Web Service Discovery Based on Unified View on Functional and Non-Functional Properties

Martin Junghans
Karlsruhe Institute of Technology (KIT),
Institute of Applied Informatics and
Formal Description Methods (AIFB),
Karlsruhe Service Research Institute (KSRI)
Email: martin.junghans@kit.edu

Sudhir Agarwal
Karlsruhe Institute of Technology (KIT),
Institute of Applied Informatics and
Formal Description Methods (AIFB),
Karlsruhe Service Research Institute (KSRI)
Email: sudhir.agarwal@kit.edu

***Abstract*—Ever increasing acceptance of service oriented architectures in combination with the acceptance of the Web as a platform for carrying out electronic business triggers a need for automated methods to find appropriate Web services. Existing discovery approaches often support a very restricted set of use cases mainly due to the ignorance of non-functional properties of services. In our formal model of Web services, we have a unified view on properties that characterize functional as well as non-functional aspects. We present how desired combinations of properties are specified and interpreted as a set of desired service descriptions and show how service descriptions that fulfill such constraints are discovered.**

I. Introduction

Web service discovery deals with finding appropriate Web services for a task at hand and is one of the central components needed for developing service oriented applications. It is realized by matching available service descriptions against a service request and identifying those ones from the given descriptions that fulfill the request.

One common problem of almost all existing and well known discovery approaches is that they apply the same formalism for describing service offers and requests. Intuitively, a service description formalizes the actual values of service properties. In contrast, a service request specifies desired property value ranges. Therefore, using the same formalism with same interpretation for both service description as well as request does not correspond with the requester's intuition. Such mismatch between the semantics of formalisms and the intuitive interpretation of the requester makes these discovery approaches hard to use in practice.

We showed in [1] that the applicability of discovered services is not guaranteed for intersection based matchmaking between service offers and requests using the same formalism. E.g., if some requested service executions are provided by a service and thus an intersect match exists, it may happen that an invocation fails as not all requested executions are provided.

Unified View on Functional and Non-Functional Properties: Non-functional properties (NFPs) are part of semantic service descriptions and supplement functionality descriptions of services. In contrast to functional descriptions that describe the service behavior, NFPs describe manifold quality attributes of services. It can be observed that non-functional requirements (NFRs) are often referred to as soft criteria and exclusively considered for ranking [2]. It is not determined per se whether properties are interpreted as hard or soft requirements and it is likewise valid to perceive NFRs as hard requirements, too. E.g. the NFP $availability$ with a value of 0.99 can be considered for discovery as well. A request may specify that services have to offer an availability of at least 90%, which is considered to be a hard requirement. When a user specifies that she prefers higher availability rates to lower ones, then such requirements are referred to as a soft requirement and can be used to rank services.

In [1], we presented a solution to the first requirement by proposing different formalisms for service descriptions and requests. However, we did not incorporate NFPs for discovery yet. In this paper we present a discovery approach that treats functional and non-functional properties uniformly.

In Section II we present our extended formal model of Web services and a syntax for describing Web services together with its semantic as a mapping to the formal model. We introduce requests in Section III. Based on the semantics of the service description and request formalisms, matchmaking is presented in Section IV. We further show performance test results of our implementation in Section V. In Section VI, we discuss the relation of our approach to others and conclude in Section VII.

II. Service Descriptions

This section introduces the formalism to describe Web services semantically. We introduce a formal Web service model that captures functional and non-functional properties in a unifying way. Subsequently, a formalism for describing such models by presenting an abstract syntax and its semantics as a mapping to the formal model is provided.

A. Formal Model of Web Services

We consider a finite set $\mathcal{P}$ of Web service property types and a finite set $\mathcal{V}$ of value sets. Each property type $P \in \mathcal{P}$ is associated with a value set $V_P \in \mathcal{V}$. We view a Web service as a finite set of property instances Q with each property instance $q \in Q$ being of a property type $t(q) \in \mathcal{P}$ that is associated with a value $v_q \in V_{t(q)}$.

The functionality of a service is also a property within the model. The formal model of this property is a labeled transition system which comprises a set of states, a set of transition labels and a labeled transition relation modeling the transition between two states. An execution of a Web service can be seen as a series of states. Here, we consider traditional Web services that have no user interactions during their execution. That is, we assume that inputs are provided with the invocation and outputs are returned at the end. In [1] we provide more details on modeling the service functionality within this formal model.

