Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010) 310–328

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Discovering Semantic Web services using SPARQL and intelligent agents

Marco Luca Sbodio [a], David Martin [b,*], Claude Moulin [c]

[a] Hewlett-Packard Italy Innovation Center, Corso Trapani 16, 10139 Torino, Italy
[b] Artificial Intelligence Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, USA
[c] Université de Technologie de Compiègne, 60205 Compiègne, France

ARTICLE INFO

Article history:
Received 1 July 2009
Received in revised form 30 March 2010
Accepted 21 May 2010
Available online 4 June 2010

Keywords:
Semantic Web services
Service discovery
SPARQL
Intelligent agents

ABSTRACT

This paper describes a novel approach to the description and discovery of Semantic Web services. We propose SPARQL as a formal language to describe the preconditions and postconditions of services, as well as the goals of agents. In addition, we show that SPARQL query evaluation can be used to check the truth of preconditions in a given context, construct the postconditions that will result from the execution of a service in a context, and determine whether a service execution with those results will satisfy the goal of an agent. We also show how certain optimizations of these tasks can be implemented in our framework.

1. Introduction

Service discovery – the identification of services that are capable of accomplishing a given objective – is a central problem in Semantic Web services (SWS) research. Most SWS work on discovery, either explicitly or implicitly, aims to support the autonomous identification of suitable services by software agents, to support the satisfaction of their goals. Effective service discovery depends directly on service descriptions that are adequately expressive. SWS service descriptions, in turn, usually include the specification of *preconditions* and *postconditions*. Preconditions are conditions that must hold true before invoking a service, to ensure successful use of the service, and postconditions are conditions that will hold true after the successful use of a service. The specification of preconditions and postconditions, and the drawing of inferences based upon them, is a distinguishing feature of most work on Semantic Web services.

In this paper, we show how the SPARQL [51] query language can be used to express the preconditions and postconditions of services, as well as the goals of agents. In addition, we show that SPARQL query evaluation can be used to check the truth of a precondition in a given context, construct the postcondition that will result from the execution of a service in a context, and determine whether a service execution with those results will satisfy the goal of an agent. In a nutshell, the truth of a service precondition indicates that the service can successfully be used, the resulting postcondition reveals what will be true after using the service, and the satisfiability of the agent's goal indicates that the service is a candidate for use in accomplishing that goal – thus providing a solution to the discovery problem. We also show how certain optimizations of these tasks can be implemented in our framework.

SPARQL has been standardized at the World Wide Web Consortium, and is by far the most widely used query language for knowledge bases employing the Resource Description Framework (RDF) [38] and/or the Web Ontology Language (OWL) [46]. Additional background on SPARQL is given in Section 3.

To situate our approach in a larger context of practice, we discuss how it may be used with OWL for Services (OWL-S) [44], but the approach is applicable to any SWS framework based on knowledge representation using RDF or OWL. OWL-S is deliberately under-constrained with respect to the specification and use of preconditions and postconditions. That is, it allows for a service description to "escape" into a language other than OWL for the specification of preconditions and postconditions. This path was chosen because OWL is not well-suited for expressing pre- and postconditions, both in terms of its expressiveness and its lack of naturalness for this purpose. For example, OWL's lack of variables makes it difficult to express conditions with a suitable degree of generality, flexibility and naturalness. Thus, in the definition of OWL-S [43], several languages are declared as candidates for use in expressing pre- and postconditions, including, in addition to SPARQL, KIF [21], SWRL [29], and several other possibilities. (This collection of possibilities is meant by the authors of OWL-S to be illustrative, rather than exclusive.) In the examples accompanying the OWL-S documentation, and in subsequent work, SWRL has been used most

* Corresponding author. Tel.: +1 650 859 4119; fax: +1 650 859 3735.
E-mail addresses: marco.sbodio@hp.com (M.L. Sbodio), david.martin@sri.com, martin@ai.sri.com (D. Martin), claude.moulin@utc.fr (C. Moulin).

1570-8268/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2010.05.002

widely – primarily because it was designed for use with OWL. SWRL, however, has its own drawbacks with respect to the expression of pre- and postconditions. These include expressiveness limitations (e.g., no disjunction), tractability, and lack of standardization.[1] Other rules languages raise similar issues, and greater difficulties in terms of integration with OWL. Still more expressive languages, such as KIF, also raise issues of tractability and, in many cases, lack of standardization and tool support.

SPARQL provides a way out of these difficulties. SPARQL queries, as we shall show, allow for a natural, flexible, and expressive formulation of conditions and goals. In addition, since SPARQL is designed to be an integral part of the Semantic Web technology family, its use with RDF and OWL is already well understood and supported by many tools and environments, and its usage is in keeping with OWL-S's objective to remain firmly situated in the world of Semantic Web standards. Additionally, it has been shown [1] that the expressive power of SPARQL is equivalent to that of non-recursive safe Datalog with negation, and hence to Relational Algebra.

