OWL 2 Web Ontology Language: Conformance and Test Cases

W3C Working Draft 02 December 2008

This version:
http://www.w3.org/TR/2008/WD-owl2-test-20081202/
Latest version:
http://www.w3.org/TR/owl2-test/
Previous version:
http://www.w3.org/TR/2008/WD-owl2-test-20081008/
Editors:
Michael Smith, Clark & Parsia
Ian Horrocks, Oxford University
Markus Krötzsch, FZI Karlsruhe
Contributors:
Sandro Hawke, W3C/MIT
Bijan Parsia, University of Manchester

This document is also available in these non-normative formats: PDF version.

Abstract

OWL 2 extends the W3C OWL Web Ontology Language with a small but useful set of features that have been requested by users, for which effective reasoning algorithms are now available, and that OWL tool developers are willing to support. The new features include extra syntactic sugar, additional property and qualified cardinality constructors, extended datatype support, simple metamodeling, and extended annotations.
This document describes the conditions that OWL 2 tools must satisfy in order to be conformant with the language specification. It also presents a common format for OWL 2 test cases that both illustrate the features of the language and can be used for testing conformance.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Set of Documents

This document is being published as one of a set of 11 documents:

1. Structural Specification and Functional-Style Syntax
2. Direct Semantics
3. RDF-Based Semantics
4. Conformance and Test Cases (this document)
5. Mapping to RDF Graphs
6. XML Serialization
7. Profiles
8. Quick Reference Guide
9. New Features and Rationale
10. Manchester Syntax
11. rdf:text: A Datatype for Internationalized Text

Last Call

The Working Group believes it has completed its design work for the technologies specified this document, so this is a "Last Call" draft. The design is not expected to change significantly, going forward, and now is the key time for external review, before the implementation phase.

Summary of Changes

Since the version of 08 October 2008, this document has been updated in the following ways:

- The introduction has been expanded.
- It has been made clear (in the introduction and elsewhere) that the document does not present actual test cases, but rather a format for test cases and pointers to test case repositories.
- More precise pointers are provided to the syntactic restrictions referred to in the definitions of syntactic conformance.
- More is said about the syntactic conformance conditions for non-RDF syntaxes.
- Conditions for Datatype Map Conformance have been clarified.
- Some introductory text on Document Conformance has been added.

- The test case format has changed to include additional test metadata.
- The format for imported ontologies in test cases has been modified to include syntax and location data.
- The test case format is described in much greater detail.

Please Comment By 23 January 2009

The OWL Working Group seeks public feedback on these Working Drafts. Please send your comments to public-owl-comments@w3.org (public archive). If possible, please offer specific changes to the text that would address your concern. You may also wish to check the Wiki Version of this document for internal-review comments and changes being drafted which may address your concerns.

No Endorsement

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

Patents

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

[Show Short TOC]

Contents

1 Introduction

This document describes conformance conditions for OWL 2, and introduces the format of OWL 2 test cases that are provided as part of the OWL 2 Test Case Repository [*OWL 2 Test Cases*].

Conformance conditions are described for both OWL 2 documents and for tools that process such documents. In particular, the conformance conditions for an OWL 2 entailment checker are described in some detail.

A categorization and common format of test cases is presented. The purpose of test cases is to illustrate various features and to help in testing conformance. The provided set of test cases is not exhaustive, however, nor does it constitute a conformance test suite for OWL: passing all the tests is no guarantee that a given system conforms to the OWL 2 specification. The presented format is intended to facilitate the use of tests by OWL system developers, e.g., in a test harness, as well as the extension of the test suite with new tests.

This document does not contain actual test cases. Test cases that have been approved by the Working Group can be found in the OWL 2 Test Case Repository [*OWL 2 Test Cases*], and a public test case repository [*Contributed Test Cases*] is provided as a platform for collecting further test cases even after the termination of the Working Group.

The italicized keywords *must*, *must not*, *should*, *should not*, and *may* are used to specify normative features of OWL 2 documents and tools, and are interpreted as specified in RFC 2119 [*RFC 2119*].

