviorally correct path as:

Definition 2. A *Behaviorally Correct Path (BCP)* is a path in a semantic network between two concepts containing *critical elements* with maximum one turn from non-hierarchical relation to hierarchical relation.

And two assumptions must be hold to ensure the existence of a BCP:

1. *Any relation in an ontology is invertible.*
   Relations have directions from range to domain. This assumption implies

that the graph representation of an ontology is undirected as for each relation there exists a inverse relation, e.g. if a relation *contains(Newspaper, Article)* exists, although not all articles are contained in newspapers, we assume a relation *ContainedIn(Article,Newspaper)* also exists.

2. *All relations are inheritable from a super-concept to a sub-concept.* This assumption implies that if there exists a relation p between concepts x and y, i.e. *p(x, y)*, and *is_ a(z, x)*, then *p(z, y)*. This eliminates the sequence of subsumption relations that might be appeared in the beginning of a BCP and also reduces the length of BCPs.

Together, definition 2, assumption 1 and 2, ensure that there always exist a behaviorally correct path between two concepts.

```
1: function SBG(O, S)
2:     SBG ← ∅
3:     CE ← CriticalElements(O, S)
4:     if |CE| > 0 then
5:         for all o ∈ S_(O) do
6:             Paths ← ∅
7:             for all i ∈ S_(I) do
8:                 Paths.append(BCP from o to i with maximum average weight)
9:             end for
10:            SBG.append(Path with maximum average weight in Paths)
11:        end for
12:    else
13:        SBG = S_(I) ∪ S_(O)
14:    end if
15:    return SBG
16: end function
```

Fig. 5. SBG Discovery

Figure 5 shows how a SBG is discovered. Firstly, if there are critical elements determined, for each pair of inputs and outputs, a BCP with maximum average weight is used to represent their behavioral connection. As not all input concepts contribute to the main behavior of the service, the path finding starts from output concepts and for each output concept, only one input concept is associated. The service behavioral graph is thus a set containing these paths.

If no critical elements can be identified, SBG will simply be a union of input and output concepts.

The SBGs of S_1 and S_2 were depicted in figure 3

4 Service Similarity

Paolucci et al. defined four degrees of matching: EXACT, PLUG-IN, SUBSUMES and FAIL, in their approach in [11] based on the hierarchical relation between

I/O pairs of service advertisement and request. They reflect the probability of conducting operation correctly of an advertised service and the satisfaction of its results with certain request. This approach guarantees the matched services can be invoked and operated correctly at lowest level, we will use these degrees as the baseline of our approach.

Algorithm in figure 6 computes the degree of match using approach from [11] at the beginning. This step eliminates the services that cannot be invoked and operated correctly even though their behaviors might be similar to certain extent. Also, this step guarantees that in the worst case, if no critical elements were found in the previous SBG discovery phase, i.e, SBGs are simply sets of input elements and output elements, the result is equivalent to Paolucci's approach.

```
1: function ServiceMatch(SBG^R, SBG^A)
2:     hierarchicalDegree ← HIERARCHICALMATCH(S^R, S^A)
3:     behavioralDegree ← 0
4:     if hierarchical = FAIL then
5:         return < FAIL, −1 >
6:     end if
7:     if SBG^R = S^R_(I,O) or SBG^A = S^A_(I,O) then
8:         return < hierarchicalDegree, −1 >
9:     end if
10:    for all Paths pr and pa in SBG^R and SBG^A do
11:        degree ← MAXPATHMATCH(pr, pa)
12:        if degree > behavioralDegree then
13:            behavioralDegree ← degree
14:        end if
15:    end for
16:    return <hierarchicalDegree, behavioralDegree>
17: end function
```

Fig. 6. Calculation of degree of match

The result of our approach is a pair, for example using the services presented in figure 2, the degree of match is <EXACT, 0.375>, the first element of this pair is the degree of match using Paolucci's approach, and the second element is the behavioral difference of two services, this structure provides requester more flexibility on interpreting the degree of match depends on their needs and environment.

This difference is computed based on the differences of paths where -1 indicates no behavioral matching has been done, 0 indicates an exact match. As a path is a sequence of concepts and relations, the differences of two paths can be defined as their edit distance. We use Levenshtein distance [10] in this paper as presented in figure 7, other distance metrics could also be used here such as Longest Common Sub-sequence (LCS).

```
1: function PathMatch(P^R, P^A)
2:     degree ← EditDistace(P^R, P^A)
3:     if degree = 0 then
4:         return 0
5:     else
6:         return degree / (P^A.length + P^R.length)
7:     end if
8: end function
```

Fig. 7. Distance between paths

5 Conclusion and Future work

This paper presents a novel but preliminary approach of calculating the degree of match between two services. This approach intends to reveal the behavioral information of services, and by comparing their similarity to achieve higher accuracy, robustness and in agreement with human intuition. The main notion behind this approach is that we consider a service as a sub-graph of semantic network that connects its inputs concepts and output concepts via critical elements, referred as *Service Behavioral Graph (SBG)*. We use syntactical information and conditions to determine the critical elements, and a SBG is discovered by exploiting these elements.

Experiments with actual realistic test cases are necessary to access the practicability of our approach. One expectable limitation of our approach is that it depends on the quality (in term of richness) of the ontology to a large extent which is highly unstable in practice. Also, in open environments, services may not use the same ontology to describe its functionality, so semantic alignments need to be performed. Our future work includes implementation, experiments and evaluation of this approach, also solving open issues such as efficient calculation of SBGs, reduction of the deviation caused by the instability of the quality of ontologies and refine the degree of match.

References

1. Owls-tc version 2.2 revision 2. http://projects.semwebcentral.org/projects/owls-tc/.
2. Z. Aleksovski, W. ten Kate, and F. van Harmelen. Exploiting the structure of background knowledge used in ontology matching. In *Ontology Matching Workshop at International Semantic Web Conference (ISWC)*. Citeseer, 2006.
3. G.G. Chowdhury. *Introduction to modern information retrieval*. Facet, 2004.
4. L. Clement, A. Hately, C. von Riegen, T. Rogers, et al. UDDI Version 3.0. 2. *UDDI Spec Technical Committee Draft*, 20041019, 2004.
5. J. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, pages 237–254, 2006.

6. D. Grigori, J.C. Corrales, and M. Bouzeghoub. Behavioral matchmaking for service retrieval: Application to conversation protocols. *Information Systems*, 33(7-8):681–698, 2008.
7. A. Günay and P. Yolum. Structural and semantic similarity metrics for web service matchmaking. In *Proceedings of the 8th international conference on E-commerce and web technologies*, pages 129–138. Springer-Verlag, 2007.
8. K.S. Jones et al. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 60:493–502, 2004.
9. M. Klusch, B. Fries, M. Khalid, and K. Sycara. Owls-mx: Hybrid owl-s service matchmaking. In *Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*, volume 142, 2005.
10. V. Levenshteiti. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics-Doklady*, volume 10, 1966.
11. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. *The Semantic Web (ISWC 2002)*, pages 333–347, 2002.
12. K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In *Proceedings of the International Conference on Web Services*, pages 395–401. Citeseer, 2003.