B. Description Formalism

The presented formal model cannot be described completely mainly because of the following reasons. (1) Service providers may not want to reveal the exact sequence of operations they perform. (2) Assuming a global set of property names is not feasible. The first issue is relevant for functionality and is the reason why we only model start and end states by pre- and postconditions in addition to the set of inputs and outputs in service descriptions. We address the second issue by modeling Web service properties as properties in a known ontology language, e.g. OWL [3] or WSML [4]. This allows us to use existing ontology reasoners to reason about the properties while not forcing a global set of property names. More precisely, we

- define for each value set $V \in \mathcal{V}$ an ontology concept V. We assume a set of common data types either available directly or modeled as ontology concepts as well.
- model for each property $P \in \mathcal{P}$ with range V_P a property P as an object property with range $\mathsf{V_P}$ if V_P is a set of individuals. Otherwise, if V_P is a data type, we model a property P as a data type property with range $\mathsf{V_P}$.

Since ontology languages allow alignment of concepts and properties in subclass-of and subproperty-of relationships resp., we achieve interoperability among properties. Note, that functionality is just another property functionality with range concept Functionality in our model of Web services.

C. Modeling Example

A service s requires three inputs: user ID, password and a book's ISBN number which are specified as $I = \{\mathsf{id}, \mathsf{pwd}, \mathsf{isbn}\}$. The service creates a shipping order for the given book (if available) to the user's address and returns an invoice to the user about the details of the order. The precondition ϕ states that it requires the user u with ID id to be registered and authenticated by its password pwd. For a successful execution it requires that the book b with ISBN isbn is in stock.

$$\begin{aligned}\phi \equiv\ & \mathsf{isRegistered}(\mathsf{id}) \wedge \mathsf{isAuthenticated}(\mathsf{id}, \mathsf{pwd}) \\ & \wedge \mathsf{User}(\mathsf{u}) \wedge \mathsf{hasID}(\mathsf{u}, \mathsf{id}) \wedge \mathsf{Book}(\mathsf{b}) \wedge \mathsf{ISBN}(\mathsf{isbn}) \\ & \wedge \mathsf{hasISBN}(\mathsf{b}, \mathsf{isbn}) \wedge \mathsf{isAvailable}(\mathsf{b}) \ldots\end{aligned}$$

The post-condition ψ states that there exists an order o about product b (which is made sure by the precondition to be the ordered book) after service execution. The order is supposed to be shipped to the user's address ad and there exists an invoice about the book's order and price.

$$\begin{aligned}\psi \equiv\ & \mathsf{Order}(\mathsf{o}) \wedge \mathsf{containsProduct}(\mathsf{o}, \mathsf{b}) \wedge \mathsf{hasPrice}(\mathsf{b}, \mathsf{p}) \\ & \wedge \mathsf{isShipped}(\mathsf{o}, \mathsf{ad}) \wedge \mathsf{hasAddress}(\mathsf{u}, \mathsf{ad}) \wedge \mathsf{Invoice}(\mathsf{i}) \\ & \wedge \mathsf{containsOrder}(\mathsf{i}, \mathsf{o}) \wedge \mathsf{containsPrice}(\mathsf{i}, \mathsf{p}) \ldots\end{aligned}$$

The service returns the book b and the corresponding invoice i to the user, which is specified by $O = \{\mathsf{b}, \mathsf{i}\}$.

$$\begin{aligned}N = \{&\mathsf{acceptsCreditCard}(\mathsf{s}, \texttt{false}), \\ &\mathsf{deliveryInDays}(\mathsf{s}, 2), \mathsf{availability}(\mathsf{s}, 0.90)\}\end{aligned}$$

Apart from the described functionality, the NFPs N of the service s state that it does not accept credit cards, delivers within 2 days and has availability 90%.

III. Service Requests

We use a request formalism that differs from the formalism of offers. In this section we describe the request description syntax and its mapping into the formal request model.

A. Request Description Syntax

Requests constrain functional and non-functional properties of services in a unifying way. Besides requesting for inclusions and exclusions of desired property values, requests also allow for the specification of certain combinations of desired property values. For example, a user might accept a longer delivery time only if the service offers credit card payment.