2 Conformance (Normative)

This section describes conformance conditions for OWL 2 documents and tools. In particular, it describes the syntactic conditions that characterize OWL 2 ontology documents, including those that conform to the various OWL 2 profiles [*OWL 2 Profiles*], and the syntactic and semantic conditions that *must* be satisfied by conformant OWL 2 tools.

2.1 Document Conformance

Several syntaxes have been defined for OWL 2 ontology documents, some or all of which could be used by OWL 2 tools for exchanging documents. However, conformant OWL 2 tools that take ontology documents as input(s) *must* accept ontology documents using the RDF/XML serialization [*OWL 2 Mapping to RDF Graphs*], and conformant OWL 2 tools that publish ontology documents *must*, if possible, be able to publish them in the RDF/XML serialization if asked to do so (e.g., via HTTP content negotiation). OWL 2 tools *may* also accept and/or publish ontology documents using other serializations, for example the XML Serialization [*OWL 2 XML Syntax*].

2.1.1 Syntactic Conformance

Any RDF document [*RDF Syntax*] is an **OWL 2 Full ontology document**.

An OWL 2 Full ontology document is an **OWL 2 DL ontology document** iff it can be successfully parsed using the procedure for mapping from RDF graphs to the structural specification described in the OWL 2 Mapping to RDF Graphs [*OWL 2 Mapping to RDF Graphs*].

An OWL 2 DL ontology document is an **OWL 2 EL ontology document** iff the result of the procedure for mapping from RDF graphs to the structural specification described in the OWL 2 Mapping to RDF Graphs [*OWL 2 Mapping to RDF Graphs*] is an OWL 2 EL ontology as defined in the OWL 2 Profiles specification [*OWL 2 Profiles*].

An OWL 2 DL ontology document is an **OWL 2 QL ontology document** iff the result of the procedure for mapping from RDF graphs to the structural specification described in the OWL 2 Mapping to RDF Graphs [*OWL 2 Mapping to RDF Graphs*] is an OWL 2 QL ontology as defined in the OWL 2 Profiles specification [*OWL 2 Profiles*].

An OWL 2 Full ontology document is an **OWL 2 RL ontology document** iff the result of the procedure for mapping from RDF graphs to the structural specification described in the OWL 2 Mapping to RDF Graphs [*OWL 2 Mapping to RDF Graphs*] is an OWL 2 RL ontology as defined in the OWL 2 Profiles specification [*OWL 2 Profiles*].

Conformance for other serializations is a direct consequence of the relevant serialization specification, the OWL 2 Syntax specification [*OWL 2 Specification*] (in particular, the definition of the canonical parsing process in the OWL 2 Syntax specification [*OWL 2 Specification*]), and the OWL 2 Profiles specification [*OWL 2 Profiles*].

Example:

An XML document is an OWL 2 DL ontology document iff it validates against the OWL 2 XML Schema [*OWL 2 XML Syntax*], and it can be successfully parsed using the canonical parsing process as defined in the OWL 2 Syntax specification [*OWL 2 Specification*]. It is an OWL 2 EL (respectively QL, RL) ontology document iff the result is also an OWL 2 EL (respectively QL, RL) ontology as defined in the OWL 2 Profiles specification [*OWL 2 Profiles*].

2.1.2 Datatype Map Conformance

In OWL 2, semantic conditions are defined with respect to a datatype map. This *must* be either the OWL 2 Datatype map (as defined in Section 4 of the OWL 2 Syntax specification [*OWL 2 Specification*]), or an extension of the OWL 2 Datatype map to include additional datatypes.

Note that OWL 2 Profiles may support only a reduced set of datatypes. This is, however, a syntactic condition that must be met by documents in order to fall within the relevant profile, and the semantic conditions on the supported datatypes are unchanged, i.e., they are defined by a (possibly extended) OWL 2 Datatype map.