OWL-S is focused on describing services and the processes that they encapsulate. Thus, agents have been left implicit in the world of OWL-S; that is, there are no ontology elements for *agent*, *goal*, or certain other central concepts from the world of agents. In many OWL-S research efforts, matchmaking techniques have been studied in isolation from the agents that might employ them. In most of these efforts, there has been more focus on classification (of services and service requests) as the basis of matchmaking, rather than on reasoning about preconditions, postconditions, and goals. We show here that SPARQL, in addition to specifying pre- and postconditions, is also well-suited to the specification of goals, thus filling a gap in the realm of OWL-S usage, and providing a single, standards-based framework that seamlessly handles the representation of these agent concepts along with OWL-S's service concepts, and a mechanism for drawing inferences from them.

In the following section, we first provide a conceptual framework for our approach, in terms of the world states and transitions of modal dynamic logic. Section 3 gives an overview of SPARQL, and then shows how it can be used to characterize Web service operations and agents' goals, with examples. In Section 4, we show, at a high level, how our approach can be used in a belief-desire-intention (BDI) agent framework. Section 5 spells out our SPARQL-based service discovery algorithm, with an example, and then goes on to discuss how it can be optimized for use with a remote registry. Section 5 also shows how SPARQL features can be used to support discovery with relaxation of preconditions and/or goals. Section 6 describes our prototype implementation and the experimental evaluation of our work. Section 7 discusses related work and Section 8 concludes.

1.1. A note on terminology

The structural elements that carry preconditions and postconditions vary across SWS and agent research. For example, in OWL-S it is the *process* that carries preconditions and effects (postconditions). In the Web Services Modeling Ontology (WSMO) [57] a service has a *capability*, and the capability has preconditions and postconditions. (Actually, WSMO distinguishes two kind of preconditions, termed *preconditions* and *assumptions*, and two kinds of postconditions, termed *postconditions* and *effects*, but these distinctions need not concern us here.) In the Web Service Description Language (WSDL) [11], although preconditions and postconditions are not specified, conceptually it is the *operation* to which they could appropriately belong. Because WSDL and the related Semantic Annotations for WSDL and XML Schema (SAWSDL) [18] are standards and familiar to a large audience, in our conceptual framework we use the concept *operation* as the bearer of preconditions and postconditions.

2. Conceptual framework

We assume that a large number of Web services are published in a networked environment (i.e., the Internet, an intranet, or perhaps some kind of virtual organization), and their syntactic and semantic descriptions are held in *registries* (syntactic and semantic descriptions are not necessarily stored in the same registry). Each Web service has a syntactic description expressed in WSDL. Each Web service also has a formal semantic description, which defines for each operation:

semantics of input and output types: input and output types are specified as OWL classes;
preconditions: logical conditions that must hold before invoking the operation;
postconditions: semantic description of the operation's effects; that is, conditions that are guaranteed to hold true after a successful execution of the operation. We use the terms *postconditions* and *effects* interchangeably.

A software agent wants to accomplish some task or to achieve some *goal*, and therefore looks for a Web service that offers an appropriate operation to achieve its intentions. The software agent is situated in an environment; that is, it has a (possibly incomplete) description of the current state of the world. This world state description is given in the agent's *knowledge base* as an RDF graph.[2]

From an abstract point of view, a Web service operation can be seen as an action that the agent can invoke if and only if some conditions hold (the *preconditions*). The preconditions are evaluated against a state description (the current world state, as known by the agent's knowledge base). The execution of the operation causes a state transition, and it has some *effects* or *postconditions*, which express what will be true in the world state resulting from the execution.

Let $\Omega = \{\omega_0, \omega_1, \omega_2, \ldots\}$ be a countably infinite set of world states; let $\Delta = \{\delta_0, \delta_1, \delta_2, \ldots\}$ be a countably infinite set of descriptions given as RDF graphs. The relation $\mathcal{D} = \Omega \times \Delta$ associates every world state with its corresponding RDF descriptions, each of which expresses what is true in that world state. The relation $\mathcal{D}$ captures the intuition that the same world state ω may be associated with multiple descriptions, each one representing a possible view on ω from the perspective of a particular agent. The agent's view on the world state ω corresponds to the content of the agent's knowledge base. Given the same world state w, different agents may have different views on it, depending on their knowledge and perceptions. In general, one may define for each agent α a function $d_\alpha : \Omega \to \Delta$, such that for every world state $w_i \in \Omega$, the result of $d_\alpha(w_i)$ is the RDF graph describing the agent's view on the world state w_i (i.e. the content of the agent's knowledge base when the world state is w_i). In this paper we always refer to a single agent α, and therefore when we write(ω_i, δ_i) we mean that $\delta_i = d_\alpha(\omega_i)$.

Let $\Sigma = \{\sigma_0, \sigma_1, \sigma_2, \ldots\}$ be the set of Web service operations published in a registry. Note that in general two distinct operations

[1] With respect to tractability, SWRL is undecidable, but it should also be noted that a decidable language can be obtained by restricting SWRL to "DL-safe" rules, which would provide a more suitable candidate for expressing preconditions and postconditions.

[2] The description is assumed to be both internally consistent and accurate with respect to the state of the world. The matter of maintaining internal consistency – the belief revision problem – is discussed briefly below.