A Goal Specification Language for Automated Discovery and Composition of Web Services

Sudhir Agarwal
Institute of Applied Informatics and Formal Description Methods (AIFB),
University of Karlsruhe (TH), Germany.
agarwal@aifb.uni-karlsruhe.de

Abstract

In order to find suitable Web services from a large collection of Web services, automatic support is needed to filter out Web services relevant according to some criteria specified by the user. In real business scenarios constraints on the types of input and output parameters are often not sufficient. Rather one wishes to specify constraints on relationships of input and output parameters, interaction pattern and non-functional properties of Web services. Therefore, there is a need for a more expressive goal specification language.

Current goal specification techniques for matchmaking and composition of Web services either lack expressivity to support real business scenarios or formal semantics to enable development of automatic algorithms. In this paper, we present a goal specification language that allow specifying constraints on functional and non-functional properties of Web services. The language is a novel combination of an expressive temporal logic μ-calculus and an expressive description logic $\mathcal{SHIQ}(\mathbf{D})$.

1. Introduction

Service orientation is a promising paradigm for offering and consuming business processes within or across organizational boundaries. Ever increasing acceptance of service oriented architecture (SOA) and the need for outsourcing business processes partially or completely will lead to large number of business processes offered as Web services. Therefore, there is a need for mechanisms to find suitable Web services automatically. Since the suitability of Web services is relative to user's criteria, one first needs a technique to specify user's criteria.

In the recent years, quite a few formalisms have been proposed for describing Web services formally. However, there has not been much work done for specifying user criteria in order to find suitable Web services. In real business scenarios, the criteria for the suitability of a Web service are much richer than just the types of its input and output parameters. Existing approaches either lack the expressivity so that finding suitable Web services still requires a lot of manual effort or formal semantics so that it is difficult to develop automatic discovery and composition algorithms.

In this paper, we present a language for specifying constraints on functional and non-functional properties of Web services. In Section 2, we identify requirements for an expressive goal specification language. In order to be able to do reasoning over functional and non-functional properties of Web services, such properties must be described first. In Section 3, we present a formal model of a Web service. In section 4, we present a formalism for specifying expressive user's goals. In section 5, we discuss some related work. Finally, we summarize our work in section 6 and discuss some open problems.

2. Language Requirements

A Web service operates on resources that can be real world objects or information objects. A Web service expects resources from the client and delivers resources to the client. A requester searching for a Web service typically wishes to define constraints on the resources expected from and delivered by the Web service.

Consider for example a library that maintains a wish list of books where the library members can enter the ISBN of books they would like to have. On the basis of some ranking (e.g. number of members that wish the book) of the desired books the library system needs to place book orders at regular intervals. At the time of designing and implementing the library system the software engineers need to find book selling Web ser-

vices. Suitable book selling services are those that accept an ISBN and deliver a book.

Requirement 1 *It must be possible to define constraints on the types of input and output parameters of a Web service.*

Considering the library scenario a bit deeper, we identify that the software engineers are actually interested in Web services that do not sell just any book, but exactly the book that is ordered by the library software. In this example a suitable Web service is one that delivers a book with the same ISBN as the one it obtains as input.

Requirement 2 *It must be possible to define constraints on the values of the properties of the resources. In other words, it must be possible to define relationships among resources.*

In realistic scenarios we must consider that there are many book selling Web services available and there is no global vocabulary for the domain of books that every Web service provider can or wants to use for describing the resources of his book selling Web service. As a result, we have to assume that in general Web service providers describe their Web services with their respective vocabularies independently of other Web service providers. Similarly, a requester needs a vocabulary that he understands to be sure that the constraints that he specifies capture the intended meaning.

Consider for example, that the output type of a book selling service is `Book`[1] and a requester searches for Web services that have output type `T2Z7Q`. If the type `Book` as well `T2Z7Q` mean the set of books, then the Web service must be detected as a match.

