Towards a First-Order Ontology for Semantic Web Services

Daniela Berardi[*] Michael Grüninger[†] Richard Hull[‡] Sheila McIlraith[§]

September 4, 2004

1 Introduction

We argue that an unambiguously, computer-interpretable description of the process model of a Web service, and a client's constraints on that process model, are critical to automating a diversity of tasks, including Web service discovery, invocation, composition, monitoring, verification and simulation. That the process model descriptions be unambiguously computer-interpretable is key to our argument, and is a shortcoming of a number of existing process modeling frameworks that have been proposed to describe aspects of Web services.

We commence this position paper with a brief overview of some existing process modeling frameworks used within the Web service community, providing analysis of some of their key merits and shortcomings. Next we present a small number of use cases that motivate and illustrate the need for computer-interpretable process models for the automation of Web service discovery and composition. With our use cases in hand, we propose a set of desiderata for a Web service description language that enables an unambiguous description of Web service process models. Our working hypothesis is that the process model be described as an ontology of first-order logic. To this end, we present the Process Specification Language (PSL) [5, 4], a first-order logic ontology for modeling processes, as a straw proposal for a foundation from which a Web service process modeling framework can be developed.

The opinions expressed in this position paper reflect the position of the Semantic Web Services Language Committee (SWSL), a subcommittee of the joint North American-EU Semantic Web Services Intitiative (SWSI) (http://www.swsi.org). They are included in *FLOWS*, SWSL's First-order Logic Ontology for Web Services [1].

2 Current State of Process Modeling for Web Services

A Web service *process model* describes the program that implements a Web service. Over the past 4 years, industry has proposed a number of process modeling languages for describing the process models of Web services. Examples include Microsoft's XLANG, a Web service process modeling language based on pi-calculus; IBM's WSFL based on Petri Nets; BPEL4WS, a Microsoft, IBM, BEA, SAP and Siebel effort, which merges XLANG and WSFL; HP's Web Service Conversation Language (WSCL); BEA, Intalio, SAP and Sun's Web Service Choreography Interface (WSCI); BPML, backed by the Business Process management initiative; the XML Process Description Langauge (XPDL) backed by the Workflow Management Coalition; the Business Process Specification Schema (BPSS) of ebXML; and now under development, a W3C Choreography effort, which draws on pi-calculus. While not exhaustive, this list identifies some of the major efforts in Web

In evaluating and comparing existing Web service process modeling efforts, a key observation is that these efforts were designed to address a diversity of process management tasks. Some like BPEL4WS were designed to address Web service orchestration issues and to standardize workflow and execution with the objective of increasing transaction reliability and synchronization. Others have focused on issues of Web service choreography, requiring defining the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal[1]. As a consequence of the diversity of uses for which these languaes have been designed,

[*]Universita di Roma La Sapienza, Rome, Italy
[†]National Institute of Standards and Technology, Gaithersburg, MD, USA
[‡]Lucent Technologies, Murray Hill, NJ, USA
[§](Contact Author) University of Toronto, Toronto, ON, Canada, Email: sheila@cs.toronto.edu, Phone: 416-946-8484.

[1]http://www.w3.org/TR/ws-glossary/

comparing them based on concept coverage is important, but not necessarily pertinent, as many of these languages could be extended to incorporate further concept descriptions.

It is our view that the most important shortcoming of these languages, and the one that is least easily addressed, is their lack of well-defined semantics. Many of these languages have their origins in existing process specification languages, process algebras, and research in formal methods. Nevertheless, none has a well-defined semantics, except for XLANG and WSFL, which have effectively been abandonned in favour of BPEL4WS. Without a well-defined semantics, the process model cannot be manipulated, queried and interpretted reliably by a computer program. Adoption of a process model without a well-defined semantics severely restricts its practical use.

We take the point of view espoused by Semantic Web Services [8], that Web service descriptions should be unambiguously computer-interpretable, and that their concept coverage should be adequate to enable automation of Web service discovery, invocation, composition, monitoring, verification and simulation. We argue that this vision necessitates unambiguously computer-interpretable process models, which in turn necessitates having a well-defined semantics for these process models. We further argue that it requires certain content coverage, not addressed by current industry Web service process description languages.

In 2001, a coalition of semantic Web researchers, under the auspices of the DARPA DAML program, undertook to develop an ontology for Web services, using the Semantic Web ontology language DAML+OIL. This has culminated in the creation of OWL-S (formerly DAML-S) [2] a Web service ontology developed in OWL (the successor of DAML+OIL) [6]. OWL is an artificial intelligence description logic-based language for describing content on the Web. Most importantly OWL, and thus OWL-S, has a well-defined semantics, which contrasts it with other efforts. Unfortunately, OWL has not proven sufficiently expressive to characterize Web service process models. While OWL-S does indeed have a description of the process model of a Web service, OWL is not sufficiently expressive to denote all and only the *intended interpretations* of that process model. As such, like other process modeling languages, the OWL process model must be human interpretted to resolve ambiguities, or translated to another, richer language, in which this new model can be unambiguously interpretted by a program. Indeed there have been four efforts towards defining the intended interpretation of the OWL-S (or DAML-S) process model: a Petri Net-based operational semantics [9], an interleaving function-based operational semantics based on subtype polymorphism [], a semantics via translation to the first-order language of the situation calculus [], and most recently a semantics provided by translation to PSL, the National Institute of Standards' Process Specification Language.

