Towards an Algebraic Composition of Semantic Web Services

Florian Lautenbacher and Peter Höfner

Institute of Computer Science, University of Augsburg, Germany
`[lautenbacher|hoefner]@informatik.uni-augsburg.de`

Abstract. To realise a software that needs to interact with other computers, service-oriented architecture and especially Web Services are gaining more and more attention. Existing components need to be composed in order to reach a predefined goal. For Web Services this can be realised using data defined in WSDL or using semantic Web Services such as SAWSDL. We apply relation algebra to describe Web Services as well as the composition of existing semantic Web Services.

1 Introduction

Most companies nowadays have business processes where frequent interaction with other companies happens on a regular basis. Hence, the involved systems must be able to interact somehow in order to realise predefined goals and finally to produce goods or services. This is only possible if every system can read the exchanged messages which means that the output of one involved system must be compatible with the input of another. The service-oriented architecture (SOA) and -paradigm can be applied to couple the systems in services which then interchange messages and can be discovered using a shared directory.

Web Services are one way to realise a SOA. The Web Services need to be described in a language such as the Web Service Description Language (WSDL) and message exchange happens using the SOAP standard. But these standards have shortcomings: since the exchanged messages are only described using human-comprehensible terms, they can be understood by different developers in various ways. Hence, standards of the semantic web can be applied to provide a unique meaning for those terms in all systems using ontologies.

The W3C has defined an extension to WSDL which is called SAWSDL, Semantic Annotations for WSDL and XML Schema, to support the use of concepts in an ontology for the definition of Web Services. Additionally, several attempts of semantic Web Services have been developed during the last years: OWL-S, SWSF or WSMO (e.g. [3]) to name just a few. The ontologies defined in these standards can then be combined using SAWSDL (see e.g. [7]).

An existing Web Service description can be used to achieve an automatic composition. To achieve a goal, several Web Services need to be composed and must interact with each other. In this paper we describe an algebra for the description of (semantic) Web Services and show how this algebra can assist the user in the composition of Web Services.

2 (Semantic) Web Service Standards

Each Web Service needs to have a specified interface in order to interact with others. WSDL [2] is a language designed to describe such an interface. A WSDL document consists of four parts which describe different levels of abstraction: `types`, `interfaces`, `bindings` and `services`. `Types` contain definitions for the datatypes used in messages. `Interfaces` describe which operations are supported by a service. `Bindings` and `services` specify the protocols and endpoints used to access the service. Input, output and fault messages specify operations of `interfaces`. They rely on data types described in `types`. Basic data types are defined by an XML Schema. They can be nested to more complex types.

SAWSDL [5], recommended by the W3C since 2007, is a set of extensions for WSDL to provide a standard description format. It allows a semantic annotation of a Web Service description using three different kinds of attributes: with `modelReference` one can create an association between a part in the WSDL document and a concept in an ontology. Type definitions, element declarations, attribute declarations as well as interfaces can be annotated. Using the attribute `liftingSchemaMapping` one can describe the mapping between the WSDL file and semantic data whereas `loweringSchemaMapping` specifies the mapping between a semantic concept and the constructs in the WSDL file.

3 A Formalisation of Web Services

In [4], we defined Web Service composition using relation algebra (e.g [9]). In this section we will recapitulate the basic definitions.

In the interfaces of WSDL several Web Methods are defined. These receive input messages and reply with output messages which both can be of a simple type such as string, integer, etc. or of a complex type. We will assume that the types of the input and the output data are known before and therefore there is a *knowledge set* $\mathcal{K}$, a set which includes the inputs and outputs as subsets. The concrete binding information or fault messages within WSDL are currently neglected for the sake of simplicity.

Definition 3.1. A *Web Method* is a tuple (I, O), where $I \subseteq O \subseteq \mathcal{K}$.

The condition $I \subseteq O$ guarantees that we do not lose any information, i.e., any information which is known before the execution of a Web Method is also known afterwards. $(\emptyset, O)$ represents a Web Method where no input is needed, i.e., it can be executed at any time.

In the following we will use a grammar-style notation to ease the reading and to focus on the input and the "real" output. In particular, we write $\{I \rightarrow O-I\}$ instead of (I, O). When possible, we will even skip the set brackets.

