Deciding semantic matching of stateless services

Andrey Bovykin[1] and **Evgeny Zolin**[2]

[1] Department of Computer Science, University of Liverpool, UK, `andrey@csc.liv.ac.uk`

[2] School of Computer Science, University of Manchester, UK, `ezolin@cs.man.ac.uk`

Abstract

We present a novel approach to describe and reason about stateless information processing Semantic Web Services. It can be seen as an extension of standard descriptions which makes explicit the relationship between inputs and outputs and takes into account OWL ontologies to fix the meaning of the terms used in a service description. This allows us to define a notion of *matching* between services that yields high precision and recall for service location. We explain why matching of these kinds of services is *decidable*, and provide examples of biomedical web services to illustrate the utility of our approach.

1 Introduction

Many of the tools and databases for analysing the data generated from genome sequencing projects like the Human Genome Project are available via Web Service interfaces. They allow biomedical scientists to use the Web as a platform to perform so-called *in silico* experiments. Large numbers of *in silico* experiments are carried out by choosing some of these Web Services, composing them into a workflow, and running them—an approach which shows considerable promise for molecular biology [3] whilst challenging current Web Service approaches. Nowadays, a wide variety of domain ontologies exist which capture the knowledge of biologists (see `http://obo.sourceforge.net/`).

The majority of these services are *stateless*, i.e., they provide information, but do not change the state of the world. Hence their descriptions do not need to include pre- and post-conditions. Here we restrict our attention to these kinds of services since they are quite common yet easier to represent, and since it turned out that defining a semantics or specifying automated reasoning algorithms for world-altering services is basically impossible in the presence of any expressive ontology [9].

The question we are interested in here is how to help a biologist to find a service he or she is looking for, i.e., a service that works with inputs and outputs the biologist can provide/accept, and that provides the required functionality. The growing number of publicly available biomedical web services, 3 000 as of February 2006, requires better matching techniques to locate services. Thus, we are concerned with the question of how to describe a service request Q and service advertisements S_i such that the notion of a service S *matching* the request Q can be defined in a "useful" way. By useful, we mean the following: (1, precision) only those services should match the request that indeed provide the requested functionality; (2, recall) all services providing the requested functionality should match the request; (3) service advertisements and requests should be formulated using terms from existing (OWL) ontologies; and (4) such that the matching problem can be decided automatically.

In this paper, we will propose *a framework to describe in formation providing stateless services that takes into account background ontologies and that allows services to be matched automatically with high precision and high recall.*

2 Services as queries

From a syntactic viewpoint, this framework can be seen as an extension of the way services are described in the OWL-S Service Profile (namely, of its part concerning description of inputs and outputs). Semantically, the service matching conditions we formulate yield the service discovery of higher precision and recall.

In our framework, a description of a service contains, in addition to the types of its inputs and outputs, an explicit specification of the *relationships* between them. Analysing numerous examples of services—including those in bioinformatics, see Section 3—it was observed that the notion of *conjunctive query* can be adopted for these purposes. Before introducing this "services as queries" approach formally, let us illustrate it with a simple example from [6] (realistic examples from the bio-informatics domain are given in Section 3).

Let S_1 and S_2 be services both having an input of type GeoRegion and an output of type Wine, and suppose that S_1 (resp., S_2) returns the list of wines that are *produced* (resp., *sold*) in the region with which it was called. If the types of inputs and outputs are the only information available to match a request to a service, then no matching algorithm can distinguish between S_1 and S_2, and thus matching cannot be precise—see (1) above.

Next, assume that a user requests a service Q that takes a FrenchGeoRegion as input and returns the list of FrenchWines that are *produced* in this region. Even though the service S_1 returns, in general, wines that may not be FrenchWines, it returns only FrenchWines when called with a FrenchGeoRegion, and thus should be matched to this request. A matching algorithm that does so has a high recall—see (2) above.