2.2 Document Checker Conformance

An OWL 2 document checker is a tool that takes one or more OWL 2 ontology documents and checks some (syntactic or semantic) condition. Semantic conditions are defined with respect to abstract structures obtained from the ontology documents, typically via a parsing process. In the case of an OWL Full ontology document, the abstract structure is an RDF graph; in all other cases it is an OWL 2 ontology as defined in the OWL 2 Syntax specification [*OWL 2 Specification*]. Given an ontology document *d*, we will denote with *Ont(d)* the abstract structure obtained from *d*.

Example:

Given an OWL 2 DL ontology document *d* in the RDF/XML serialisation, *Ont(d)* is the ontology obtained from *d* via the procedure for mapping from RDF graphs to the structural specification described in the OWL 2 Mapping to RDF Graphs [*OWL 2 Mapping to RDF Graphs*]. Similarly, given an OWL 2 Full ontology document in RDF/XML syntax, *Ont(d)* is the RDF graph corresponding to *d*, as defined in [*RDF Syntax*].

As noted above, any conformant OWL 2 tool *must* accept ontology documents using the RDF/XML serialisation [*OWL 2 Mapping to RDF Graphs*]; it *may* also accept ontology documents using other serializations, for example the XML Serialization [*OWL 2 XML Syntax*].

The conformance conditions related to entailment checking and query answering are defined below. Other OWL 2 tools *must* satisfy similar conditions. In particular, they *must* be consistent with the Direct Semantics [*OWL 2 Direct Semantics*] and/ or the RDF-Based Semantics [*OWL 2 RDF-Based Semantics*].

2.2.1 Entailment Checker

An OWL 2 entailment checker takes as input two OWL 2 Full, DL, EL, QL, or RL ontology documents d_1 and d_2 and checks if *Ont*(d_1) entails *Ont*(d_2) with respect to its datatype map and either the Direct Semantics [*OWL 2 Direct Semantics*] or the RDF-Based Semantics [*OWL 2 RDF-Based Semantics*]. When using the Direct Semantics, *Ont*(d_1) and *Ont*(d_2) must satisfy the restrictions described in Section 2.5 of the Direct Semantics [*OWL 2 Direct Semantics*]. Additionally, an OWL 2 entailment checker:

- *must* provide a means to determine the datatypes supported by its datatype map, and any limits it has on datatype literals and datatype values [*OWL 2 Specification*] — for example, by listing them in its supporting documentation; and
- *must* provide a means to determine the semantics it uses (either the Direct Semantics [*OWL 2 Direct Semantics*] or the RDF-Based Semantics [*OWL 2 RDF-Based Semantics*]) — for example, by specifying in its supporting documentation which of the two semantics it uses.

An OWL 2 entailment checker returns a single result, being `True`, `False`, `Unknown` or `Error`. `True` indicates that the relevant entailment holds; `False` indicates that the relevant entailment does not hold; `Unknown` indicates that the algorithm used by the checker is not able to determine if the entailment holds; `Error` indicates that the checker encountered an error condition such as receiving an invalid input or exceeding resource limits. While sometimes needed (for example, for pragmatic reasons), `Unknown` and `Error` are not desired responses for valid inputs.

Additionally, an OWL 2 entailment checker:

- *must* return `Error` if the parsing process fails (for example, due to network errors);
- *must* return `Error` if an input document uses datatypes that are not supported by its datatype map or literals that it does not support (for example, very large integers)—see Section 4 of the OWL 2 Syntax specification [*OWL 2 Specification*]; and
- *must* return `Error` if the computation fails, for example as a result of exceeding resource limits.

Five different conformance classes of OWL 2 entailment checkers are defined:

An OWL 2 Full entailment checker is an OWL 2 entailment checker that takes RDF documents as input. It *must* return `True` only when *Ont(d_1)* entails *Ont(d_2)* with respect to the RDF-Based Semantics [*OWL 2 RDF-Based Semantics*], and it *must* return `False` only when *Ont(d_1)* does not entail *Ont(d_2)* with respect to the RDF-Based Semantics. It *should not* return `Unknown`.

