A Framework for Describing Information Providing Web Services

Andrey Bovykin[⋆]
[⋆]Department of Computer Science
University of Liverpool, UK
andrey@csc.liv.ac.uk

Evgeny Zolin[†]
[†]School of Computer Science
University of Manchester, UK
ezolin@cs.man.ac.uk

Abstract

We introduce a formal framework for describing stateless *information providing* Semantic Web Services using Description Logic. The extended notion of service description includes, besides the types of its inputs and outputs, a specification of relationships between inputs and outputs, which has the form of a conjunctive query. From syntactic point of view, it is an extension of the way services are described in OWL-S Service Profile. However, semantically, the definition of *service matching* we introduce here yields a high precision service discovery algorithm. We show that the reasoning problem of service matching for this kind of descriptions is reducible to checking subsumption between two conjunctive queries w.r.t. an ontology, which is a standard reasoning task.

1 Introduction

The aim of this work is to provide a formal framework for describing stateless information providing web services. When such a service is executed, it accepts from a user an input data of a specified format ("typed data") and returns back to the user some information as an output. Most services of this kind are *stateless*, i.e., they only provide information about the current state of the world, but do not change that state.

We develop our framework in the context of *service discovery* problem: given a description of a service Q that is to be found (called a *service request*), a search engine tries to match the request to service descriptions stored in a repository (they are called *service advertisements*), using its reasoning system and based on the background *ontologies* to which the request and the advertisements refer to. Therefore, our task is to formulate the reasonable conditions of *service matching* that allows for high precision service discovery and such that the matching problem can be decided automatically.

Let us point out the distinguishing features of this framework. Firstly, a description of a service is entirely based on standard background ontologies (i.e., we do not introduce any concepts or roles specially dedicated to describing web services). Therefore we can reuse the existing end emerging ontologies, and, most important, we can use the semantics defined by these ontologies for service matching purposes. Secondly, since the services we describe are stateless, their descriptions need not to contain pre- or post-conditions. Thirdly, the notion of service matching formalised in our framework is decidable and reducible to a standard reasoning task, so one can reuse the existing reasoning systems that are capable to solve the corresponding reasoning problem.

2 Describing services

Definition 2.1 (Service description). A *service description* is an expression of the form $S := \langle \vec{x} : \vec{X};\ \ \vec{y} : \vec{Y};\ \ \Phi(\vec{x}, \vec{y}) \rangle$, where $\vec{x} : \vec{X}$ and $\vec{y} : \vec{Y}$ are tuples of *input* and *output* variables together with their types (which are DL concepts), and $\Phi(\vec{x}, \vec{y})$ is a *conjunctive query*, i.e., an expression of the form $\exists \vec{z}\ \big(\mathit{term}_1(\vec{x}, \vec{y}, \vec{z}) \wedge \ldots \wedge \mathit{term}_k(\vec{x}, \vec{y}, \vec{z})\big)$, where each conjunct $\mathit{term}_i(\vec{x}, \vec{y}, \vec{z})$ is either an expression of the form $w : C$ with C being a concept, or wRw' with R being a role, and w, w' are variables from the lists $\vec{x}, \vec{y}, \vec{z}$, or individual names.

The meaning of the whole expression is that, given a tuple of individuals $\vec{a}$ from $\vec{X}$ as an input, the service S returns as its output the (unordered) set of all tuples of individuals $\vec{b}$ that belong to $\vec{Y}$ and satisfy the condition $\Phi(\vec{a}, \vec{b})$.

Let us illustrate this definition by an example adopted from (3). Consider two services S and Q, both with an input of type GeoRegion and an output of type Wine. The service S takes a name of a geographical region (e.g., 'France') as an input and retrieves the list of (names of) wines that are *produced* in this region. The service Q, on the contrary, returns the list of wines that are *sold* in that region. In our framework, they will be described as follows:

$$S = \langle\, x{:}\,\mathsf{GeoRegion};\ \ y{:}\,\mathsf{Wine};\ \exists z\,(\, z{:}\,\mathsf{WineGrower}\ \wedge\ z\,\mathsf{isLocatedIn}\,x\ \wedge\ z\,\mathsf{produces}\,y\,)\,\rangle$$
$$Q = \langle\, x{:}\,\mathsf{GeoRegion};\ \ y{:}\,\mathsf{Wine};\ \exists z\,(\, z{:}\,\mathsf{Shop}\ \wedge\ z\,\mathsf{isLocatedIn}\,x\ \wedge\ z\,\mathsf{sells}\,y\,)\,\rangle$$

Using OWL-S Service Profile, one cannot distinguish between these two services. However, the notion of service matching introduced below takes into account their functionality and hence allows to distinguish between them.