More formally, when run with a GeoRegion g, the service S_1 returns all those wines w *for which there exists some winegrower* f *who produces* w *and who is located in* g. In our framework, this service can be described as follows:

```
INPUT  g:GeoRegion
OUTPUT w:Wine
THERE IS SOME f   [ WineGrower(f),
      LocatedIn(f,g), Produces(f,w) ]
```

The terms Wine, LocatedIn, etc., are defined in some ontology. In contrast, the service S_2 returns all those wines w *for which there exists some shop* s *who sells* w *and who is located in* g, and can thus be described as follows:

```
INPUT  g:GeoRegion
OUTPUT w:Wine
THERE IS SOME s
 [ Shop(s), LocatedIn(s,g), Sells(s,w) ]
```

In what follows we show that matching service descriptions of this kind is reducible to query containment w.r.t. an ontology—a task whose decidability and complexity is relatively well understood.

2.1 Describing services

We assume the reader to be familiar with OWL-DL and its semantics [11]. Throughout this paper, we borrow the term *TBox* for a class-level ontology (i.e., a finite set of OWL-DL axioms) and *ABox* for a factual ontology (i.e., a finite set of OWL-DL facts). The union of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ is called a *knowledge base* and denoted by $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$. We use $\mathcal{KB} \models \Psi$ to denote the fact that $\mathcal{KB}$ implies Ψ, i.e., Ψ holds in every interpretation that satisfies $\mathcal{KB}$.

Definition 1 (Service syntax). A *service description* $S{=}\langle \vec{x} : \vec{X}; \vec{y} : \vec{Y}; \Phi(\vec{x}, \vec{y}, \vec{z}) \rangle$ consists of

- a list $\vec{x} : \vec{X} = \langle x_1 : X_1, \ldots, x_m : X_m \rangle$ of pairs of variables x_i and classes X_i; this list enumerates *input* variables and their "types";
- a list $\vec{y} : \vec{Y} = \langle y_1 : Y_1, \ldots, y_n : Y_n \rangle$ of pairs of variables y_j and classes Y_j; this list enumerates *output* variables and their "types";
- a *relationship specification* Φ of the form
  $$\mathit{term}_1(\vec{x}, \vec{y}, \vec{z}) \wedge \ldots \wedge \mathit{term}_k(\vec{x}, \vec{y}, \vec{z}),$$
  where each $\mathit{term}_i(\vec{x}, \vec{y}, \vec{z})$ is either an expression of the form $w : C$ with C a class or wRw' with R a property, and w, w' variables from $\vec{x}, \vec{y}, \vec{z}$ or individual names.

Definition 2 (Service semantics). A service s *implements* a service description S over a TBox $\mathcal{T}$ if, for any ABox $\mathcal{A}$ and any tuple of individuals $\vec{a}$ in $\mathcal{T}, \mathcal{A}$, if $\mathcal{T}, \mathcal{A} \models \vec{a} : \vec{X}$, then

1. s accepts $\vec{a}$ as input and
2. when run with $\vec{a}$ as input, it returns the set of all those tuples of individuals $\vec{b}$ from $\mathcal{A}$ such that $\mathcal{T}, \mathcal{A} \models \vec{b} : \vec{Y} \wedge \exists \vec{z}\ \Phi(\vec{a}, \vec{b}, \vec{z})$.

Intuitively, this means that the service s must accept all tuples $\vec{a}$ that conform the input type $\vec{X}$ declared in S, and return as its output the set of those instances $\vec{b}$ of the output type $\vec{Y}$ that bear the relationship Φ to the input tuple $\vec{a}$.

2.2 Matching services

Matching is the problem of determining whether a given service description S conforms to another service description Q. Matching algorithms can be used for *service discovery* purpose, and we can think of S as being a service advertisement and of Q as being a service requested by a user. Let us first give a formal definition and then provide explanations. Here we use $|\vec{x}\,|$ to denote the length of a vector $\vec{x}$.