A Web Method is the simplest form of a Web Service, but it is atomic. Following WSDL, a Web Service may contain more than one Web Method, hence we define the concept of a *simple Web Service*.

Definition 3.2. A *simple Web Service* is a collection of Web Methods.

Operations like choice or sequential composition can also be applied to Web Services. Choice is the set-theoretic union and composition is the multiplication. As described in [4] this definition yields a strange behaviour. Therefore, we define an (extended) Web Service and Web Services composition as follows:

Definition 3.3. The *(extended) Web Service* of a simple Web Service W is the relation $\{(I \cup E, O \cup E) : (I, O) \in W, E \subseteq \mathcal{K} \backslash O\}$ and denoted by $\ll W \gg$. *Web Service composition* of V and W is then defined by $V \circ W =_{df} \ll V \gg ; \ll W \gg$, where ; denotes standard sequential composition of relations.

In this definition E is the context and the extension of the simple Web Service W, which just takes any information that is not needed as input for execution and adds this information unchanged to the output.

It is straightforward to show that extended Web Services are closed under choice and Web Service composition. Therefore they form an idempotent semiring[1]. Finite iteration of Web Services can be determined by the reflexive and transitive closure, denoted by *. It is easy to show that $\ll V^* \gg = \ll V \gg^*$. Henceforce, Web Services form also a Kleene algebra.

4 Web Service Restriction

In order to express the needed input data to perform a certain action or to describe goals we introduce the concept of tests. This allows an automatic composition of Web Services as described in [4].

One defines a *test* in an idempotent semiring [6] to be an element $p \leq 1$ that has a complement q relative to 1, i.e., $p + q = 1$ and $p \cdot q = 0 = q \cdot p$. In relation algebra every subidentity can be seen as a test.

Tests can be used to model assertions for Web Services. But they are also the basis for defining modal operators [1] which are used for modelling termination and to determine whether some goals can be reached by composing Web Services or not. Additionally, they can be used to compute an initial state that allows a composition of Web Services to be executed.

Informally, in the context of Web Services the forward diamond $|a\rangle p$ characterises the set of possible information with at least one successor in p when executing the Web Service a, i.e., the preimage of the set p under a. $|a]p$ characterises the situation where there is no execution of a, that starts in p and terminates in $\neg q$, the complement of q. Whenever an execution of a terminates in $\neg q$, the execution has to start in $\neg p$ and therefore $|a]p$ models the possible infomation from which execution of a is guaranteed to terminate in an element of p or the execution is not possible. The combination of both modal operators ($|a\rangle p$ and $|a]p$) guarantees that at least one result of the Web Service a exists and all resulting information is in p. This can be expressed by $|a\rangle p \cdot |a]p$.

[1] A semiring is called idempotent if $a + a = a$ for all elements.

5 Algebraic Composition of Semantic Web Services

Since all inputs and outputs of a Web Service can be annotated with concepts from an ontology, we have a closer look at these annotations now. Concepts in an ontology (e.g. defined in OWL [8]) can be connected with other concepts through user-defined *ObjectProperties* or *DataProperties*. With `SubClassOf` one can define inheritance and using `EquivalentClasses` the equivalence of two classes is specified. To formalise these concepts we discuss two possibilities:

First alternative

After each Web Service we include an additional one that extends the output parameters of the former with the concepts that can be "reasoned" through the ontology. Analogous to Definition 3.3 we define an ontology-based Web Service composition as follows:

Definition 5.1. *Ontological Web Service composition* of V and W is defined as $V \mathbin{\hat{\circ}} W =_{df} V \circ Ont^* \circ W$, whereas Ont is a set of tuples (s,t) describing relations of concepts in the ontology with $s,t \in \mathcal{K}$ and hence a Web Service itself.

These relations have been stored e.g. in an OWL-file and need to be translated into tuples. After each call of Ont^* the available data has been extended with the `EquivalentClasses` or `SubClassOf` of the output data.

The Kleene star supports also transitive relations in the ontology. Now the second Web Service can be invoked and will find its input data, if it has been computed by V or by Ont^*.