A request always refers to a desired service s of type Service in order to refer to its properties and specify constraints on their values by expressions of the form $\mathsf{hasProperty}(\mathsf{s}, \mathsf{propertyValueSet})$. Thus, the structure of a request reflects the service model from Section II-A. The below example sketches a request $\mathcal{R}$ for a book selling service.

$$\begin{aligned}\mathcal{R} \equiv\ & \mathsf{Service}(\mathsf{s}) \wedge \mathsf{hasFunctionality}(\mathsf{s}, \mathsf{f}) \wedge \mathsf{hasPrice}(\mathsf{s}, \mathsf{p}) \wedge \\ & \mathsf{hasInputs}(\mathsf{f}, \text{"}\mathsf{i} \wedge \mathsf{id} \wedge \nexists \mathsf{d}\text{"}) \wedge \mathsf{hasOutputs}(\mathsf{f}, \text{"}\mathsf{book} \wedge \mathsf{inv}\text{"}) \wedge \\ & \mathsf{hasPrecond}(\mathsf{f}, \text{"} \ldots \mathsf{ISBN}(\mathsf{i}) \wedge \mathsf{Bday}(\mathsf{d})\text{"}) \wedge \\ & \mathsf{hasPostcond}(\mathsf{f}, \text{"}\mathsf{Book}(\mathsf{book}) \wedge \mathsf{hasISBN}(\mathsf{book}, \mathsf{i}) \wedge \\ & \mathsf{Invoice}(\mathsf{inv})\text{"}) \wedge \mathsf{hasDeliveryTime}(\mathsf{s}, \mathsf{dt}) \wedge \mathsf{lessThan}(\mathsf{dt}, 7)\end{aligned}$$

Constraints on functional properties are expressed by logical expressions [1]. A set of desired inputs, outputs, pre- and postconditions of desired services are characterized within requests. This example states that the desired sets of inputs must contain an ISBN and must not require any date of birth information. Furthermore, the desired service must return a book that is identified by the given ISBN as well as an invoice.

Constraints on NFPs: Non-functional requirements, denoted by $\mathcal{N}$, constrain values of NFPs such that the request describes a set of desired values. Analogously to service descriptions, each of the desired services in a request is described by a finite set of property instances Q. A property instance $q \in Q$ of a property type $t(q) \in \mathcal{P}$ is restricted to a set of desired values $V_{R,q} \subseteq V_{t(q)}$. As an example, the maximum

delivery time of 7 days of the desired service s is expressed by $\mathsf{hasDeliveryTime}(\mathsf{s},\mathsf{dt}) \wedge \mathsf{lessThan}(\mathsf{dt},7)$.

B. Semantics of Service Requests

The semantics $\mathcal{I}$ of a request maps a set of property-values sets into the formal model that is described by sets of desired property-value sets. Translated into the formal model, a request is a set of sets of property instances $q \in Q$ of type $t(q)$, which is assigned to a value $v \in V_{R,q}$ that is member of the desired value set $V_{R,q}$.

Let $q = (p, V_R)$ denote a requested property instance of user's concern. $V_{R,q} \subseteq V_{t(q)}$ is the set of desired property values of the property $p = t(q) \in \mathcal{P}$. Then, the interpretation

$$\mathcal{I} : \{\mathcal{Q} : \mathcal{Q} \subseteq \bigcup_{q \in Q} \{q\} \times 2^{V_{R,q}}\} \rightarrow \{\mathcal{Q} : \mathcal{Q} \subseteq \bigcup_{q \in Q} \{q\} \times V_{R,q}\}$$

of a property request is $(p, V_{R,p})^{\mathcal{I}} = \{(p, v) | v \in V_{R,p}\}$, which is the set of property-value pairs that is constructed by considering each value $v \in V_{R,p}$ that is member of the set of acceptable values individually.

This interpretation provides us means to formalize desired properties of services. Again, we refer to the work in [1] that introduces modeling of the desired value set of the functionality property as a set of labeled transition systems.

IV. Matchmaking

After we introduced two formalisms and their translations into a common model, we define a match and how it is determined. A match is given if a service w meets all requirements of a request $\mathcal{R}$, i.e. property values of the offer are in the sets of desired values of corresponding property instances.

Within the two formal models of service offers and requests, a match is computed by checking whether the offer is contained in the set of desired service descriptions of the request. As each desired property is modeled as a set of desired values, the matchmaker checks for a containment relation between service offer and request. This applies to all property-value pairs including the labeled transition system based formal interpretation of offered and requested functionalities.

A service description was interpreted as a set

$$Q_w^{\mathcal{I}} \subseteq \bigcup_{P \in \mathcal{P}} P \times V_P \tag{1}$$

of property instances comprising functional and non-functional properties in a unifying way. Within the formal model of service descriptions, the property instances $q \in Q$ model the assignment of a property $t(q)$ to a value $v_{t(q)} \in V_{t(q)}$.