Requirement 3 *The formalism for specifying constraints on resources must be able to consider mappings among the vocabularies while checking the satisfiability of constraints.*

When a Web service executes, it may cause changes in the knowledge state of the participants. For example, charging requester's credit card as a consequence of the order he has placed is performed by means of an update (setting the available credit amount to a lower value) in the database of the corresponding bank. While searching for desired Web services, a requester may be interested in specifying which effects he wishes to take place and which not.

1 Classically defined as complex type in an XML schema; in case of semantic Web services as a concept of an ontology.

Requirement 4 *The formalism for specifying constraints must support specification of desired and undesired changes in the resources.*

Because of the vast heterogeneity of the available information, Web service providers and users, security becomes extremely important. Security related aspects are mostly classified in three categories, namely confidentiality, integrity and availability [9, 15, 10].

When a Web service uses other Web services in order to achieve its functionality, it may be possible that a requester trusts the composite Web service but not its component Web services.

Requirement 5 *The formalism should allow a requester to specify constraints on the set of parties involved in the execution of a Web service.*

A suitable Web service for the library system must have the matching (opposite pole) communication pattern so that the interaction between the Web service and the client can take place. If the Web services had only one input activity and only one output activity, one may assume that a Web service always perform the output activity after the input activity. As a consequence matching the communication patterns of the client and a Web service would be trivial. However, in general this is not the case. Even if Web services abstract from implementation details, the process triggered by invoking a Web service may be very complex involving multiple interactions with the client or other Web services. Consider a book selling Web service that after receiving a book order, sends a confirmation to the client and expects a back confirmation. Only after receiving the back confirmation from the client, it sends the ordered book to the client. Even if the Web service receives an ISBN in the first input activity and output the book with the same ISBN in the last activity, it may not be suitable for the library system, if the library system does not foresee to receive an order confirmation and sending a back confirmation. In other cases, an opposite situation may occur. That is, a client system is ready to receive a book only after it has received an order confirmation and sent a back confirmation, but the Web service neither sends any order confirmation nor it waits for a back confirmation before sending the book to the client.

Requirement 6 *In order to find Web services that can be incorporated in the client system it must be possible for the requester to specify constraints on the communication pattern of a Web service.*

While the above requirement concerns the public communication protocol of the Web services (choreog-

raphy), there are use cases, in which a requester may be interested in specifying constraints on the internal communication structure of a Web service (orchestration). Consider, for example a company internal Web service exposing a complex order process. It may be desired that the ordered good must be approved by at least two managers before it is bought. In other words, this means that in order to check whether the Web service is compliant to company's policies, one must be able to reason about the information flow and the order of activities that transport information.

Requirement 7 *In order to find Web services that can be incorporated in the client system it must be possible for the requester to specify constraints on the internal communication of a Web service with other services.*

Finally, a requester may wish to combine different types of constraints. In the following, we give some examples of the combination of different types of constraints.

- **Combination of Effects and Behavior** A user may wish to search for services that charge the credit card after the shipment of the ordered book.
- **Combination of Resources, Behavior and Trust** A Web service may send some user specific information, e.g. his credit card number or his date of birth, to its component Web services that a requester (potential user) may not trust. In order to make sure that a Web service acts in compliance with his privacy policies, he wishes to specify constraints like "user's date of birth should not be sent to a party that the user does not trust"
- **Combination of Trust and Effects** The bank Web service should check that the book selling Web service is authorized (by the user) to cause an effect in the user's bank account.

Requirement 8 *The formalism should allow to specify different types of constraints in a unifying and composable manner. That is, it must be possible to specify complex constraints from simpler constraints that may be of different types.*

3. Formal Model of Web Services

In this section, we fix the model of Web services that we consider for developing the goal specification language. Providing a concrete syntax for describing such a formal model is out of the scope of this paper. In our earlier works, we have presented a π-calculus and $\mathcal{SHIQ}(\mathbf{D})$ based syntax for describing such models [5, 1].

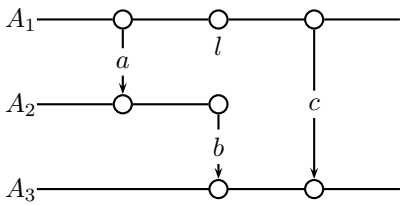

Figure 1. Formal model of Web services illustrated with an example of three agents.

