Building the WSMO-Lite Test Collection on the SEALS Platform

Liliana Cabral, Ning Li, and Jacek Kopecký

KMI, The Open University
Milton Keynes, UK
{L.S.Cabral,N.Li,J.Kopecky}@open.ac.uk

Abstract. We present a test collection for WSMO-Lite that is suitable for evaluating systems, tools or algorithms for Semantic Web Service discovery or matchmaking. We describe the design of the test collection and how the collection has been implemented on the SEALS platform. In addition, we discuss lessons learned with respect to the WSMO-Lite ontology and our implementation of the test collection.

Keywords: Semantic Web Service Evaluation, WSMO-Lite, Test collection, Service Discovery, Service matchmaking

1 Introduction

Semantic Web Service (SWS) technologies enable the automation of discovery, selection, composition, mediation and execution of Web Services by means of semantic descriptions of their interfaces, capabilities and non-functional properties. SWS build on Web service standards such as WSDL[1], SOAP[2] and REST (HTTP), and as such provide a layer of semantics for service interoperability. Current results of SWS research and industry efforts include a number of reference service ontologies, such as OWL-S[3], WSMO[4] and WSMO-Lite[5] and semantic annotation extension mechanisms, as provided by SAWSDL[6], SA-REST[7] and MicroWSMO[8].

The SWS discovery activity consists of finding Web Services based on their semantic descriptions. Tools for SWS discovery or matchmaking can be evaluated on retrieval performance, where for a given goal, i.e. a semantic description of a service request, and a given set of service descriptions, i.e. semantic descriptions of service offers, the tool returns the match degree between the goal and each

[1] http://www.w3.org/TR/wsdl
[2] http://www.w3.org/TR/soap
[3] http://www.w3.org/Submission/OWL-S/
[4] http://www.wsmo.org
[5] http://www.w3.org/Submission/WSMO-Lite/
[6] http://www.w3.org/2002/ws/sawsdl/
[7] http://www.w3.org/Submission/SA-REST/
[8] http://sweet.kmi.open.ac.uk/pub/microWSMO.pdf

Proceedings of the Second International Workshop on
Evaluation of Semantic Technologies (IWEST 2012)
May 28th, Heraklion, Greece
CEUR Workshop Proceedings Vol. 843

service, and the evaluation platform measures the rate of matching correctness based on a number of metrics.

Currently, there are two de-facto test collections, namely OWLS-TC[9] and SAWSDL-TC[10] used for the evaluation of OWL-S and SAWSDL matchmaking algorithms respectively, which have been used as part of the S3[11] evaluation initiative. In conjunction with this, there are other test collections provided in natural language that allows evaluations across different semantic annotation approaches, such as the JGD[12] test collection used in one of the tracks of the S3 contest, and the Discovery scenarios from the Semantic Web Service Challenge[13].

As known from evaluation communities, the publication of test data collections foster the uptake and evaluation of current standards as well as related technologies. In this paper we provide a test collection for WSMO-Lite - WSMO-LITE-TC - that is combined with SAWSDL as well as formats for Semantic Web rules. Our goal was to create a test collection that leveraged the annotation features of WSMO-Lite for enabling automatic discovery of Web Services. As WSMO-Lite has been prescribed to complement SAWSDL, we have created the WSMO-LITE-TC initially as a counterpart of SAWSDL-TC and extended the test collection with annotations specific to the WSMO-Lite ontology. We also planned additional variants in order to account for the use of rules in formats such as RIF[4] and SPIN[5], or from languages such as WSML, which are not present in existing test collections.

As a more recent initiative on the evaluation of Semantic technologies, the SEALS (Semantic Evaluation at Large Scale) project[14] has undertaken the task of creating a lasting reference infrastructure for semantic technology evaluation - the SEALS platform - which is technology independent, open, scalable and extensible. The platform will allow online evaluation of semantic technologies by providing access to an integrated set of evaluation services and test collections. Semantic Web Services are one of the technologies supported by SEALS [1]. The platform supports the creation and sharing of evaluation artifacts (e.g. datasets and measures) and services (e.g. retrieving data sets from repositories and automatic execution of tools), making them widely available according to evaluation scenarios, using semantic based terminology.

WSMO-LITE-TC is one of the outcomes of the SEALS project and as such adopts the SEALS test suite metadata and it is available through the SEALS Dataset repository. Accordingly, the WSMO-LITE-TC can be accessed using the SEALS platform services. In this paper, we describe the design of the test collection and how it has been implemented on the SEALS platform. In addition, we discuss lessons learned with respect to the WSMO-Lite ontology and our implementation of the test collection.