Constraints on properties Q_R in a request are formalized as a set of values assigned to a property.

$$Q_R \subseteq \bigcup_{P \in \mathcal{P}} P \times 2^{V_P} \tag{2}$$

$$Q_R^{\mathcal{I}} \subseteq \left\{Q : Q \subseteq \bigcup_{P \in \mathcal{P}} P \times V_P\right\} \tag{3}$$

As can be easily derived from Equations (1) to (3), a set of property instances $Q_w^{\mathcal{I}}$ of a Web service description matches against the requested properties $Q_R^{\mathcal{I}}$ if and only if $Q_w^{\mathcal{I}} \in Q_R^{\mathcal{I}}$. I.e., there exists a set $Q'_R \in Q_R^{\mathcal{I}}$ of property instances that equals the set $Q_w^{\mathcal{I}}$. Then there exists $q_R \in Q'_R$ for each property instance $q_w \in Q_W^{\mathcal{I}}$ with $q_R = q_w$ and this in turn means that resp. types $t(q_R) = t(q_w)$ and values $v_{q_R} = v_{q_w}$ of both property instances are equal per definitionem.

Matchmaking, as introduced above, detects the match between the example service description from Section II and the request from Section III. For instance, the NFPs N match the NFRs $\mathcal{N}$, because the offered service is that fast that it delivers in less than three days although it does not accept a credit card. Furthermore, the pre-condition also matches the request as the it only requires that the service identifies books by ISBN and the post-condition as it delivers the proper book with an invoice and the user receives reward points. Another book selling service that identifies books by author name and book title does not match this example request $\mathcal{R}$.

V. Implementation and Evaluation

We fed service descriptions in form of ontologies into a WSML reasoner[1] and used the WSML core language dialect for Web service descriptions within the EU-funded project SOA4All. Notice that our approach is not bound to these technologies. It can be also used in conjunction with standardized languages like OWL and a corresponding reasoner.

A user interface allows to formulate and submit requests that may contain a combination of constraints on desired properties. The discovery engine translates the user request into proper WSML syntax and sends this WSML query to the reasoner. The reasoner executes the query upon its knowledge base, which models all service descriptions. For each service modeled in the knowledge base, the reasoner determines whether the descriptions of offered inputs I, outputs O, pre-condition ϕ, post-condition ψ, and NFPs N are a model of the requested combination of inputs $\mathbb{I}$, outputs $\mathbb{O}$, pre-condition Φ, post-condition Ψ, and NFRs $\mathcal{N}$, resp. Therefore, the reasoner checks all mappings between variables of query and service description. A service is in the result set if there is a mapping that fulfills pre- and post-condition plus the requested sets of inputs and outputs cohere to this mapping

Performance Results: We created a repository of randomly generated service descriptions. We use the Semantic Web for Research Community ontology as domain knowledge to model service descriptions. We measured the reasoner's mean query answering time on a quad core Xeon CPU (2.33GHz) powered machine. Small, medium and large conjunctive queries with 6, 9, 12 variables and 9, 12, 15 properties resp. within the desired pre- and post-conditions were sent to the reasoner. Queries further comprised 2, 4, 6 NFRs, resp. The mean query answering time ranges from 2.8s, 4.2s, 5.0s with 5K service descriptions to 17s, 23s, 33s with 30K descriptions for small, medium, large sized queries, resp.

We omit a comparison with existing discovery approaches as our approach features different expressivity. We intend to

[1] See http://tools.sti-innsbruck.at/wsml2reasoner for reasoning details.

show the feasibility of the presented approach only. Obviously, query answering time highly depends on size and structure of the used domain ontologies, size and complexity of the query and service descriptions.

VI. Related Work

Many semantic Web service discovery approaches have been proposed. OWL-S Profile [5] models Web services semantically with inputs, outputs, pre-conditions and effects (IOPE). OWL-S matchmaker uses OWL-S profile for describing offers as well as requests but considers types of inputs and outputs only. The semantic Web community with focus on languages provides description logic (DL) based service description approaches [6], [5], [7]. Li et al. represents objects like inputs and outputs as concepts in description logics in [8] and further combine the use of DL with DAML+OIL and DAML-S. Service descriptions and requests are similarly structured comprising IOPE. Intersection based matchmaking is reduced to subsumption checking of input and output types. However, DL-based approaches fail to reason about the dynamics of services since DL reasoners can not reason about changing knowledge bases. Thus, formalisms like state based approaches that cover service dynamics were developed recently.

Stollberg et al. uses a state based formal model of service descriptions [9]. The service functionality is formally described by possible Web service executions while each normal execution is determined by its start and end states. The discovery algorithm [10] relies on the assumption $\phi \Rightarrow \psi$ that the pre-condition logically implies the effect. However, modeling a transition as a logical implication can be problematic, e.g., a service that deletes a certain fact, when the existence of a fact would imply non-existence of this fact.