An OWL 2 DL entailment checker is an OWL 2 entailment checker that takes OWL 2 DL ontology documents as input. It *must* return `True` only when *Ont(d_1)* entails *Ont(d_2)* with respect to the Direct Semantics [*OWL 2 Direct Semantics*], and it *must* return `False` only when *Ont(d_1)* does not entail *Ont(d_2)* with respect to the Direct Semantics. It *should not* return `Unknown`.

An OWL 2 EL entailment checker is an OWL 2 entailment checker that takes OWL 2 EL ontology documents as input. It *must* return `True` only when *Ont(d_1)* entails *Ont(d_2)* with respect to the Direct Semantics [*OWL 2 Direct Semantics*], and it *must* return `False` only when *Ont(d_1)* does not entail *Ont(d_2)* with respect to the Direct Semantics. It *should not* return `Unknown`.

An OWL 2 QL entailment checker is an OWL 2 entailment checker that takes OWL 2 QL ontology documents as input. It *must* return `True` only when *Ont(d_1)* entails *Ont(d_2)* with respect to the Direct Semantics [*OWL 2 Direct Semantics*], and it *must* return `False` only when *Ont(d_1)* does not entail *Ont(d_2)* with respect to the Direct Semantics. It *should not* return `Unknown`.

An OWL 2 RL entailment checker is an OWL 2 entailment checker that takes RDF documents as input. It *must* return `True` only when *Ont(d_1)* entails *Ont(d_2)* with respect to the RDF-Based Semantics [*OWL 2 RDF-Based Semantics*], and it *must* return `False` only when *Ont(d_1)* does not entail *Ont(d_2)* with respect to the RDF-Based Semantics; it *may* return `Unknown` if *FO(Ont(d_1))* ∪ *R* does not entail *FO(Ont(d_2))* under the standard first-order semantics, where *R* denotes the OWL 2 RL/RDF rules [*OWL 2 Profiles*], and *FO(Ont(d_i))* denotes the FO theory corresponding to *Ont(d_i)* in which triples are represented using the `T` predicate — that is, `T(s, p, o)` represents an RDF triple with the subject `s`, predicate `p`, and the object `o`.

An OWL 2 entailment checker is **terminating** if, given sufficient resources (memory, addressing space, etc.), it will always return `True`, `False`, or `Unknown` in a finite amount of time (i.e., CPU cycles) on syntactically-valid inputs.

An OWL 2 entailment checker is **complete** if, given sufficient resources, it will always return `True` or `False` on syntactically-valid inputs.

Note: Every OWL 2 Full entailment checker is also an OWL 2 RL entailment checker.

Note: Every OWL 2 DL entailment checker is also an OWL 2 EL and OWL 2 QL entailment checker.

Note: From Theorem PR1 of Profiles [*OWL 2 Profiles*], it follows that an OWL 2 RL entailment checker can return `False` if *Ont(d_1)* and *Ont(d_2)* satisfy the constraints described in Theorem PR1, and *FO(d_1)* ∪ *R* does not entail *FO(d_2)* under the standard first-order semantics, where *R* denotes the OWL 2 RL/RDF rules [*OWL 2 Profiles*], and *FO(d_i)* denotes the FO theory corresponding to *Ont(d_i)* in which triples are represented using the `T` predicate — that is, `T(s, p, o)` represents an RDF triple with the subject `s`, predicate `p`, and the object `o`. Implementations not wishing to check whether *Ont(d_1)* and *Ont(d_2)* satisfy the relevant constraints can simply return `Unknown` whenever they are not able to return `True`.

2.2.2 Query Answering

Query answering is closely related to entailment checking. A query can be thought of as an ontology *Q* in which some of the terms have been replaced by variables $x_1, ..., x_n$. Given an ontology *O*, a tuple $t = <t_1, ..., t_n>$ is **an** answer for *Q* with respect to *O* if *O* entails $Q_{[x/t]}$, where $Q_{[x/t]}$ is derived from *Q* by substituting the variables $x_1, ..., x_n$ with $t_1, ..., t_n$; **the** answer to *Q* with respect to *O* is the set of all such tuples.