2.1 Service matching

Definition 2.2 (Service matching). Given two service descriptions with $|\vec{x}\,| = m = |\vec{z}\,|$ and $|\vec{y}\,| = n = |\vec{w}\,|$:

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

we say that the service S *matches* the request Q w.r.t. the ontology $\mathcal{T}$ if there exist two permutations $\tau\colon \{1, \ldots, m\} \to \{1, \ldots, m\}$ and $\sigma\colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that the following two conditions hold:

(i) Applicability: $\mathcal{T} \models X_i \sqsupseteq Z_{\tau(i)}$, for all $i \leqslant m$, i.e., the type of x_i subsumes the type of $z_{\tau(i)}$ w.r.t. the ontology $\mathcal{T}$.
Intuitively: one can map the inputs of S to inputs of Q so that all the user's input data will be acceptable by S.

(ii) Coherence: for any ABox $\mathcal{A}$ and any tuples of individuals $\vec{a}\,, \vec{b}$ in the knowledge base $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A}\rangle$ with $|\vec{a}\,| = m$ and $|\vec{b}\,| = n$, if $\mathcal{KB} \models \vec{a}\colon \vec{Z}$, then $\mathcal{KB} \models \sigma(\vec{b}\,)\colon \vec{Y} \wedge \Phi(\tau(\vec{a}\,), \sigma(\vec{b}\,))$ iff $\mathcal{KB} \models \vec{b}\colon \vec{W} \wedge \Psi(\vec{a}\,, \vec{b}\,)$.
Intuitively: modulo some re-arrangement of the input and output vectors, the services Q and S return the same answers on any input that conforms to the user's request Q.

Theorem 1. *The service matching problem is reducible to subsumption of conjunctive queries w.r.t. an ontology.*

2.2 Service composition

Suppose that we are given two services $S = \langle x\colon X; y\colon Y; \Phi(x, y)\rangle$ and $S' = \langle x'\colon X'; y'\colon Y'; \Phi'(x', y')\rangle$. Our task is to formulate the conditions when the composition $S \circ S'$ (to be read as 'first S runs, then S' runs on the output produced by S') is meaningful (i.e., when these services are compatible) and when it matches a user's request $Q = \langle z\colon Z;\ w\colon W;\ \Psi(z, w)\rangle$.

Definition 2.3 (Service composition). A composition of services $S \circ S'$ *matches* a request Q w.r.t. a TBox $\mathcal{T}$ if:

(a) Applicability: $\mathcal{T} \models X \sqsupseteq Z$. This ensures that S accepts all inputs described in the request Q.

(b) Compatibility: $\mathcal{T} \models \forall x, y\big(x\colon Z \wedge \Phi(x, y) \wedge y\colon Y \to y\colon Y'\big)$. The outputs of S (on user's inputs) are accepted by S'.

(c) Coherence: $\mathcal{T} \models \forall x, w\ \big(\Psi(x, w) \wedge w\colon W \iff \exists y\ \big(\ \Phi(x, y) \wedge y\colon Y \wedge \Phi'(y, w)\ \big) \wedge w\colon Y'\big)$.
On the user's inputs, the application of S and then S' yields the same answers as Q.

Theorem 2. *Matching a composition of services to another service is reducible to query supsumption w.r.t. an ontology.*

2.3 Extensions and generalizations

The definition of service matching given above covers only the case when the number of inputs (and outputs) of the service S is equal to that of the service Q. It is almost straightforward to generalize the definition of service matching to the case when user's request Q has "redundant" inputs and/or the service S has "redundant" outputs. There are several possible approaches on how to deal with the remaining cases (when Q has less inputs than S requires, or S has less outputs that Q requires). The framework is also extended to describe services with *structured output* (e.g., linearly ordered or equipped with any other additional structural information) and with *boolean* output. There is an ongoing work on an extension of this framework to describe services whose inputs and outputs are values from concrete domains (i.e., integers, strings, etc.). All these extensions are considered in more details and illustrated by examples in the technical report (1).

Acknowledgements

The research is supported by an EPSRC grant GR/S63182/01, GR/S63168/01 as part of the DynamO project. The authors would like to thank Ian Horrocks, Ulrike Sattler, and Frank Wolter for their help during the research.

References

[1] A. Bovykin and E. Zolin. A Framework for Describing Information Providing Web Services. Tech. report. University of Manchester, 2005. `http://dynamo.man.ac.uk/publications/services_as_queries_TR.pdf`

[2] Terry R. Payne, Massimo Paolucci, and Katia Sycara. Advertising and Matching DAML-S Service Descriptions. Semantic Web Working Symposium (SWWS), 2001.

[3] The DAML Services Coalition. Bringing Semantics to Web Services: The OWL-S Approach. Proc. of the 1st Int. Workshop on Semantic Web Services and Web Process Composition (SWSWPC'2004), July 6-9, 2004, San Diego, California, USA. `http://www.daml.org/services/owl-s/`