[9] http://semwebcentral.org/projects/owls-tc/
[10] http://semwebcentral.org/projects/sawsdl-tc/
[11] http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html
[12] http://fusion.cs.uni-jena.de/professur/jgd
[13] http://sws-challenge.org/wiki/index.php/Scenarios
[14] http://about.seals-project.eu

The remainder of the paper is organised as follows. In Section 2 we provide an overview of the related evaluation initiatives and test collections. In Section 3 we provide an overview of WSMO-Lite. In Section 4 we describe the design of WSMO-LITE-TC, considering previous test collections and the features of the WSMO-Lite ontology. In Section 5 we present the implementation details of WSMO-LITE-TC. In Section 6 we discuss our results, and finally in Section 7 we present our conclusions and future work.

2 Related Work

The evaluation of Semantic Web Services is currently being pursued by a few initiatives using different evaluation methods. In particular, we refer to the S3 (Semantic Service Selection) contest and corresponding test collections, as mentioned previously.

S3 is about the retrieval performance evaluation of matchmakers for Semantic Web Services. It is a virtual and independent contest, which runs annually since 2007. It provides the means and a forum for the joint and comparative evaluation of publicly available Semantic Web service matchmakers over given public test collections. S3 features three tracks: OWL-S matchmaker evaluation (over OWLS-TC); SAWSDL matchmaker evaluation (over SAWSDL-TC); and cross evaluation (using JGD collection). The participation in the S3 contest consists of: a) implementing the SME2[15] plug-in API for the participant's matchmaker together with an XML file specifying additional information about the matchmaker; and b) using the SME2 evaluation platform for testing the retrieval performance of the participant's matchmaker over a given test collection. This platform has a number of metrics available and provides comparison results in graphical format. The presentation and open discussion of the results with the participants is performed by someone from the organisational board at some event like the SMR2 (Service Matchmaking and Resource Retrieval in the Semantic Web) workshop.

The OWL-S Test Collection (OWLS-TC) is intended to be used for evaluation of OWL-S matchmaking algorithms. OWLS-TC is used worldwide (it is among the top-10 download favourites of semwebcentral.org) and the de-facto standard test collection so far. It has been initially developed at DFKI, Germany, but later corrected and extended with the contribution of many people from a number of other institutions (including e.g. universities of Jena, Stanford and Shanghai, and FORTH). The OWLS-TC4 version consists of 1083 semantic web services described with OWL-S 1.1, covering nine application domains (education, medical care, food, travel, communication, economy, weapons, geography and simulation). OWLS-TC4 provides 42 test queries associated with binary as well as graded relevance sets. The relevance sets were created with the SWS-RAT (Semantic Web Service Relevance Assessment Tool) developed at DFKI. The graded relevance is based on a scale using 4 values: highly relevant, relevant, potentially relevant, and non-relevant. 160 services and 18 queries contain Precondition and/or Effect as part of their service descriptions.

[15] `http://semwebcentral.org/projects/sme2/`

The SAWSDL Test Collection (SAWSDL-TC) is a counterpart of OWLS-TC, that is, it has been semi-automatically derived from OWLS-TC. SAWSDL-TC is intended to support the performance evaluation of SAWSDL service matchmaking algorithms. The SAWSDL-TC3 version provides 1080 semantic Web services written in SAWSDL (for WSDL 1.1) and 42 test queries with associated relevance sets. Model references point to concepts described in OWL2-DL exclusively.

3 Overview of WSMO-Lite

WSMO-Lite (Lightweight Semantic Descriptions for Services on the Web)[2] is a recent (Aug, 2010) submission to W3C. It can be used for annotations of various WSDL elements using the SAWSDL annotation mechanism. It exploits RDF and RDFS as well as their various extensions such as OWL and RIF for semantic service descriptions. WSMO-Lite annotations cover semantic aspects of Web services, and are intended to support tasks such as (semi-)automatic discovery, negotiation, composition and invocation of services. Inspired by the distinctions made by Sheth [7], WSMO-Lite groups service semantics into four orthogonal parts:

- Functional description specifies service *functionality*, that is, what a service can offer to its clients when it is invoked.
- Nonfunctional description defines any *contract and policy* parameters of a service, or, in other words, incidental details specific to the implementation or running environment of the service.
- Behavioral model specifies the *actions* and the *process* (in other words, the operations and their ordering) that a client needs to follow when consuming a service's functionality.
- *Information model* defines the input, output and fault messages of the actions.