Although highly inefficient in practice, query answering could be performed simply by iterating through all possible n-tuples formed from terms occurring in *O* and checking the corresponding entailment using an OWL 2 entailment checker. The properties of OWL 2 entailment checkers mean that the resulting answer will always be **sound**, i.e., every tuple occurring in the answer set is an answer to the query. If any one of the entailment checks might return `Unknown`, then the answer to the query may be **incomplete**, i.e., there may exist a tuple *t* that is an answer to the query but that does not occur in the answer set; implementations *should* issue a warning in this case.

The properties of OWL 2 Full, DL, EL and QL entailment checkers mean that query answering *should* be both sound and complete. In the case of OWL RL, query answering *should* be sound, and *should* also be complete if both the ontology and the query satisfy the constraints described in Theorem PR1.

3 Test Cases

This section introduces various types of test cases. Each test case describes certain inputs that can be provided to OWL 2 tools and specifies the behavior required to satisfy the conformance conditions described above, given the inputs. Test cases adhere to a common format that simplifies automatic processing, e.g. in a test harness, which is detailed below.

Concrete sets of test cases can be found in various repositories as described below. They are divided into a fixed set of test cases that have been approved based on a process defined later in this section, and an open set of user-contributed test cases that can be collected via a dedicated web site.

3.1 Test Types

There are several distinguished types of test cases detailed in the following sub-sections. The type of a test determines the task and expected outcome of the test. The type thus also affects the data associated to a test case, e.g., since only certain kinds tests require the specification of an entailed ontology.

While all test cases have some primary purpose specified by their type, it is often possible to use the provided data for other tests as well. For example, the inputs of any negative entailment test can also be used in a consistency test. Such re-interpretations of test cases can generally be useful, depending on the tool being validated and the goal of validation. For this reason, a concrete test case may have more than one type and thus allow multiple uses.

3.1.1 Syntactic Tests

Syntactic tests can be applied to OWL tools that process OWL ontology documents, or that transform between various syntactic forms of OWL. These modes of operation are not covered by any conformance requirement, but syntactic tests may still be useful in tool development.

3.1.1.1 Profile Identification Tests

Profile identification tests validate a tool's recognition of Syntactic Conformance. These tests require at least one input ontology document. Each test describes the conformance of *all* provided input ontology documents relative to structural and syntactic restrictions that are specified by the test case.

Since all test cases usually specify the profiles (and species) of the input ontology documents, essentially all test cases can be used as profile identification tests.

3.1.1.2 Syntax Translation Tests

Syntax translation tests validate the translation of OWL ontology documents from one syntax to another, using the definition of structural equivalence defined in [*OWL 2 Specification*]. Each test case of this type specifies input ontology documents in multiple syntactic forms which describe structurally equivalent ontologies. Tools that parse and serialize ontology documents may use this data to verify their correct operation. Note that tests of this kind do not prescribe a particular syntactic form to be the outcome of a syntactic translation: Different serializations are correct as long as they describe the same ontological structure.

Tests of this type specify multiple input ontology documents, and indicate which of the provided syntactic forms are normative for the translation test.

3.1.2 Semantic Tests

Semantic tests specifically address the functionality of OWL entailment checkers. Each test case of this type specifies necessary requirements that *must* be satisfied by any entailment checker that meets the according conformance conditions.

Each semantic test case also specifies whether it is applicable to the [*OWL 2 Direct Semantics*], to the [*OWL 2 RDF-Based Semantics*], or to both. A test is only relevant for testing conformance of tools that use a semantics to which the test applies.

Semantic tests specify one or more OWL 2 ontology documents and check semantic conditions defined with respect to abstract structures obtained from the ontology documents, typically via a parsing process. In the case of an OWL Full ontology document, the abstract structure is an RDF graph; in all other cases it is an OWL 2 ontology as defined in the OWL 2 Syntax specification [*OWL 2 Specification*]. We will denote with *Ont(d)* the abstract structure obtained from the ontology document *d*.