OWL-S has many strong features. In particular, the concept coverage of OWL-S provides a firm foundation for our process modeling efforts. Further, OWL's expressiveness limitations, which OWL-S inherits, exist to address the important trade-off between expressiveness and decidability and tractability, and so, while limiting in this case, are certainly easily defensible.

Thus, we take OWL-S as our conceptual starting point and adopt much of its conceptual model which we do not describe here for lack of space. Nevertheless, we differ importantly in our choice of language. Rather than adopting OWL, we propose a first-order logic language for Web services descriptions. Before describing this language in further detail, we present a small set of motivating use cases.

3 Motivating Use Cases

In this section we present a small number of use cases that motivate and illustrate the need for computer-interpretable Web service process models in the context of Web service discovery and composition[2]. While many of the use cases are presented as client-side requests, these requests serve as constraints on existing process models and must be answered by manipulating server-side descriptions of the process models of Web services. As such, the challenge put forth in these use cases is to develop descriptions of the process models of relevant services, client-side requests, and (ideally) automated reasoning machinery to process the request.

Use Cases for Automating Web Service Discovery

1. Find me an airline service that enables me to reserve a flight before providing a credit card number.

[2]We chose these two tasks for diversity. Use cases requiring process modeling can likewise be developed for the other Semantic Web Service tasks.

2. Find me a book-buying service that returns a list of available second-hand copies of my requested book, if the book is out of print.

3. Find me a travel service that books hotels, flights and trains and that coordinates the timing. (I.e., that uses the output of one selection as input to the search for another.)

4. Find me a florist that enables me to pay with PayPal.

5. Find me an online financial advice service that queries www.morningstar.com prior to making mutual fund recommendations.

Use Cases for Automating Web Service Composition

1. Given a workshop registration service, a flight booking service, a car rental service, a taxi reservation service, and a hotel service, Ima's home address, and a Ima's online schedule, please book Ima Cheapskate's trip to the W3C Workshop on Web Services.

2. Modify problem 1, to add the constraints that Ima wants to travel on October 11, from Montreal, return on October 13 or 14, is unwilling to take an overnight flight, prefers to stay at the Sheraton in Palo Alto, but will stay at any hotel within 8 miles of the workshop and would like to rent a convertible car, if no rain is forecast. She does not want to register for the workshop until after her travel plans are made and does not want to rent a car, preferring a taxi, if her flight arrives in California between 3pm and 6pm.

4 Desiderata Derived from Use Case Analysis

Analysis of these and other use cases confirm the need for a computer-interpretable process model, and prompt a list of representational desiderata. We simply list these desiderata as follows: model-theoretic semantics, atomic and composite processes represented as first-class objects in the language, taxonomic representation, leverages OWL-S, embraces and integrates with exisitng and emerging industry Web service standards, provides for explicit representation of messages and dataflow, captures activities, process preconditions and program side effects, captures process execution history.

Our proposal is to develop a First-order Logic Ontology for Web Services (FLOWS). Our working hypothesis is to use PSL as the foundation for our ontology and to augment and customize it to Web service processes, specifically. Significant work has been done on the development of FLOWS over the last 8 monhts, primarily under the auspices of SWSL, the language committee of the joint North American-EU Semantic Web Services Initiative.

5 Towards FLOWS

5.1 The Case for First-Order Logic

OWL is too weak to completely axiomatize the intended semantics of OWL-S, and consequently, any implementations must resort to extralogical mechanisms if they are to conform to the OWL-S semantics. FLOWS (First Order Logic Ontology for Web Services) provides a first-order axiomatization of the intended semantics of OWL-S, and implementations of FLOWS will be able to use the axioms directly.

First-order logic provides a well-understood model-theoretic semantics. Its rich expressive power (e.g., variables, quantifiers, terms, etc.) overcomes the expressiveness issues that have haunted OWL-S.

First-order logic enables characterization of reasoning tasks for semantic web services in terms of classical notions of deduction and consistency. For example, web service discovery can be characterized as deductive queries, and web service composition, reachability, and liveness as satisfiability. This enables exploitation of off-the-shelf systems such as existing FOL reasoning engines and DB query engines, thereby facilitating implementation and improving our understanding of the reasoning tasks.

First-order logic has been criticized because it is semi-decidable (as opposed to OWL-DL, which is decidable). However, the motivating scenarios for semantic web services show that in general we will need to solve intractable reasoning problems. Intractable reasoning problems are inherently intractable – using a different language does not make them tractable. The restriction to a language that is tractable simply means that there will exist reasoning problems that cannot be specified in the language.

Furthermore, many intractable tasks often prove easily solved in practice. One powerful strategy is to explicitly axiomatize the extensions of a first-order theory that have attractive computational properties. The idea is to focus on the complexity of a particular first-order theory, and to introduce domain assumptions that can be used to guarantee that a particular reasoning problem using the theory is tractable.