The business processes underlying a Web service may consist of many activities. In some cases, the order of execution of the activities is important. Hence, the formalism should be able to specify the temporal behavior of business processes. Due to Web service technology, more and more business processes run partially or completely inside the Web. In general, the process triggered by invoking a Web service may involve many actors, that may or may not be Web services. In order to specify such business processes, we need a formalism that allows modeling of processes running in a mixed environment in a unifying way.

The Web service providers will obviously describe the non-functional properties of their Web service in a way that their Web services are found often. For example, almost every Web service provider will tend to say that his Web service has fast response time. So, non-functional properties of a Web service described by the provider of the Web service do not have much significance. Rather, the non-functional properties of Web services must be describable by other parties without a central control and in a way that users can build their trust in them.

We denote the set of agents with $\mathcal{A}$. Each agent $A \in \mathcal{A}$ has functional properties and non-functional properties. Functional properties consist of resources, resource schemata and behavior of an agent whereas non-functional properties describe credentials of an agent.

3.1. Functional Properties

3.1.1. Resources and Resource Schemata Each agent $A \in \mathcal{A}$ is associated with a resource schema S_A. Furthermore, each agent $A \in \mathcal{A}$ has a finite set of resources R_A. These resources can be real world objects or information objects.

3.1.2. Behavior The agents run concurrently to each other and communicate by exchanging messages (refer to Figure 1). In case of Web services, the messages are exchanged via Web protocols like

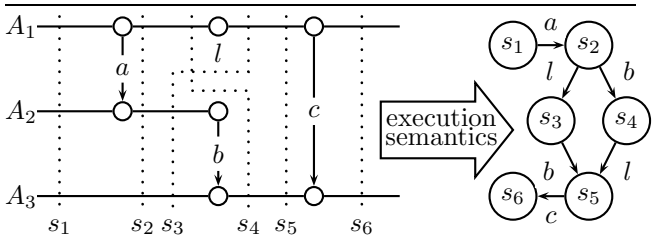

Figure 2. From execution model to labeled transition system with execution semantics.

HTTP and are serialized in some Web standard syntax like SOAP. Furthermore, the actors may perform operations locally, e.g., adding and removing resources to and from their respective set of resources respectively or defining relationships among the resources in their respective set of resources.

In general, more than one resource can be transported or communicated from one agent to another in a communication activity. Protocol types are important in practice, since many business processes run in a mixed environment.

Figure 2 shows on the left hand side various situations or states that may occur during the execution of the example Web service depicted in Figure 1. The right hand side of the Figure 2 shows the labeled transition system of the example Web service. For $\mathcal{A} = \{A_1, \ldots, A_n\}$, the sets of resources $R_{A_1}, \ldots, R_{A_n}$ of the agents $A_1, \ldots, A_n$, together with their behaviors $B_{A_1}, \ldots, B_{A_n}$ at some given point of time, describe the state of the system at that time.

3.2. Non-Functional Properties

Apart from the resources, we consider the properties of an agent in our formal model. For example, *agent i is a university* or *agent j is above 18 years.*

We denote with $\mathcal{P}$ the set of all non-functional properties. The basic idea behind modeling non-functional properties of actors in an interoperable way is to interpret a non-functional property as a set of agents and assume a subset relationships among such sets. For example, the statement $\mathsf{above21} \subseteq \mathsf{adult}$ says that any agent who above 21 years of age is also adult. Hence, agents possessing the non-functional property above21 can be automatically selected when a user wishes to select agent that possess the property adult. As we have mentioned before, it is necessary to consider who issues whom a non-functional property, so that users can build their trust or distrust in them. However, trustworthiness of the non-functional properties is not the focus of this paper and we refer to our previous works [3, 4], in which we have proposed to use SPKI/SDSI certificates for issuing non-functional properties to agents.

