

OWLS-TC

OWL-S Service Retrieval Test Collection

Version 4.0

User Manual

September 20th, 2010

Matthias Klusch, Mahboob Alam Khalid, Patrick Kapahnke,
Benedikt Fries, Martin Vasileski

Saarbrücken, Germany

Table of contents

INTRODUCTION 3

INSTALLATION 4

 INSTALL XAMPP 4

STRUCTURE 6

 DOMAINS 6

 SERVICES 6

 REQUESTS AND RELEVANCE SETS 11

 USED ONTOLOGIES 13

DISCLAIMER 15

SUPPORT AND CONTACT 15

RELEASE HISTORY 15

Introduction

This is the fourth version of the OWL-S service retrieval test collection named OWLS-TC4. The collection is intended to support the evaluation of the performance of OWL-S service matchmaking algorithms. It provides 1083 semantic Web services written in OWL-S 1.1 (and for backward compatibility OWLS 1.0) from nine different domains (education, medical care, food, travel, communication, economy, weapons, geography and simulation). It provides a set of 42 test queries which are associated with relevance sets to conduct performance evaluation experiments.

Major improvements of OWLS-TC4 to former versions are the following:

- Added 60 services and 10 queries in the *geography* domain, translated from SAWSDL to OWL-S 1.1 from Jena Geography Dataset 50¹ test collection
- Added 16 services and 3 queries in the *simulation* domain, developed by DFKI for a project in simulated reality
- 160 services and 18 queries contain Precondition and/or Effect as part of their services' *process:AtomicProcess*. PDDL 2.1² and SWRL³ syntaxes are used for writing the condition expressions
- Revising and editing the text descriptions of all services
- Bug fixes

Part of the services contained in OWLS-TC4 were retrieved from public IBM UDDI registries, and semi-automatically transformed from WSDL to OWL-S. Up to this version, around 30 people (DFKI, University of Jena, University of Thessaloniki and others) have worked on improvements and extensions. Relevance sets have been defined collaboratively.

Please note that no *standard* test collection for OWL-S service retrieval does exist yet. As a consequence, OWLS-TC can only be considered as one possible starting point for any activity towards achieving such a standard collection by the community as a whole.

OWLS-TC is available at semwebcentral.org: <http://projects.semwebcentral.org/projects/owls-tc/>

¹ Jena Geography Dataset collection, <http://fusion.cs.uni-jena.de/professur/jgd/>

² International Planning Competition, <http://planning.cis.strath.ac.uk/competition/>

³ SWRL: A Semantic Web Rule Language, <http://www.w3.org/Submission/SWRL/>

Installation

1. Install a local web server like the “Apache HTTP Webserver”: We recommend to install **XAMPP** (see following subsection “Install XAMPP”).⁴
2. Copy the folders “services”, “ontology”, “queries” and “wsdl” of the OWLS-TC4 release to the public http root folder (e.g. apache/htdocs)

The usage of a local web server is necessary, since all ontologies used by a matchmaking tool, e.g. OWLS-MX, are expected to be available locally, and problems with incorrect paths to access ontologies used to describe service I/O concepts can thus be avoided.

Install XAMPP

Homepage: <http://www.apachefriends.org/en/xampp.html>

Download:

<http://prdownloads.sourceforge.net/xampp/xampp-win32-1.5.0-pl1-installer.exe?download>

XAMPP is an easy to install Apache HTTP web server distribution also containing MySQL, PHP and Perl. Despite the fact that it contains more applications than needed for OWLS-TC it allows a first time user to avoid the difficult configuration of the Apache HTTP web server.

The following manual should guide you through the process of installing XAMPP for the use with OWLS-TC, while using the above linked Windows version.

1. The installer starts with the language selection. Once you selected your desired language it will prompt you for an installation directory. You can select anything here, but we will consider the default “c:\apachefriends\xampp\” in this walkthrough.
2. The installation and configuration will take some minutes and temporarily open a command prompt. Once it is finished the installer will ask you if you want to install the web server as a windows service. Unless you want to start the local web server at windows startup you should deny this question. For the sole purpose of OWLS-TC the usual manual startup suffices.
3. Now you can start the Apache control panel, which will allow you to start the server by answering yes to the next question.
4. Using the control panel you can start the server by simply pressing the according Start” button next for the Apache. If you use Windows XP SP2 (or have any other firewall installed) your firewall will ask you if the program “Apache HTTP Server” should be blocked from the internet. It is necessary that you allow access to the program.
5. The installation of the web server is finished at this point. To install the test collection you can simply copy the directory “ontology”, “services”, “queries” and “wsdl” to the

⁴ The installer available from the Apache homepage in the Web, unfortunately, caused some problems with OWLS-MX on a WinXP Professional/Java1.5 equipped laptop.

apache web root directory (unless you specify something else it is “htdocs” in the XAMPP directory, so in our case “c:\apache\friends\xampp\htdocs”).