Informally, WSMO-Lite represents these types of semantics as follows:

- Functional semantics are represented in WSMO-Lite as *capabilities* and/or functionality *classifications*. A capability defines *preconditions* which must hold in a state before the client can invoke the service, and *effects* which hold in a state after the service invocation. Functionality classifications define the service functionality using some classification ontology (i.e., a hierarchy of *categories*). [16]
- Nonfunctional semantics are represented using an ontology that semantically captures some policy or other nonfunctional properties.
- Behavioral semantics are represented by annotating the service operations with functional descriptions, i.e., capabilities and/or functionality classifications. These functional annotations of operations can serve for ordering of operation invocations.

[16] The distinction of capabilities and categories is the same that is made by Sycara et al. [8] between "explicit capability representation" (using taxonomies) and "implicit capability representation" through preconditions and effects.

– Information semantics are represented using domain *ontologies*, which are also involved in the descriptions of the other types of semantics.

WSMO-Lite is formally materialized as an RDFS ontology as shown in Listing 1.1(N3 format).

```
1 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2 @prefix wl: <http://www.wsmo.org/ns/wsmo-lite#> .
3
4 wl:FunctionalClassificationRoot rdfs:subClassOf rdfs:Class .
5 wl:Condition a rdfs:Class .
6 wl:Effect a rdfs:Class .
7 wl:NonfunctionalParameter a rdfs:Class .
```

Listing 1.1. WSMO-Lite Service Semantics Ontology in RDFS

In the interest of simplicity of the RDF form of actual concrete semantic service descriptions, the ontology is not a straightforward implementation of the formal terms (such as *classification*, *capability*, or *ontology for nonfunctional semantics*). We discuss below some of the considerations that led to the proposed form of the ontology classes.

wl:FunctionalClassificationRoot is a class that marks the roots of service functionality classifications. All subclasses of the root class are included in the particular classification. Authoring tools can suggest all known instances of *wl:FunctionalClassificationRoot* and their subclasses for annotating the functionality of services and operations.

wl:Condition, wl:Effect together form a *capability* in a functional service description. Instances of these classes are expected to contain some logical expressions. In common with OWL-S, the WSMO-Lite service ontology does not specify the concrete language for the logical expressions, or their processing. Logical expressions can be specified in any suitable language, such as SWRL [3] or RIF [4], and embedded in RDF semantic descriptions as literals.

A precondition and an effect implicitly make up a capability of a service (or of a particular operation). WSMO-Lite does not model the capability itself in the RDF ontology, as as it would be an unnecessary indirection between the service (or operation) and its capability's precondition and effect.

wl:NonfunctionalParameter is a class of concrete, domain-specific nonfunctional parameters. For a particular ontology of nonfunctional semantics, its instances would be instances of this class. WSMO-Lite places no further restrictions on nonfunctional parameters; research in this area, which is out of scope of this article, has not yet converged on a common set of properties that nonfunctional parameters should have.

Finally, there is no explicit marker for information ontologies used to describe the information model of the service. Earlier versions of WSMO-Lite specified the class *wl:Ontology*, but it turned out not to be useful in practice and is about to be deprecated. The TC described in this article does not use this class.

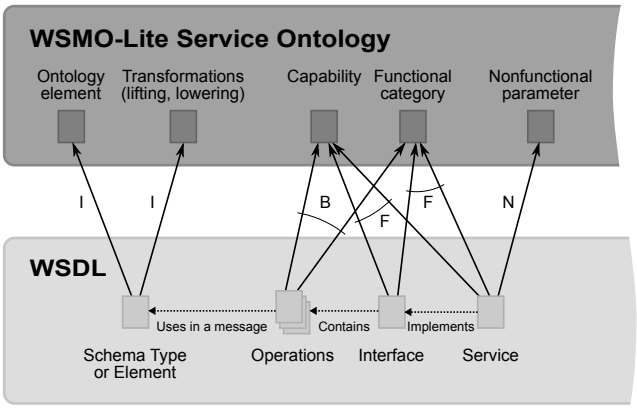

Fig. 1. Use of WSMO-Lite semantics on WSDL components annotated with SAWSDL; annotations: F—functional, N—nonfunctional, B—behavioral, I—information model

3.1 Relation to SAWSDL

Until 2007, there were no agreed standards for semantic Web services; then the World Wide Web Consortium (W3C) produced its Recommendation called SAWSDL [6], based on an earlier effort called WSDL-S. SAWSDL is not a fully-fledged SWS framework; instead it only provides hooks in WSDL where semantic annotations can be attached, leaving the specification and standardization of concrete service semantics for later. WSMO-Lite investigates such concrete semantics.