4. Specification of Expressive Goals

In this section, we present our main contribution, a formalism to specify constraints on Web services. The novelty of the formalism lies in its holistic nature as it allows specification of constraints on resources, behavior and non-functional properties of Web services in a unifying way. In the previous section, we have introduced our formal model of Web services and shown how the executable process model can be translated to a labeled transition system. The main aim of a goal specification language is to provide a syntax and semantics to restrict the set of states in such an LTS. That is, by specifying constraints in a goal specification language, a user should be able to fix the set of states that may/should occur in the execution of a Web service or a Web service composition.

As we have stated earlier, a Web service is in general a process involving multiple actors. μ-calculus is one of the most expressive temporal logics while still being decidable[13]. In this section, we will first show how constraints on the behavior of a Web service can be specified with μ-calculus. Then we will add a facility to specify constraints on the involved resources by giving structure to μ-calculus propositions and actions. Finally, we consider specification of constraints on the non-functional properties and show how they can added to our goal specification language, thus providing a single very expressive goal specification language.

4.1. Specifying Constraints on Behavior with μ-calculus

μ-calculus, as it is mostly used today was first introduced in [13]. Let Var be an (infinite) set of variables names, typically indicated by $X, Y, Z \ldots$; let $Prop$ be a set of atomic propositions, typically indicated by $P, Q, \ldots$; and let A be a set of actions typically indicated by $a, b, \ldots$. The syntax of formulae (with respect to $(Var, Prop, A)$ is presented in column two of Table 1.

Definition 1 (Structure) *A structure $\mathcal{T}$ (over $Prop$, A) is a labeled transition systems, namely a set $\mathcal{S}$ of states and a transition relation $\rightarrow \subseteq \mathcal{S} \times A \times \mathcal{S}$, together with an interpretation $\mathcal{V}_{Prop} : Prop \rightarrow \mathcal{P}(\mathcal{S})$ for the atomic propositions. We often write $s \xrightarrow{a} t$ for $(s, a, t) \in \rightarrow$.*

Given a structure $\mathcal{T}$ and an interpretation $\mathcal{V} : Var \rightarrow \mathcal{P}(\mathcal{S})$ of the variables, the set $[\![\phi]\!]_{\mathcal{V}}$ of states

Name	Syntax	Semantics
True	`true`	$\mathcal{S}$
False	`false`	$\emptyset$
Atomic Proposition	P	$\mathcal{V}_{Prop}(P)$
Conjunction	$\phi_1 \wedge \phi_2$	$[\![\phi_1]\!]_{\mathcal{V}} \cap [\![\phi_2]\!]_{\mathcal{V}}$
Universal Quantifier	$[a]\phi$	$\{s \mid \forall t\ s \xrightarrow{a} t \Rightarrow t \in [\![\phi]\!]_{\mathcal{V}}\}$
Negation	$\neg\phi$	$\mathcal{S} \setminus [\![\phi]\!]_{\mathcal{V}}$
Minimal Fixpoint	$\mu Z.\phi$	$\bigcap\{S \subseteq \mathcal{S} \mid S \subseteq [\![\phi]\!]_{\mathcal{V}[Z:=S]}\}$
Disjunction	$\phi_1 \vee \phi_2$	$[\![\phi_1]\!]_{\mathcal{V}} \cup [\![\phi_2]\!]_{\mathcal{V}}$
Existential Quantifier	$\langle a\rangle\phi$	$\{s \mid \exists t\ s \xrightarrow{a} t \wedge t \in [\![\phi]\!]_{\mathcal{V}}\}$
Maximal Fixpoint	$\nu Z.\phi$	$\bigcup\{S \subseteq \mathcal{S} \mid S \subseteq [\![\phi]\!]_{\mathcal{V}[Z:=S]}\}$

Table 1. μ-calculus Syntax and Semantics

satisfying a formula ϕ is defined in column three of Table 1.

With fixed point operators, one can define all known constraints on the temporal behavior. In the following, we give examples of some most widely known temporal operators. The formula $\mu X.(\phi_2 \vee (\phi_1 \wedge \langle - \rangle \texttt{true} \wedge [-]X))$ describes $\phi_1\, \texttt{until}\, \phi_2$, as it can be read as: either ϕ_2 holds in the current state or sooner or later the process reaches a state in which ϕ_2 holds and until then ϕ_1 holds. The modality $\texttt{eventually}\, \phi$ can be easily defined as $\texttt{true}\, \texttt{until}\, \phi$.