3.1.2.1 Entailment Tests

Entailment tests (or positive entailment tests) specify two ontology documents: a premise ontology document d_1 and a conclusion ontology document d_2 where *Ont*(d_1) entails *Ont*(d_2) with respect to the specified semantics. If provided with inputs d_1 and d_2 (and, if applicable, with access to any imported ontologies), a conforming entailment checker *should* return True, it *should not* return Unknown, and it *must not* return False.

In all entailment tests, the ontologies *Ont*(d_1) and *Ont*(d_2) are consistent. Therefore, all entailment tests are also consistency tests.

Editor's Note: The current format disallows tests validating that inconsistent ontologies must entail everything. This could be changed.

3.1.2.2 Non-Entailment Tests

Non-Entailment tests (or negative entailment tests) specify two ontology documents: a premise ontology document *Ont*(d_1) and a non-conclusion ontology *Ont*(d_2) where *Ont*(d_1) does not entail *Ont*(d_2) with respect to the specified semantics. If provided with inputs d_1 and d_2 (and, if applicable, with access to any imported ontologies), a conforming entailment checker *should* return False, it *should not* return Unknown, and it *must not* return True.

In all non-entailment tests, the ontologies *Ont*(d_1) and *Ont*(d_2) are consistent. Therefore, all non-entailment tests are also consistency tests.

3.1.2.3 Consistency Tests

Consistency tests validate a tool's recognition of consistency, as defined in the [*OWL 2 Direct Semantics*] and the [*OWL 2 RDF-Based Semantics*]. These tests specify an input ontology document, the premise ontology document *d*, where *Ont(d)* is consistent with respect to the specified semantics.

Entailment checkers that directly support consistency checking *should* determine *Ont(d)* to be consistent, and *must not* determine *Ont(d)* to be inconsistent. Entailment checkers that do not support this operation may execute consistency tests like non-entailment test: if the ontology *Ont(d)* is consistent, then *Ont(d)* does not entail the inconsistent ontology O_{in} (see Appendix). Given inputs *d* and d_{in}, a conforming entailment checker *should* thus return False, it *should not* return Unknown, and it *must not* return True.

3.1.2.4 Inconsistency Tests

Inconsistency tests validate a tool's recognition of consistency, as defined in the [*OWL 2 Direct Semantics*] and the [*OWL 2 RDF-Based Semantics*]. These tests specify an input ontology document, the premise ontology document *d*, where *Ont(d)* is inconsistent with respect to the specified semantics.

Entailment checkers that directly support inconsistency checking *should* determine *Ont(d)* to be inconsistent, and *must not* determine *Ont(d)* to be consistent. Entailment checkers that do not support this operation may execute inconsistency tests like entailment test: if the ontology *Ont(d)* is inconsistent, then *Ont(d)* entails the inconsistent ontology O_{in} (see Appendix). Given inputs *d* and d_{in}, a conforming entailment checker *should* thus return True, it *should not* return Unknown, and it *must not* return False.

3.2 Test Case Format

Test cases are described using OWL, based on a test case ontology documented in this section. The given test case format is mainly based upon two design choices. Firstly, each test case in OWL 2 can be completely represented within a single file, and the location of this file is not relevant. In this way, all test cases adhering to this format are completely portable, and can be published and distributed freely.

A second design choice was to allow individual test case documents to be processed with OWL tools, no matter whether these tools support OWL DL or OWL Full. Thus the presented test case ontology and all test case documents using it syntactically conform to OWL DL. This design choice also motivates the use of dedicated URIs (based on the namespace prefix http://www.w3.org/2007/OWL/testOntology#) for all elements of the test case ontology: Existing test ontologies, such as the ones used by the WebOnt working group [*OWL Test Cases*], have been crafted for OWL Full and do not meet all OWL DL conformance requirements.