6. Now you can test if you can open the directory <http://127.0.0.1/services/> in your web browser.

The **archive of the test collection** contains the following **subdirectories**,

- **services**
All services of the test collection (subdirectories for the different supported OWLS-Versions). *Supposed to be copied to the root directory of the local web server.*
- **queries**
All services requests of the test collection (subdirectories for the different supported OWLS-Versions). *Supposed to be copied to the root directory of the local web server*
- **ontology**
Ontologies used by the services and service requests. *Supposed to be copied to the root directory of the local web server*
- **domains**
Contains all services sorted according to their domains
- **wsdl**
Contains the WSDL groundings for all of the services. *Supposed to be copied to the root directory of the local web server*

Structure

Domains

OWL-S Service Retrieval Test Collection version 4.0 consists of 1083⁵ indexed OWL-S services from the following 9 domains:

- 1- education
- 2- medical care
- 3- food
- 4- travel
- 5- communication
- 6- economy
- 7- weapon
- 8- geography
- 9- simulation

For information on the OWL-S web service description language standard of the W3C, please check, for example, <http://www.daml.org/services/>.

Services

The current status of the number of services and queries related to each of these domains are as follows.

DOMAIN	#services	#queries
education	286	6
medical care	73	1
food	34	1
travel	197	6
communication	59	2
economy	395	12
weapon	40	1
geography	60	10
simulation	16	3

The restrictions of OWL-S 1.1 service descriptions of this version of OWLS-TC are as follows:

(a) The services are not *describedby* Process Model, but instead are directly *describedby* Atomic process.

(b) Grounding in WSDL is provided for all of the OWL-S 1.1 services. These WSDL groundings are created from the existing OWL-S services using the tool OWLS2WSDL⁶. For each service, by using the option *owls2wsdl* from this tool, the WSDL file was created automatically

⁵ Some services appear in more than one category. Therefore the number of services is 1083 if just the first occurrence of each service is considered and 1140 if we consider repetitions across different categories.

⁶ <http://semwebcentral.org/projects/owls2wsdl/>

(although some manual changes had to be done, i.e. fixing the namespaces) and by using the option `owl-s` the groundings for the OWL-S services were created.

The concepts used in the service input/output parts refer to ontologies (.owl files) that are stored in the `/ontology` directory.

Preconditions and Effects

In this version of OWLS-TC 160 Semantic Web Service descriptions, including 18 service requests, are extended with preconditions (logical formulas that must hold true in order for the service to be successfully executed) and effects (logical formulas which describe the results of the service execution over the state of the world). For describing the expressions of these preconditions and effects, PDDL 2.1 (Planning Domain Definition Language) syntax and SWRL (Semantic Web Rule Language) syntax is used. These expressions are written as part of the service's *process:AtomicProcess* description.

PDDL syntax offers more freedom for defining condition formulas offering wide range of logical operators, like conjunction, disjunction, atomic negation or quantified formulas. On contrary, SWRL syntax is more restricted and doesn't support negation of whole atoms and disjunction of atoms. On the other hand this is not a major problem because by using rules of inference, disjunction and negation operators can be transformed in operators supported by SWRL. Classical negation of class predicates is supported in SWRL by using *owl:complementOf*.

The XML structure for describing preconditions and effects in PDDL is in the following format:

```
<process:hasPrecondition>
  <pddlexpr:PDDL-Expression>
    <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Precondition Formula
    </expr:expressionBody>
  </pddlexpr:PDDL-Expression>
</process:hasPrecondition>

<process:hasResult>
  <process:Result>
    <process:hasEffect>
      <pddlexpr:PDDL-Expression>
        <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          Effect Formula
        </expr:expressionBody>
      </pddlexpr:PDDL-Expression>
    </process:hasEffect>
  </process:Result>
</process:hasResult>
```

The precondition formulas follow this syntax:

- Atomic formula: (PREDICATE_NAME ?ARG1 ?ARG2 ... ?ARG_N)
- Conjunction of atomic formulas: (and ATOM1 ATOM2 ... ATOM_N)
General conjunction: (and CONDITION_FORMULA1 ... CONDITION_FORMULA_N)
- Disjunction of atomic formulas: (or ATOM1 ATOM2 ... ATOM_N)
General disjunction: (or CONDITION_FORMULA1 ... CONDITION_FORMULA_N)
- Negation of atomic formula: (not ATOMIC_FORMULA)
General negation: (not CONDITION_FORMULA)
- Quantified formula: (exists (?ARG1 ?ARG2 ...) CONDITION_FORMULA)

And the effect formulas follow this syntax:

- Added atom from service execution: (PREDICATE_NAME ?ARG1 ... ?ARG_N)
- Deleted atom from service execution: (not (PREDICATE_NAME ?ARG1 ... ?ARG_N))
- Conjunction of atomic effects: (and ATOM1 ... ATOM_N)
- Conditional effect: (when CONDITION_FORMULA EFFECT_FORMULA)

All predicates and arguments are represented by their corresponding URIs, which point to specific classes or object properties described in ontologies.

In the following example, precondition and effect formulas of the service *bookpersoncreditcardaccount__service.owl*s, used for purchasing books online, are described.

```
<process:hasPrecondition>
  <pddlexpr:PDDL-Expression>
    <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      (and
        (http://127.0.0.1/ontology/ontosem.owl#Accepted
         ?http://127.0.0.1/services/1.1/bookpersoncreditcardaccount__service.owl#_CREDITCARDACCOUNT)

        (http://127.0.0.1/ontology/core-plus-office.owl#Authorized
         ?http://127.0.0.1/services/1.1/bookpersoncreditcardaccount__service.owl#_PERSON)

        (http://127.0.0.1/ontology/Mid-level-ontology.owl#accountHolder
         ?http://127.0.0.1/services/1.1/bookpersoncreditcardaccount__service.owl#_PERSON
         ?http://127.0.0.1/services/1.1/bookpersoncreditcardaccount__service.owl#_CREDITCARDACCOUNT))
      </expr:expressionBody>
    </pddlexpr:PDDL-Expression>
  </process:hasPrecondition>
```

As can be seen, the service precondition formula holds when the given user is authorized to use the service, it has been proven that the given credit card account belongs to her and her account is valid. The arguments *_PERSON* and *_CREDITCARDACCOUNT* are concepts which belong to the service's input, and the predicates *Accepted*, *Authorized* and *accountHolder* are classes and object properties described by corresponding ontologies.

```
<process:hasResult>
  <process:Result>
    <process:hasEffect>
      <pddlexpr:PDDL-Expression>
        <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
```



```
(http://127.0.0.1/ontology/ShoppingCart.owl#ShoppingCartRequestItems
?http://127.0.0.1/services/1.1/bookpersoncreditcardaccount__service.owl#_BOOK)
```

```
</expr:expressionBody>
</pddlexpr:PDDL-Expression>
</process:hasEffect>
</process:Result>
</process:hasResult>
```

As result of the execution of this service, a book with given book title becomes part of a shopping cart of requested items. The argument *_BOOK* is also a parameter which is part of the service's input.

The XML structure used for describing preconditions and effects in SWRL has the following format:

```
<process:hasPrecondition>
  <expr:SWRL-Condition rdf:ID=" ">
    <expr:expressionLanguage rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/
Expression.owl#SWRL"/>
    <expr:expressionBody rdf:parseType="Literal">
      <swrl:AtomList>
        <rdf:first>
          Atom Description
        </rdf:first>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              Atom Description
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                ....
                ....
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
      </swrl:AtomList>
    </expr:expressionBody>
  </expr:SWRL-Condition>
</process:hasPrecondition>

<process:hasResult>
  <process:Result rdf:ID=" ">
    <process:hasEffect>
      <expr:SWRL-Expression>
        <expr:expressionLanguage rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/
Expression.owl#SWRL"/>
        <expr:expressionBody rdf:parseType="Literal">
          <swrl:AtomList>
            <rdf:first>
              Atom Description
            </rdf:first>
            <rdf:rest>
```

```

        <swrl:AtomList>
        ....
        ....
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </swrl:AtomList>
    </rdf:rest>
    </swrl:AtomList>
</expr:expressionBody>
</expr:SWRL-Expression>
</process:hasEffect>
</process:Result>
</process:hasResult>

```

As can be seen, the atoms are defined in RDF atom lists which represent conjunction of component atoms. Each atom list consists of head (*rdf:first*) and a rest (*rdf:rest*) which recursively contains the atom list. The list of atoms ends when the body is defined as empty. Atoms can be formed from unary predicates (classes), binary predicates (properties), equalities or inequalities.