Definition 3. Given two service descriptions:

$$S = \langle\, \vec{x} : \vec{X};\ \ \vec{y} : \vec{Y};\ \ \Phi(\vec{x}, \vec{y}, \vec{u}\,)\,\rangle,$$
$$Q = \langle\, \vec{z} : \vec{Z};\ \ \vec{w} : \vec{W};\ \Psi(\vec{z}, \vec{w}, \vec{v}\,)\,\rangle,$$

with $|\vec{x}\,| = m = |\vec{z}\,|$ and $|\vec{y}\,| = n = |\vec{w}\,|$, we say that the service S *matches* the request Q w.r.t. the TBox $\mathcal{T}$ if there exist two permutations

$$\begin{aligned} \pi &: \{1,\ldots,m\} \to \{1,\ldots,m\} \\ \rho &: \{1,\ldots,n\} \;\to \{1,\ldots,n\} \end{aligned}$$

such that the following two conditions hold:

(i) $\mathcal{T} \models Z_{\pi(i)} \sqsubseteq X_i$, for all $1 \leqslant i \leqslant m$.

Intuitively, this means that we can map the inputs from S to the inputs from Q such that all input data that the user intends to provide will be accepted by S.

(ii) for any ABox $\mathcal{A}$ and any individuals $\vec{a}, \vec{b}$ in the knowledge base $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$, if $\mathcal{KB} \models \vec{a} : \vec{Z}$, then the equivalence holds:

$$\begin{aligned} &\mathcal{KB} \models \rho(\vec{b}) : \vec{Y} \;\wedge\; \exists \vec{u}\; \Phi(\pi(\vec{a}), \rho(\vec{b}), \vec{u}\,) \quad \text{iff} \\ &\mathcal{KB} \models \vec{b} : \vec{W} \;\wedge\; \exists \vec{v}\; \Psi(\vec{a}, \vec{b}, \vec{v}\,), \end{aligned}$$

where $\pi(\vec{a})$ and $\rho(\vec{b})$ are the permutations of $\vec{a}$ and $\vec{b}$ according to π and ρ.

Intuitively, this means that, modulo some re-arrangement of the input and output vectors, the services S and Q return the same answers on any input that conforms to the request Q.

The need to permute inputs and outputs of Q to "fit" the ones of S is by no means new—it is present in any reasonable definition of service matching. Thus, in order to check whether S matches Q, a reasoning system must "guess" two appropriate permutations or exhaustively explore all possible assignments. Condition **(i)** is quite standard; for example, it can be found in definitions for matching of OWL-S services [5]. In contrast, condition **(ii)** is—to the best of our knowledge—new, and it is not expressible in terms of OWL-S service profiles.

The above definition covers only the case when S and Q have the same number of inputs and the same number of outputs. Various generalisations and extensions are presented in more detail in the technical report [1]. Therein, the reader can also find the proof of the following main statement for our approach.

Theorem. *The service matching problem w.r.t. an ontology is decidable. More precisely, it is reducible to the subsumption of conjunctive queries w.r.t. a TBox.*

The decidability and complexity of the latter problem is extensively explored for many practically interesting Description Logics (cf. [4, 7, 8, 10]; see also an overview of these results in [1]).

3 Describing and matching biomedical services

In this section, we will show the applicability of our approach to the realistic examples of web services from the biomedical domain.

3.1 Matching atomic services

Consider the following two services which extract the DNA sequence from a GenBankRecord.

```
S1: INPUT  x:GenBankRecord
    OUTPUT  y:DNASeqRepresentation
    [ hasPart(x,y) ]

S2: INPUT  x:GenBankRecord
    OUTPUT  y:DNASeqRepresentation
    THERE IS SOME d,e
   [ DNASequence(d), EMBLRecord(e),
   about(x,d), about(e,d), hasPart(e,y)]
```

They coincide on their inputs and outputs, yet they will behave in slightly different ways. The first service simply extracts the DNASequence from the input, whereas the second one first extracts the DNA sequence and then translates the syntax from GenBankForm to EMBLForm. Since there at least 20 different formats for representing DNA sequences, we have to distinguish between a DNA sequence and its representation in one of these formats.