In SAWSDL, the main annotation property is a *model reference*, which points from any WSDL component to the associated semantics. Each concrete model reference value is always identified with a URI. Multiple values of a model reference on a single component all apply to the component; for example, an input message element can be associated with two ontology classes because this one element contains information that fits both of those classes.

Since SAWSDL presents only a thin annotation layer over WSDL but no actual semantics, it must be complemented by ontologies that express the semantics of the various WSDL components. **WSMO-Lite** is such an ontology. Figure 1 illustrates graphically the WSMO-Lite annotations in relation to WSDL. In short, WSDL services and interfaces can be annotated with model references to functional semantics by using the classes *wl:FunctionalClassificationRoot*, *wl:Condition* and *wl:Effect*; the same classes used in model reference annotations on operations express behavioral semantics of the service. A service can also be annotated with model references to nonfunctional properties (instances of the class *wl:NonfunctionalParameter*), and operation input and output messages can be annotated with model references to ontology elements, or with lifting and lowering schema mapping pointers to data transformations.

Note that while SAWSDL only describes the use of its *modelReference* annotations on WSDL *interface* components (along with some of their subcompo-

nents, such as *operations*) and on XML Schema *element declaration* and *type definition* components, it allows the annotation of all the other components in WSDL, including *service*. The use of *modelReference* annotations on *service* components in WSMO-Lite is fully within the spirit of SAWSDL.

4 WSMO-Lite TC Design

WSMO-LITE-TC has been initially designed as a counterpart of SAWSDL-TC, and as such contains the same number of queries and services descriptions as SAWSDL-TC in addition to the rules descriptions from OWLS-TC. In future extensions we will provide variants of WSMO-LITE-TC according to the language used for rules. Thus, accordingly, we created the first variant using OWL and SWRL and have planned additional variants for PDDL, RIF, SPIN and WSML.

More specifically, our design approach consisted of a) deriving WSMO-LITE-TC from SAWSDL-TC, keeping the OWL semantic annotations for inputs and outputs; b) adding functional classification annotations to the service according to the domains defined for these collections, and c) adding (pre) conditions and effects annotations to the service operations according to the SWRL rules provided in OWLS-TC.

We use the Web Service (WSDL) extract shown in Listing 1.2 taken from the WSMO-LITE-TC (file bookperson_price_service.wsdl) as an example to describe the WSMO-Lite annotations next.

```
1  name="PersonbookPriceSoap"> <wsdl:operation name="get_PRICE" sawsdl:
       modelReference="http://127.0.0.1/rules/SWRL/
       bookperson_price_service_SWRL.owl#AuthorizedPerson">
2      <wsdl:input message="get_PRICERequest"/>
3      <wsdl:output message="get_PRICEResponse"/>
4    </wsdl:operation>  </wsdl:portType>
5  <wsdl:service name="PersonbookPriceService" sawsdl:modelReference="http
       ://127.0.0.1/ontology/serviceCategories.owl#Economy">
6    <wsdl:port name="PersonbookPriceSoap" binding="
       PersonbookPriceSoapBinding">
7      <wsdlsoap:address location="http://127.0.0.1/services/PersonbookPrice"/>
8    </wsdl:port> </wsdl:service>
```

Listing 1.2. Web Service (WSDL) Annotations Example from WSMO-LITE-TC.