Class atoms are described by the tag `<swrl:ClassAtom/>` and consist of a class description defined in an ontology and one argument either an individual name or a variable name. Property atoms are described by the tag `<swrl:IndividualPropertyAtom/>` and consist of a property name (e.g. object property) and two elements that can be individual names, variable names or data values. Equalities and inequalities are described by the tags `<swrl:DifferentIndividualsAtom/>` and `<swrl:SameIndividualAtom/>` respectively, and consists of arbitrary number of elements that can be variable names or individual names.

In the following examples are described class atom, property atom and inequality atom:

```

<swrl:ClassAtom>
    <swrl:classPredicate rdf:resource="http://127.0.0.1/ontology/core-plus-office.owl#Authorized"/>
    <swrl:argument1 rdf:resource="#_PERSON"/>
</swrl:ClassAtom>

```

```

<swrl:IndividualPropertyAtom>
    <swrl:propertyPredicate rdf:resource="http://127.0.0.1/ontology/Mid-level-ontology.owl#accountHolder"/>
    <swrl:argument1 rdf:resource="#_PERSON"/>
    <swrl:argument2 rdf:resource="#_CREDITCARDACCOUNT"/>
</swrl:IndividualPropertyAtom>

```

```

<swrl:DifferentIndividualsAtom>
    <swrl:argument1 rdf:resource="#_GEOGRAPHICAL-REGION1"/>
    <swrl:argument2 rdf:resource="#_GEOGRAPHICAL-REGION2"/>
</swrl:DifferentIndividualsAtom>

```

Requests and Relevance sets

Overview

All relevance sets created for OWLS-TC4 are available in XML format in **owls-tc4.xml** provided in the main directory of the distribution. These were created using the SWSRAT (**S**emantic **W**eb **S**ervice **R**elevance **A**ssessment **T**ool) tool which was developed by Steffen Metzger in collaboration with Patrick Kapahnke and Matthias Klusch at the [R&D division I2S](#) of the German Research Center for Artificial Intelligence (DFKI Saarbruecken, Germany). It contains binary as well as graded relevance sets for all 39 queries. Different users of the SWSRAT tool subjectively assessed the service request/offer pairs. Not all possible combinations have been rated, thus the relevance sets can not be considered as *complete*. A pooling strategy as used in TREC⁷ based on the top-100 results of participants of the S3 contest⁸ in 2008 has been conducted.

Graded Relevance

The graded relevance sets have been created using the following 4-graded scale:

- **highly relevant** (value: 3) - Any service offer that is exactly what the user asked for (or even better for him, e.g. by giving additional information)
- **relevant** (value: 2) - Any service offer that *might* answer the request completely or does the requested job at least partially
- **potentially relevant** (value: 1) - Any service offer that *may* be helpful.
- **nonrelevant** (value: 0) - Anything totally irrelevant to the service request

Binary Relevance

The collaboratively created graded relevance assessments were projected onto a binary relevance scale using SWSRAT. The approach of *relaxed* binary relevance is chosen for this, which means that a service offer is considered as binary relevant to a query, if it is at least potentially relevant according to the graded scale given above.

⁷ Text Retrieval Conference, <http://trec.nist.gov/>

⁸ Semantic Service Selection, <http://www-ags.dfki.uni-sb.de/~klusch/s3/>

Structure of the XML Relevance File

```
<binaryrelevanceset>    //for the binary relevance sets
  <request ...>
    <name ... />
    <uri .../>
    <ratings>
      <offer ...>    //each request contains multiple offers
        <name ... />
        <uri .../>
        <relevant>value</relevant>    //here the value can be either 0 or 1
      </offer>
      ...
    </ratings>
  </request>
  ...
</binaryrelevanceset>

<relevancegrades>    //for the graded relevance assessments
  <services>
    <request ...>
      <name ... />
      <uri .../>
      <ratings>
        <offer ...>
          <name ... />
          <uri .../>
          <grade ...>
            <name>NameOfGrade</name>    //highly relevant (value 3); relevant (value 2);
            <value>ValueOfGrade</value>    //potentially relevant (value 1); nonrelevant (value 0)
          </grade>
        </offer>
        ...
      </ratings>
    </request>
    ...
  </services>
</relevancegrades>
```

Used Ontologies

To describe the semantics of I/O concepts of services in this test collection, different ontologies have been used. They were retrieved from various public sources in the Web for non-commercial use, and are included in the collection for your convenience. Some of them are listed in the following.