Now consider the following request, which describes services taking a GenBankRecord and returning the corresponding DNA sequence in EMBL format:

```
Q: INPUT  x:GenBankRecord
   OUTPUT  y:DNASeqRepresentation
   THERE IS SOME d,e
[DNASequence(d), Record(e), about(x,d),
 about(e,d), hasPart(e,y), EMBLform(y)]
```

Note that, according to our definition of matching, the service S1 does not match our request Q since it cannot guarantee that the output is in EMBL format. In contrast, if our TBox contains

```
SubClassOf(EMBLRecord
           restriction(hasPart
           allValuesFrom EMBLform))
```

which ensures that all entries in an EMBLRecord are in EMBLform, then service S2 matches our request—which is indeed useful. Similarly, in the presence of the above OWL axiom, S2 even matches the following request—despite the fact that the output of this request is more specific than that provided by the service S2.

```
Q1: INPUT  x:GenBankRecord
   OUTPUT  y:IntersectionOf(
        DNASeqRepresentation  EMBLform)
   THERE IS SOME d,e
  [ DNASequence(d), EMBLRecord(e),
  about(x,d), about(e,d), hasPart(e,y)]
```

3.2 Matching complex services

Next, we consider a request for a service, which takes a BlastReport and extracts its DNA sequence representation:

```
Q2: INPUT  x:BlastReport
   OUTPUT  y:DNASeqRepresentation
   [ hasPart(x,y) ]
```

This is a rather simple request, but consider the following Web Service:

```
S3: INPUT  x:BlastReport
   OUTPUT  y:DNASeqRepresentation
   THERE IS SOME p [ hasPart(x,p),
   PairWiseAlignm(p), hasPart(p,y) ]
```

If our TBox contains a statement that hasPart is transitive, then S3 matches Q2. Similarly, if we consider the following two services, then their composition S4 $\circ$ S5 matches our request.

```
S4: INPUT  x:BlastReport
   OUTPUT  y:PairWiseAlignmnt
   [ hasPart(x,y) ]

S5: INPUT  x:PairWiseAlignmnt
   OUTPUT  y:DNASeqRepresentation
   [ hasPart(x,y) ]
```

In contrast, the following service does not match Q2 because, besides extracting the sequence, it also computes its reverse complement.

```
S6: INPUT  x:BlastReport
   OUTPUT  y:DNASeqRepresentation
   THERE IS SOME p,z,w [ hasPart(x,p),
    PairWiseAlignmnt(p), hasPart(p,z),
    complementOf(z,w), reverseOf(w,y) ]
```

Let us point out again that our definition of matching yields both a higher precision and a higher recall than any comparison of inputs and outputs could possibly yield: it matches services whose inputs or outputs do not match in an obvious way (such as Q2 and S2 above), and it does not match services despite their in- and outputs matching (such as S6 and Q2). The latter point is especially important for biomedical Web Services since many take strings as in- and outputs—and thus all services would match on the grounds of in- and outputs.

Acknowledgments

The work is supported by an EPSRC grants GR/S63168/01 and GR/S63182/01 as part of the DynamO project. For more information, please visit http://dynamo.man.ac.uk/.

References

[1] A. Bovykin, E. Zolin. *Describing information providing services.* Technical Report, University of Manchester 2005. http://dynamo.man.ac.uk/publ/aaai06tr.pdf

[2] R. D. Stevens, H. J. Tipney, C. Wroe, T. Oinn, M. Senger, P. W. Lord, C. Goble, A. Brass, and M. Tassabehji. Exploring Williams-Beuren syndrome using myGrid. *Bioinformatics*, vol. 20. 2004.

[3] L. Stein. Creating a bioinformatics nation. *Nature*, 417:119–120, 2002.

[4] Tessaris, S. *Questions and answers: reasoning and querying in Description Logic.* PhD thesis, Univ. of Manchester, 2001.

[5] T. R. Payne, M. Paolucci, and K. Sycara. Advertising and Matching DAML-S Service Descriptions. *Semantic Web Working Symposium (SWWS)*, 2001.

[6] The DAML Services Coalition. Bringing Semantics to Web Services: the OWL-S approach. In *Proc. of SWSWPC'2004.*

[7] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proc. of LPAR'2000.*

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, 149–158.

[9] F. Baader, C. Lutz, M. Miličić, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. of AAAI'2005.*

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of DL'2005.*

[11] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *J. of Web Semantics*, 1, 2003.