4.2. Adding Structure to μ-calculus Propositions and Actions with $\mathcal{SHIQ}(\mathbf{D})$

μ-calculus in its pure form abstracts from the meaning and structure of the propositions. In our formal model, the set of propositions of an agent correspond to the facts in the knowledge base of the agent. The facts (explicit or derived) in the knowledge base of an agent at some point of time represent the set of propositions that are true at that point of time.

4.2.1. Short Introduction to $\mathcal{SHIQ}(\mathbf{D})$ A $\mathcal{SHIQ}(\mathbf{D})$ description logic knowledge base consists of a set of axioms, which can be distinguished into terminological axioms (building the so-called TBox $\mathcal{T}$) and assertional axioms or assertions (constituting the ABox $\mathcal{A}$). Based on names for concepts (as C, D,...), roles (R, S,...), and individuals (a, b,...), $\mathcal{SHIQ}(\mathbf{D})$ provides constructors like *negation*, *conjunction*, *disjunction*, *existential quantifier*, *universal quantifier* and *qualified number restrictions* to build complex concepts from simpler ones. Further, it supports concrete datatypes and there exist corresponding axioms for quantifiers and cardinality constraints for roles with a datatype range. A TBox consists of a finite set of concept inclusion axioms $C \sqsubseteq D$, where C and D are either both concepts or relations. The A-Box consists of a finite set of *concept assertions* $C(a)$, *role assertions* $R(a, b)$, *individual equalities* $a = b$, and *individual inequalities* $a \neq b$. Those assertional axioms or assertions introduce individuals, i.e. instances of a class, into the knowledge base and relate individuals with each other. For details about the semantics of $\mathcal{SHIQ}(\mathbf{D})$ constructors, T-Box axioms and A-Box axioms, we refer to [7, 12, 14].

We give μ-calculus atomic propositions (denoted by P above) the structure of the form $Q_F@Q_A$. A proposition $P = Q_F@Q_A$ is true if the $\mathcal{SHIQ}(\mathbf{D})$ concept Q_F has instances in the knowledge base of an agent that has non-functional properties described by the $\mathcal{SHIQ}(\mathbf{D})$ concept Q_A. We will see in the next section how user's agent selection criteria based on non-functional properties can be specified as $\mathcal{SHIQ}(\mathbf{D})$ concept.

For actions (see μ-calculus formulas $\langle a\rangle\phi$ and $[a]\phi$), we make similar structural extensions. We differentiate between input and output actions by using the sign '+' or '−' for input and output actions respectively. We give an input action a the structure $+(P, A, v_1{:}T_1, \ldots, v_m{:}T_m)@Q_A$, which means an agent satisfying the non-functional properties described in the query Q_A performs an input action that can receive m values of types $T_1, \ldots, T_m$ respectively over a channel of protocol P at the address A. Similarly, we use $-(P, A, Q_1, \ldots, Q_m)@Q_A$ for an output action, which means an agent satisfying the non-functional properties described in the query Q_A performs an output action that sends m values which are answers of the queries $Q_1, \ldots, Q_m$ respectively over a channel with protocol P and address A.

4.3. Specifying Constraints over Non-Functional Properties

Now we turn our attention to the '$@Q_A$' part of the structures that we introduced above. Q_A is a $\mathcal{SHIQ}(\mathbf{D})$ concept that specifies user's agent selection policy based on the non-functional properties of an agent. We have mentioned in Section 3.2, that a non-functional property can be interpreted as a set of agents that have the property. Considering the set-theoretic

semantics of $\mathcal{SHIQ}(\mathbf{D})$, we can view a non-functional property as a $\mathcal{SHIQ}(\mathbf{D})$ concept. By doing this, on one side it becomes possible to use $\mathcal{SHIQ}(\mathbf{D})$ constructors to build complex policies on the other side $\mathcal{SHIQ}(\mathbf{D})$ subsumption relations can be used to obtain interoperability among the non-functional properties.