4.1 Functional Classification Annotations

As explained in Section 3, the WSMO-Lite functionality classification is used to define the service functionality using some classification ontology. Currently, the services in OWLS-TC and SAWSDL-TC are coarsely classified into nine domains (communication, economy, education, food, medical, simulation, travel, weapon and geography). Thus, we decided to use these domains as service categories for functional classification annotation. Since there was no single ontology that formally described those domains, we created a new ontology, referred to as *Service Category Ontology*, as shown in Listing 1.3. As can be seen, the domain

classes are subclasses of the class called EvaluationDomain, a type of WSMO-Lite *FunctionalClassificationRoot* class. Therefore, these classes can be used for the annotations of services with respect to its functional classification. Accordingly, service discovery tools based on the WSMO-Lite service ontology can be developed to search for services according to the service functional classification.

```
1  @prefix owl: <http://www.w3.org/2002/07/owl#> .
2  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3  @prefix wl: <http://www.wsmo.org/ns/wsmo-lite#> .
4  :EvaluationDomain a owl:Class,  wl:FunctionalClassificationRoot .
5  :Communication a owl:Class; rdfs:subClassOf :EvaluationDomain  .
6  :Economy a owl:Class; rdfs:subClassOf :EvaluationDomain  .
7  :Education  a owl:Class; rdfs:subClassOf :EvaluationDomain  .
8  :Food a owl:Class; rdfs:subClassOf :EvaluationDomain  .
9  :Geography a owl:Class;  rdfs:subClassOf :EvaluationDomain  .
10 :Medical a owl:Class;  rdfs:subClassOf :EvaluationDomain  .
11 :Simulation a owl:Class;  rdfs:subClassOf :EvaluationDomain  .
12 :Travel a owl:Class;  rdfs:subClassOf :EvaluationDomain  .
13 :Weapon a owl:Class;  rdfs:subClassOf :EvaluationDomain  .
```

Listing 1.3. Service Category Ontology.

As illustrated in the exemplar service description shown in Listing 1.2, we annotated the service by attaching the *Economy* category to the *wsdl:service* element via the *sawsdl:modelReference* extension element. Accordingly, the same is done for the Goals (queries) in the test collection.

4.2 Condition and Effect Annotations

We follow on from Section 3 for representing conditions and effects by hooking an ontological abstraction of language-dependent descriptions of conditions and effects to service operations through SAWSDL modelReference.

OWLS-TC uses SWRL as one of the languages for describing service conditions and effects, whereas, in SAWSDL-TC, conditions and effects are not present. Thus, we decided to derive all logical expressions from the corresponding services and queries from OWLS-TC and represent them as types of WSMO-Lite class Condition and Effect accordingly. These are then hooked to service operations through SAWSDL modelReference. As illustrated in the exemplar service description shown in Listing 1.2, we annotated the operation by attaching the *AuthorizedPerson* condition to the *wsdl:operation* element via the *sawsdl:modelReference* extension element. Accordingly, the same is done for the conditions in Goals (queries) in the test collection.

To create the corresponding WSMO-Lite annotations from OWL-S descriptions we applied the following mappings: 1) the *owls:hasPrecondition* is mapped to *wl:Condition*; 2) since no *owls:Result* has more than one *owls:Effect*, the *owls:Effect* is mapped to *wl:Effect*. This is also due to the fact that they are similar in the way that they are both described with a logic language and they share the same semantics; 3) the *owls:inCondition* is ignored in WSMO-LITE-TC because it is not, unlike *owls:hasPrecondition*, a pre-requirement that must

be satisfied for using the service; instead, it is a condition depending on the running status of the service; 4) the local references to OWL-S elements are replaced with the corresponding element in the rule language (e.g. SWRL variables) and corresponding datatype.

The representation in WSMO-Lite of the condition *AuthorizedPerson* from the example in Listing 1.2 is shown in Listing 1.4. The *AtomList* part of the rule declaration has been extracted from the corresponding service in OWLS-TC, but the value of the local value used in *argument1* has been replaced with the corresponding datatype. This has been done for all rules in order for them to stand alone.

```
1  <rdf:Description rdf:about="http://127.0.0.1/rules/SWRL/
       bookperson_price_service_SWRL.owl#AuthorizedPerson">
2    <rdf:type rdf:resource="http://www.wsmo.org/ns/wsmo-lite#Condition"/>
3    <rdf:value rdf:parseType="Literal">
4      <swrl:AtomList xmlns:swrl="http://www.w3.org/2003/11/swrl#" xmlns:rdf="
       http://www.w3.org/1999/02/22-rdf-syntax-ns#">
5        <rdf:first>
6          <swrl:ClassAtom>
7            <swrl:classPredicate rdf:resource="http://127.0.0.1/ontology/core-plus-
       office.owl#Authorized"></swrl:classPredicate>
8              <swrl:argument1 rdf:resource="http://127.0.0.1/wsdl/PersonbookPrice#
       _BOOK"></swrl:argument1>
9          </swrl:ClassAtom>
10       </rdf:first>
11       <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil
       "></rdf:rest>
12     </swrl:AtomList>
13   </rdf:value>
14 </rdf:Description>
```

Listing 1.4. WSMO-Lite Condition Rule (SWRL) Example.