The mentioned approaches did not use NFPs for discovery. In OWL-S, NFPs are considered as human-readable meta-data, e.g. service name. WSML [4] does not include NFPs into the logical model. Consequently, no reasoning on them is possible. The WSMO specification defined NFPs, however there is so far no prominent implementation available that considers them, such as the Internet Reasoning Service [11]. O'Sullivan [12] described a set of NFPs relevant for modeling Web services which were formalized in a WSMO deliverable.

Goal-driven approaches like [10], [13], [14] do neither consider NFPs nor do they explicitly specify inputs in goals. The usability of one global hierarchy of goal templates is hardly feasible in an decentralized and open setting like the Web. Furthermore, [10], [15] do not deal with possible inconsistencies between functional description of Web services with their classification. In a goal, constraints on inputs can be useful, in particular if a user wishes to exclude a particular input parameter. Another difference to our approach [1] is the different interpretation of offers and requests.

VII. Conclusion and Outlook

We presented a formal service model based on a unified view on functional and non-functional properties and enhanced the formalisms introduced in [1] in order to capture NFPs. We have shown how expressive requests with combinations of constraints on NFPs and functional properties are formulated. Based on the semantics and the common formal model of service and request descriptions, we defined a match between both and also showed how it can be computed. Finally, we completed the presented work by a implementation that is part of the larger system developed in the EU project SOA4All.

Our long term goal is to establish a scalable semantic service search framework, which tightly integrates discovery and ranking into the search of services. Both functional and non-functional properties are not distinguished and likewise considered for discovery and ranking. We aim to achieve efficiency and scalability by developing indexing structures and reducing the search space by early ranking. Known service matching techniques like subsume and plugin match may be exploited to develop indexing among service descriptions as these techniques use the same formalism.

Acknowledgments: We acknowledge funding by the European project SOA4All (FP7-215219, http://www.soa4all.eu).

References

[1] M. Junghans, S. Agarwal, and R. Studer, "Towards Practical Semantic Web Service Discovery," in *7th Extended Semantic Web Conference*, ser. LNCS. Springer, 2010.

[2] S. Agarwal, S. Lamparter, and R. Studer, "Making Web services tradable - A policy-based approach for specifying preferences on Web service properties," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 1, pp. 11–20, Januar 2009.

[3] W3C OWL Working Group, *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009, available at http://www.w3.org/TR/owl2-overview/.

[4] J. de Bruijn, D. Fensel, M. Kerrigan, U. Keller, H. Lausen, and J. Scicluna, *Modeling Semantic Web Services: The Web Service Modeling Language*. Berlin: Springer, 2008.

[5] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated Discovery, Interaction and Composition of Semantic Web Services," in *Journal of Web Semantics*, vol. 1, no. 1, Dec. 2003, pp. 27–46.

[6] B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani, "On automating Web services discovery," *The VLDB Journal*, vol. 14, no. 1, pp. 84–96, 2005.

[7] J. Gonzalez-castillo, D. Trastour, and C. Bartolini, "Description Logics for Matchmaking of Services," in *KI-2001 Workshop on Applications of Description Logics*, 2001.

[8] L. Li and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology," *Int. J. Electron. Commerce*, vol. 8, no. 4, pp. 39–60, 2004.

[9] U. Keller, H. Lausen, and M. Stollberg, "On the Semantics of Functional Descriptions of Web Services," in *Proc. of the 3rd European Semantic Web Conf.*, 2006.

[10] M. Stollberg, M. Hepp, and J. Hoffmann, "A Caching Mechanism for Semantic Web Service Discovery," in *The Semantic Web. 6th Int. Semantic Web Conf.*, ser. LNCS 4825, K. Aberer and et al., Eds. Busan, Korea: Springer, 2007, pp. 480–493.

[11] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci, "IRS-III: A broker-based approach to semantic Web services," *Web Semant.*, vol. 6, no. 2, pp. 109–132, 2008.

[12] J. O'Sullivan, "Towards a precise understanding of service properties," Ph.D. dissertation, Queensland University of Technology, 2006.

[13] R. Lara, M. Corella, and P. Castells, "A Flexible Model for Locating Services on the Web," *Int. J. Electron. Commerce*, vol. 12, no. 2, 2008.

[14] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel, "Automatic Location of Services," in *Proceedings of the 2nd European Semantic Web Symposium (ESWS2005)*, Heraklion, Crete, 5 2005.

[15] T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel, "WSMO-Lite Annotations for Web Services," in *5th European Semantic Web Conf.*, ser. LNCS 5021. Springer, 2008.