Having introduced our goal specification language, we now look back to the requirements identified in Section 2 and discuss how our language covers the requirements. In an input formula $+(P, A, v_1{:}T_1, \ldots, v_m{:}T_m)@Q_A$, $T_1, \ldots, T_n$ are desired input types and in an output formula $-(P, A, Q_1, \ldots, Q_m)@Q_A$, $Q_1, \ldots, Q_m$ are description logic queries. So Requirement 1 is fulfilled. The queries $Q_1 \ldots, Q_m$ can use the process variables, e.g. those in an input formula. This way it becomes possible to establish relationships between inputs and outputs thus covering the Requirement 2. Requirement 3 is covered by the fact that we use description logic $\mathcal{SHIQ}(\mathbf{D})$ for specifying constraints on terminologies. Simple mappings between terminologies can be expressed via subsumption relationship, whereas complex mappings with DL-safe rules [11]. DL-safe rules [14] are a decidable rule extension of description logics and subset of SWRL[12]. Introduction of propositions of the form $Q_F@Q_A$ covers the Requirement 4. The Requirement 5 is covered by the suffix "$@Q_A$" in propositions, input formulas and output formulas. Requirements 6 and 7 are covered by formula types $[a]\phi$ and $\langle a\rangle\phi$. Requirement 8 is fulfilled by the logical connectors available in the language.

5. Related Work

OWL-S Matchmaker [16] uses OWL-S Profile for describing Web services as well as describing the goal. Recall, that even if OWL-S Profile is designed for modeling pre and post conditions in addition to the types of input and output parameters of the Web services, there is still no concrete formalism fixed for describing the conditions. As a result, the goal specification reduces to the types of input and output parameters. So, it neither allows the specification of relationships between inputs and outputs nor constraints on the temporal structure of a Web service.

[6] presents a procedure for the verification using SPIN model checker of OWL-S process models by mapping them to PROMELA. SPIN model checker supports LTL as a constraint specification language. However, in order to be able to use the SPIN model checker, authors needed to abstract from the types of input and output parameters, which loses the added value of OWL-S.

A WSML goal specification consists of capability and interfaces. A capability description consists of shared variables, pre- and post conditions as well as assumptions and effects. Due to the availability of shared variables WSML goals are more expressive than OWL-S Profiles. The interface description is used to specify the desired choreography and orchestration of a Web service. In other words, WSMO interface used in a goal specifies constraints on the dynamic behavior of a Web service. However, currently there is no concrete proposal for describing the choreography and orchestration. Furthermore, due the separation of capabilities and dynamic behavior already at the conceptual level, one would not be able to specify temporal constraints on effects. That is, constraints like "credit card should be charged (effect) should take place after delivery (choreography)". Furthermore, WSMO goal specification does not address trust and access control policies. However, we believe that access control policies can be integrated in WSML as pre-conditions.

Perhaps, the work that is closest to our work is [8], in which an approach is presented to characterize Web services with their transition behavior and their impacts on the real world (modeled as relational databases). We model the local knowledge bases of the participating actors with decidable description logics, which can be helpful in proving decidability of discovery and composition algorithms and also more suitable since the Web ontology language OWL standardized by W3C is based on description logics. Another major difference is the choice of the logic for specifying constraints on Web services. [8] uses propositional dynamics logic (PDL) whereas we have used a more expressive logic μ-calculus.

6. Conclusion and Outlook

In this paper, we have developed an expressive formalism for modeling goals. We showed how temporal and security constraints as well as constraints on the objects involved in a Web service process can be specified with one logic. The logic developed in this paper is a novel combination of $\mathcal{SHIQ}(\mathbf{D})$ and μ-calculus. We have shown in our earlier works, that credentials of an agent can be modeled with description logics and the verification of eligibility can be checked with description logic queries [3, 4]. Hence, our goal specification formalism presented in this paper covers resources, behavior and access control/trust policies of the agents. We have implemented a Java API for modeling goals programmatically. Furthermore, we

have developed a concrete syntax for the language so that the formulas can be written by an end user at user interface. The connection between concrete syntax and the Java API is established by a parser generated with the well-known JavaCC[2] tool (see [1] and `http://kasws.sourceforge.net/` for more details about the implementation).