5 WSMO-Lite TC Implementation

WSMO-LITE-TC is available from the SEALS Testdata repository[17]. Through this URL the metadata of the test collection can be accessed. To retrieve the test data ZIP file, the value of the Accept header in the HTTP request needs to be changed to "application/zip"[18]. The URL also indicates the rule language used for annotations (suffix *SWRL*) and whether the collection contains binary or graded relevance values for the matching judgements (suffix *b* or *g* respectively). The metadata for describing testdata is provided by the SEALS metadata ontology[19].

The test collection, encapsulated in a ZIP file, includes the testsuite metadata, within the file named Metadata.rdf, which is used to describe and retrieve testdata. The metadata defines a *Suite*, which consists of multiple *SuiteItems*,

[17] http://seals.sti2.at/tdrs-web/testdata/persistent/WSMO-LITE-TC-SWRL/1.0-4b
[18] To change the Accept header the Firefox add-on Modify Headers can be used
[19] http://www.seals-project.eu/ontologies/SEALSMetadata.owl

which in turn contain *DataItems*. A *DataItem* refers to the location of a file containing evaluation data. An example of the testsuite metadata instance for WSMO-LITE-TC is shown in Listing 1.5. This instance describes the SWS Discovery testsuite[20]. Each suiteItem represents a testdata consisting of data items describing a goal and related services and expert judgements.

The repository also offer RESTfull services to access data items within a specific suite item in the test collection. For example, the repository allows access to a specific service description[21], goal description[22] or judgement document (with binary relevance values)[23] within a testdata.

```
1   @prefix purl: <http://purl.org/dc/terms/> .
2   @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3   @prefix seals: <http://www.seals-project.eu/ontologies/SEALSMetadata.owl
      #> .
4   <#testdata22>     a seals:SuiteItem;
5       purl:identifier "22";
6       seals:hasDataItem :binaryRelevance22, :goal22, :service707, :service805
7       ...
8   :binaryRelevance22     a seals:DataItem;
9       purl:identifier "binaryRelevance22";
10      seals:hasComponentType "JudgementDocument";
11      seals:isLocatedAt "./relevance/ binaryRelevance22.xml" .
12  :goal22     a seals:DataItem;
13      purl:identifier "goal22";
14      seals:hasComponentType "GoalDocument";
15      seals:isLocatedAt "./goals/shoppingmall_cameraprice_service.wsdl" .
16  :service707     a seals:DataItem;
17      purl:identifier "service707";
18      seals:hasComponentType "ServiceDocument";
19      seals:isLocatedAt "./services/printedmaterialperson__service.wsdl" .
20  :service805     a seals:DataItem;
21      purl:identifier "service805";
22      seals:hasComponentType "ServiceDocument";
23      seals:isLocatedAt "./services/shoppingmall_cameraprice_service.wsdl" .
```

Listing 1.5. Testsuite Metadata Example.

The files within the test collection (ZIP file) are organized into six folders as follows. The *ontology* folder contains all the original SAWSDL ontology files referred to by any other files. It also contains the ServiceCategories Ontology (described in Section 4). The *rules* folder contains files describing WSMO-Lite conditions and effects (SWRL rules), which were generated from the original ones in OWLS-TC with names appended with the sufix "_SWRL". The *services* folder contains the service files (WSDL) generated from the original SAWSDL-TC and extended with WSMO-Lite annotations. The *goals* folder contains the goal files

[20] ../WSMO-LITE-TC-SWRL/1.0-4b/suite/
[21] ../WSMO-LITE-TC-SWRL/1.0-4b/suite/22/service707
[22] ../WSMO-LITE-TC-SWRL/1.0-4b/suite/22/goal22
[23] ../WSMO-LITE-TC-SWRL/1.0-4b/22/binaryRelevance22

(WSDL) generated from the original SAWSDL-TC and extended with WSMO-Lite annotations. The *relevance* folder contains XML files which describe, each, the binary (or graded) relevance judgements between a goal and a set of services. The *liftingSchemaMapping* folder contains the lifting schema mapping files from the original SAWSDL-TC.