1. the "**Suggested Upper Merged Ontology (SUMO)**"

upon which several other ontologies rely.

Project: <http://reliant.teknowledge.com/>

Ontology: <http://reliant.teknowledge.com/DAML/>

2. **Larflast** ontology (University of Sofia)

Project: <http://www-it.fmi.uni-sofia.bg/larflast/>

Ontology: <http://www.larflast.bas.bg/>

3. **Wine and food ontology** (W3C)

Project: <http://www.w3.org/>

Ontology: <http://www.w3.org/TR/2003/PR-owl-guide-20031215/>

4. The **AKT Reference Ontology** v2 (AKT Technologies)

Project: <http://www.aktors.org/>

Ontology: <http://d3e.open.ac.uk/akt/2002/ref-onto.html>
<http://www.aktors.org/ontology/>

5. **University ontology** by Zhengxiang Pan

Dept. of Computer Science and Engineering, Leigh University

Project: <http://www.lehigh.edu/~zhp2/>

Ontology: <http://www.lehigh.edu/~zhp2/2004/0401/>

6. Taken from the Protege demo owl ontology site: **travel.owl** A tutorial ontology for a Semantic Web of tourism. Contributed by Holger Knoblauch.

Project: <http://protege.stanford.edu>

Ontology: <http://protege.stanford.edu/plugins/owl/owl-library/>
<http://www.owl-ontologies.com/>

7. Taken from German Research Center for Artificial Intelligence website

Project: <http://www.dfki.de/scallops/>

Ontology: <http://www.dfki.de/scallops/health-scallops/>

8. Simple **Book ontology** (DFKI): **book.owl**
9. **Simplified SUMO ontology** (DFKI): **simplified_sumo.owl**
Extract from ontology 1 to allow faster parsing
10. My Ontology (DFKI): anonymous ontology
11. Geography domain ontology: **geographydataset.owl**,
PROTON ontologies: **protont.owl**, **protonu.owl** and **protons.owl**
Project: <http://fusion.cs.uni-jena.de/professur/jgd/>
Ontology: <http://fusion.cs.uni-jena.de/professur/jgd/>
12. Ontology **ontosem.owl**
University of Maryland
Ontology: <http://morpheus.cs.umbc.edu/aks1/>
13. Ontology **core-plus-office.owl**
Project: IRIS Semantic Desktop, <http://www.openiris.org/>
Ontology: <http://www.openiris.org/downloads/IRIS-nightly/doc-current/data/inProgress/>
14. PDDL Expression ontology (DFKI): **PDDLExpression.owl**
15. Simulation ontologies (DFKI): **messemodul.owl** and **spatial_ontology.owl**

Disclaimer

Copyright (C) 2010, DFKI

OWLS-TC is free for non-commercial use only. You can redistribute it and/or modify it under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation; version 2 of the License.

DISCLAIMER OF WARRANTY

This test collection is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of fitness for a particular purpose. See the GNU General Public License for more details. For a copy of the GNU General Public License, please write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Support and contact

Please report any bugs and errors you encounter to us; we are continuously working to improve OWLS-TC! Errors can be reported by emailing the developers or preferably by using the online tracker system available at the project homepage at [semwebcentral.org: http://projects.semwebcentral.org/tracker/?group_id=89](http://projects.semwebcentral.org/tracker/?group_id=89)

For OWLS-TC support and help, please contact

- Patrick Kapahnke at patrick.kapahnke@dfki.de
- Martin Vasileski at martin.vasileski@dfki.de

For general and scientific questions, please contact:

- PD Dr. Matthias Klusch at klusch@dfki.de, <http://www.dfki.de/~klusch>

Release history

- 20.09.2010 Fourth release, OWLS-TC version 4
- 10.11.2009 OWLS-TC version 3, revision 1
- 24.06.2009 Third release, OWLS-TC version 3
- 23.05.2008 OWLS-TC version 2.2, revision 2
- 17.12.2007 OWLS-TC version 2.2, revision 1
- 9.11.2007 OWLS-TC version 2.2
- 6.10.2006 OWLS-TC version 2.1
- 15.11.2005 Second release, OWLS-TC version 2
- 6.04.2005 Initial release, OWLS-TC version 1