For Web service descriptions having the semantics of the formal model presented in Section 3, it becomes possible to develop an expressive discovery based on local model checking techniques known for μ-calculus and query answering for description logics [2]. In some situations however, there may be combinations of Web services that fulfill a given goal. The problem of finding such combinations is often referred to as automatic composition and there is already some work done to address this problem. Most of the approaches are based on AI-planning techniques, in which a goal denotes a situation that should be reached. However, we believe that classical AI planning techniques are not sufficient for the problem of automatic composition of Web services, since a goal (as in our goal specification) may just describe constraints on the overall properties of a composite Web service and not only a final state that should be reached. Developing automatic Web services composition algorithms for goals specified in our goal specification language is an interesting and open research problem.

Acknowledgements

Research reported in this paper was partially supported by the EU in the IST project NeOn (IST-2006-027595, `http://www.neon-project.org/`)).

References

[1] S. Agarwal. *Formal Description of Web Services for Expressive Matchmaking.* PhD thesis, University of Karlsruhe (TH), May 2007.

[2] S. Agarwal. Model Checking Expressive Web Service Descriptions (Short Paper). In *IEEE 5th International Conference on Web Services*, Salt Lake City, Utah, USA, July 2007. IEEE Computer Society.

[3] S. Agarwal and B. Sprick. Access Control for Semantic Web Services. In L.-J. Zhang, editor, *2nd International Conference on Web Services*, pages 770–773, San Diego, California, USA, July 2004. IEEE Computer Society.

[4] S. Agarwal and B. Sprick. Specification of Access Control and Certification Policies for Semantic Web Services. In K. Bauknecht, B. Pröll, and H. Werthner, editors, *6th International Conference on Electronic Commerce and Web Technologies*, volume 3590 of *Lecture Notes in Computer Science*, pages 348–357, Copenhagen, Denmark, August 2005. Springer.

[5] S. Agarwal and R. Studer. Automatic Matchmaking of Web Services. In L.-J. Zhang, editor, *IEEE 4th International Conference on Web Services*, pages 45–54, Chicago, USA, September 2006. IEEE Computer Society.

[6] A. Ankolekar, M. Paolucci, and K. P. Sycara. Towards a Formal Verification of OWL-S Process Models. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2005.

[7] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory Implementation and Applications.* Cambridge University Press, 2003.

[8] D. Berardi, D. Calvanese, G. D. Giacomo, R. Hull, and M. Mecella. Automatic Composition of Transition-Based Semantic Web Services with Messaging. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, and B. C. O. Per-ke Larson, editors, *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 613–624, Trondheim, Norway, August-September 2005. ACM.

[9] M. Bishop. *Computer Security – Art and Science.* Addison Wesley, 2003.

[10] D. E. Denning. *Cryptography and Data Security.* Addison Wesley, 1982.

[11] P. Haase and B. Motik. A Mapping System for the Integration of OWL-DL Ontologies. In A. Hahn, S. Abels, and L. Haak, editors, *IHIS 05: Proc.s of the 1st Int. Workshop on Interoperability of Heterogeneous Information Systems*, pages 9–16. ACM Press, November 2005.

[12] I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.

[13] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[14] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proc. of the 3rd. Int. Semantic Web Conference (ISWC 2004)*, volume 3298 of *LNCS*, Hiroshima, Japan, November 2004. Springer.

[15] P. Samarati and S. Capitani di Vimercati. Access Control: Policies, Models, and Mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design (FOSAD)*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. FOSAD 2000, Bertinoro, Italy, Springer Verlag, Berlin, October 2001.

[16] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated Discovery, Interaction and Composition of Semantic Web Services. *Journal of Web Semantics*, 1(1):27–46, December 2003.

2 `https://javacc.dev.java.net/`