5.2 The Role of PSL

We are exploring the use of PSL as the basis for this first-order axiomatization. The Process Specification Language (PSL) [5, 4] has been designed to facilitate correct and complete exchange of process information [3]. The PSL Ontology is a set of theories in the language of first-order logic. All core theories within the ontology are consistent extensions of a theory referred to a PSL-Core, which introduces the basic ontological commitment to a domain of activities, activity occurrences, timepoints, and objects that participate in activities. There is also a set of theories referred to as PSL Outer Core, which supports the explicit representation of state, concurrency, and complex actions. In particular, one of the theories in PSL Outer Core is equivalent to the first-order theory of Reiter's axiomatization of the situation calculus. Additional core theories capture the basic intuitions for the composition of activities, and the relationship between the occurrence of a complex activity and occurrences of its subactivities.

The FLOWS Process Model incorporates the following concepts and features from PSL:

- Process decomposition (e.g., subactivities)
- Ordering and temporal constraints
- Simple workflows
- Iterated processes
- Duration constraints
- Concurrency
- Explicit representation of state and state constraints
- World state conditions, inputs, and outputs, epistemic states of actors
- Preconditions and effects
- Conditional processes
- Occurrence constraints
- Composition
- Nondeterminism (e.g. alternative processes)
- Interactions with external activities
- Incomplete process specifications.

The model theory of PSL provides a rigorous abstract mathematical characterization of the semantics of the terminology of ontology. This characterisation defines the meanings of terms with respect to some set of intended interpretations represented as a class of mathematical structures. Furthermore, for each theory in the PSL-Ontology, there are theorems that prove that every intended interpretation is a model of the theory, and every model of the theory is isomorphic to some intended interpretation.

This first-order semantics has several advantages. Since complex activities and their occurrences are elements of the domain, PSL sentences can explicitly describe and quantify over complex activities. This makes it possible to

[3] As of June 2004, PSL has been accepted as an International Standard (ISO 18629) within the International Organisation of Standardisation.

express in an explicit manner a broad variety of properties and constraints on composite activities. Service design analysis and verification can therefore be expressed as inference problems that are solved by inference techniques that are sound and complete with respect to models of the theories. Also, a process ontology with a first-order axiomatization can be more easily integrated with other ontologies (which are almost all first-order theories themselves).

Other approaches to process representation lack one or more of the formal properties of the PSL Ontology – many ontologies do not provide a model theory; those that do specify a formal semantics, typically do not provide an axiomatization of this semantics.

PSL has already proven useful as exchange language between manaufacturing and business process software applications. Similar techniques can be used to integrate the diverse range of existing research efforts in semantic web services.

Gruninger has already demonstrated that PSL can provide a first-order characterization of the semantics for central concepts in the OWL-S ontology [3]. The close model-theoretic relationship between PSL and Reiter's axiomatization of the situation calculus supports the extension of implementations of Golog to perform web service composition as described in [7].

Preliminary work has been done on characterizing the equivalence between activity-based finite state machines (as illustrated in §C2) and restricted classes of PSL process descriptions. Given an activity-based fsm $M = (\Sigma, T, \delta, s, F)$, we have developed an algorithm for creating a PSL process description Φ_M corresponding to M. The set of fluents used in Φ_M includes symbols for each state of M, and we assume the existence of atomic activities for each of the activities of M, i.e., for each letter of Σ. Φ_M contains several sentences that describe properties that must hold in order for an activity tree A to correspond to accepting executions of M. For example, for each state $t \in T$ and activity $a_n \in \Sigma$, if the transition function $\delta(t_m, a_n)$ is defined, then the following sentence is in Φ_M.

6 Closing Remarks

In closing, we re-articulate our position that an unambiguous computer-interpretable process model of Web service is essential to automation of many Web service tasks. We have motivated this claim and have presented results towards FLOWS, a first-order logic ontology for Web services.

References

[1] D. Berardi, M. Gruninger, S. Hull, and S. McIlraith. Flows: A first-order logic ontology for web services, 2004. Manuscript in preparation.

[2] DAML-S Coalition: A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic markup for Web services. In *Proc. International Semantic Web Working Symposium (SWWS)*, pages 411–430, 2001. http://www.daml.org/services/.

[3] M. Gruninger. Applications of psl to semantic web services. In *Workshop on the Semantic Web and Databases*, Berlin, 2003.

[4] M. Gruninger. A guide to the ontology of the process specification language. In S. Staab R. Studer, editor, *Handbook of Ontologies*, pages 575–592. Springer-Verlag, 2003.

[5] M. Gruninger and C. Menzel. Process specification language: Theory and applications. *AI Magazine*, 24:63–74, 2003.

[6] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[7] S. McIlraith and T. Son. Adapting golog for composition of semantic web services. In *Proc. of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, pages 482–493, 2002.

[8] S. McIlraith, T. Son, and H. Zeng. Semantic Web services. *IEEE Intelligent Systems (Special Issue on the Semantic Web)*, 16(2):46–53, March/April 2001.

[9] S. Narayanan and McIlraith S. Analysis and simulation of web services. *Computer Networkds*, 42:675–693, 2003.