Let us have a look at a small example: Assume Web Services B and Z. The former extracts the `birthdate` of a person from a database, i.e., $B = (\texttt{firstName lastName} \rightarrow \texttt{birthdate})$. Z needs some `date` to calculate the corresponding zodiac sign, i.e., $Z = (\texttt{date} \rightarrow \texttt{zodiacSign})$. In Ont the relations $(\texttt{birthdate}, \texttt{date})$ and $(\texttt{firstName}, \texttt{name})$ have been stored. $B \mathbin{\hat{\circ}} Z$ yields that B and Z can be composed, since `birthdate` and `date` are related to each other in the ontology. The composed Web Service is then $(\texttt{firstName lastName} \rightarrow \texttt{date zodiacSign})$.

Second alternative

For each Web Method we define additional ones by replacing the input data with equivalent classes and sub-classes.

Definition 5.2. Let S, T be sets with $S, T \subseteq \mathcal{K}$. A *refinement relation* is defined as $S \sqsubseteq T \Leftrightarrow_{df} \forall s \in S\ \exists t \in T : (s,t) \in Ont^*$. The *refined Web Service* of W is given by $W_{\sqsubseteq} =_{df} \{(\hat{I} \rightarrow O) \mid (I \rightarrow O) \in W,\ \hat{I} \sqsubseteq I,\ \hat{I} \subseteq \mathcal{K}\}$.

Applying the same example as before, we can now change Z and determine the Web Service $Z_{\sqsubseteq}$. In particular, $Z_{\sqsubseteq}$ includes $(\texttt{birthdate} \rightarrow \texttt{zodiacSign})$. B can be composed with $Z_{\sqsubseteq}$, since the input of B and output of $Z_{\sqsubseteq}$ is identical.

Discussion

The first alternative uses a straight-forward relationship between the ontology and our characterisation of Web Services. That is why the construction is quite natural. Nevertheless, it creates a lot of tuples that are not needed for the composition. Moreover, the output produced through the ontology might lead to confusion since the same information can now occur in different data of the output. For example `birthdate` will occur in `birthdate` and `date`.

The second alternative does not create unnecessary data; but the Web Services must be changed. At first sight this seems more complicated. Nevertheless, this modification has to be done only once. Moreover, if a Web Service accepts a `date` as input, it is obvious that it also accepts `birthdate` as input. Therefore, this approach is also natural.

We prefer the second approach since it reflects the intuition what data a Web Service may accept. Moreover, this approach can be extended with preconditions and effects lateron as needed in the area of semantic Web Services.

6 Conclusion and Outlook

In this paper we presented a method to describe Web Services based on a relation algebra. Additionally, we outlined two ideas how to describe semantic Web Services and ontologies. As this is current research, there are several things that still need to be done: An ontology does not only support `EquivalentClasses` or `SubClassOf`, but also other relationships between classes that we want to take care of. The tuples of the ontology might be extended to tripels to store these information. Also `lowering-` or `liftingSchemaMapping` are currently ignored. We cover only some parts of WSDL in our approach, but most of the others are not necessary for the composition anyway, but only for the enactment lateron.

References

1. J. Desharnais, B. Möller, and G. Struth. Termination in modal Kleene algebra. In J.-J. Lévy, E. W. Mayr, and J. C. Mitchell, editors, *IFIP TCS2004*, pages 647–660. Kluwer, 2004.
2. W. Dostal, M. Jeckle, I. Melzer, and B. Zengler. *Service-orientierte Architekturen mit Web Services (in German).* Spektrum Akademischer Verlag, 2005.
3. D. Fensel, H. Lausen, J. de Bruijn, M. Stollberg, and D. Roman. *Enabling Semantic Web Services - The Web Service Modeling Ontology.* Springer, 2007.
4. P. Höfner and F. Lautenbacher. Algebraic Structure of Web Services. In S. Escobar and M. Marchiori, editors, *WWV 07*, ENTCS (to appear), 2008.
5. J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell. SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Computing*, Nov./Dec.:60–67, 2007.
6. D. Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.
7. D. Martin, M. Paolucci, and M. Wagner. Toward Semantic Annotations of Web Services. In *OWL-S: Experiences and Directions, June, 6*, 2007.
8. B. Motik, P. F. Patel-Schneider, and I. Horrocks. OWL 1.1 Web Ontology Language - Structural Specification. Member subm., W3C, 2006.
9. G. Schmidt and T. Ströhlein. *Relations and Graphs: Discrete Mathematics for Computer Scientists.* Springer, 1993.