6 Discusssion

The WSMO-LITE Test collection described in this paper is intended for evaluation of SWS discovery tools. However, the test collection itself can be evaluated on the coverage of the features of the underlying formalism and in our case we covered most of the WSMO-Lite constructs intended for discovery. More importantly, a test collection can be evaluated on how useful it is for tool developers, their testing and cross-tool comparisons. As we have built WSMO-LITE-TC as a counterpart of the existing SAWSDL-TC and OWLS-TC, we believe that this will complement previous efforts and benefit the comparison across the underlying formalisms.

When it comes to facilitate different service tasks, not all types of annotations are always needed, namely informational, functional and non-functional. For service discovery, an algorithm can operate on functional descriptions, i.e. capabilities or categories. The algorithm may check (in case conditions are present) that the goal complies with the service's conditions, and that the service effects fit the goal. With categories, the goal specifies a need for services in a given category, so the discovery algorithm can check for (sub)category membership as a filtering mechanism. More generally, these three groups of semantic descriptions should be working closely with each other to jointly decide the level of match of offer services to a given goal in order to determine the final similarity score. Compared to SAWSDL-TC, the features that have been added from WSMO-LITE-TC are the service and goal's functional categories and the operation's (pre)conditions and effects. Non-functional properties can also be added as annotations, but these were not present in the existing collections.

With respect to the way in which OWLS-TC and SAWSDL-TC use the same template for both service descriptions and for goal descriptions (queries), we consider that WSMO-Lite does not rely on any given representation for discovery queries. We argue that SWS discovery test collections could have plain-text discovery queries that discovery tool implementers can translate to any suitable formal structure supported by their tools. A plain-text form can mirror closely typical kinds of discovery queries as they would be formulated by users.

Regarding the underlying formalism, WSMO-Lite is intended as a step towards convergence of earlier frameworks on top of SAWSDL as shown in Section 3. It is intentionally lightweight and independent of any nonstandard languages and tools. In addition, WSMO-Lite is not constrained to WSDL-based services, as it can also be applied to Web APIs.

7 Conclusions and Future Work

WSMO-LITE-TC is the first test collection in the evaluation community that uses the WSMO-Lite service ontology to describe service semantics. We hope

that this test collection will foster the use of WSMO-Lite as well as clarify the use of its annotation features.

There are currently very few WSMO-Lite-compatible discovery tools to allow a fair evaluation. But, we envisage that a comparison across the languages in the three corresponding test collections would be beneficial for the SWS community.

In this paper we have presented the implementation of the first variant of the WSMO-LITE-TC, which uses OWL as the ontology language and SWRL for the representation of conditions and effects. The details presented here can be used by evaluators, who intend to provide their own test collections via the SEALS platform.

In the future we intend to provide implementations of other variants of WSMO-LITE-TC using different languages as mentioned in the introduction. As WSMO-Lite is disassociated from the representation language, it allows us to be flexible with respect to evaluation requirements while using the same test collection design.

Acknowledgments. This work has been partially funded by the European Commission under the SEALS project (FP7-238975).

References

1. Cabral, L. and Toma, I.: Evaluating Semantic Web Services Tools using the SEALS Platform. In: IWEST Workshop at ISWC 2010, Shanghai, China (2010)
2. Kopecky, J. and Vitvar, T.: WSMO-Lite: Lowering the Semantic Web Services Barrier with Modular and Light-weight Annotations. In proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008), Santa Clara, CA, USA (2008)
3. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S, Grosof, B. and Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical report, Joint US/EU ad hoc Agent Markup Language Commit- tee. Available at http://www.daml.org/2003/11/swrl/ (2003)
4. RIF Core Dialect. W3C Recommendation 22 June 2010. Available at http://www.w3.org/TR/rif-core/ (2010)
5. SPIN. W3C Member Submission 22 February 2011. Available at http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/ (2011)
6. Semantic Annotations for WSDL and XML Schema. Recommendation, W3C, August 2007. Available at http://www.w3.org/TR/sawsdl/ (2007)
7. Sheth, A.: Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration. Invited Talk, Workshop on E-Services and the SemanticWeb, at World WideWeb Conference. Available at http://lsdis.cs.uga.edu/lib/presentations/WWW2003-ESSW-invitedTalk-Sheth.pdf (2003)
8. Sycara, K., Paolucci, M., Ankolekar, A. and Srinivasan, N.: Automated Discovery, Interaction and Composition of Semantic Web services. Web Semantics: Science, Services and Agents on the World Wide Web, 1(1):27–46 (2003).