Overall, the given design is intended to ensure maximal compatibility and ease of use in a variety of different tools. This section describes various elements of the test ontology grouped according to their purpose, and it includes axioms using the Functional Syntax. The complete test ontology is summarized in the last section. This ontology uses OWL as a tool for conceptual modeling, describing the intended structure of test case documents – it is, however, not necessary to compute entailments of this ontology in order to use the provided test case documents. Details on the changes of the test case format as compared to WebOnt are found in Section 3.5.

3.2.1 Input Ontologies

The *:inputOntology* data property associates a test with one or more input ontologies. Values of this property (and thus of all of its subproperties) are of type *xsd:string*. Subproperties are used to differentiate among multiple input ontologies that are provided for different purposes depending on the type of test:

```
Declaration( DataProperty( :inputOntology ) )
PropertyRange( :inputOntology xsd:string )

Declaration( DataProperty( :premiseOntology ) )
Declaration( DataProperty( :conclusionOntology ) )
Declaration( DataProperty( :nonConclusionOntology ) )

SubPropertyOf( :premiseOntology :inputOntology )
SubPropertyOf( :conclusionOntology :inputOntology )
SubPropertyOf( :nonConclusionOntology :inputOntology )
```

Similarly, further subproperties of *:inputOntology* are used to indicate the syntax of the input ontology:

```
Declaration( DataProperty( :fsInputOntology ) )
Declaration( DataProperty( :owlXmlInputOntology ) )
Declaration( DataProperty( :rdfXmlInputOntology ) )

SubPropertyOf( :fsInputOntology :inputOntology )
SubPropertyOf( :owlXmlInputOntology :inputOntology )
SubPropertyOf( :rdfXmlInputOntology :inputOntology )
```

To fully specify the purpose *and* syntax of a given input ontology, the test case ontology specifies "intersection properties" that combine (i.e. are subproperties of) two of the above properties. An example is the property *:rdfXmlPremiseOntology*, used to denote a premise ontology in RDF/XML syntax. These more specific properties are used in many test cases, the only exception being pure syntactic tests where the purpose of the given input ontologies does not need to be specified.

3.2.2 Type

All test cases are individuals in the *:TestCase* class. Subclasses of this class are used to map tests to the test types described above. The axioms below describe the relationships between the test types and the input ontology requirements of each test type.

```
Declaration( Class( :TestCase ) )
Declaration( Class( :ProfileIdentificationTest ) )
Declaration( Class( :ConsistencyTest ) )
Declaration( Class( :InconsistencyTest ) )
Declaration( Class( :PositiveEntailmentTest ) )
Declaration( Class( :NegativeEntailmentTest ) )

SubClassOf( :ProfileIdentificationTest :TestCase )
SubClassOf( :ConsistencyTest :ProfileIdentificationTest )
SubClassOf( :InconsistencyTest :ProfileIdentificationTest )
SubClassOf( :PositiveEntailmentTest :ConsistencyTest )
SubClassOf( :NegativeEntailmentTest :ConsistencyTest )

SubClassOf( :ConsistencyTest MinCardinality( 1 :premiseOntology ) )
SubClassOf( :InconsistencyTest MinCardinality( 1 :premiseOntology ) )
SubClassOf( :PositiveEntailmentTest MinCardinality( 1 :conclusionOntolo
SubClassOf( :NegativeEntailmentTest MinCardinality( 1 :nonConclusionOnt

DisjointClasses( :ConsistencyTest :InconsistencyTest )
```

Note that the cardinatlity restrictions only specify minimal cardinalities. In practice, semantic tests will indeed have only one premise, conclusion, or non-conclusion, but for convenience each of those may be provided in multiple syntactic forms. This is the reason why the above assertions do not require exact cardinalities.

3.2.3 Normative Syntax

The *:normativeSyntax* object property associates a test case with individuals indicating one or more syntactic forms which are normative for all input ontologies associated with this test case. For convenience, test cases may still provide redundant input ontologies using additional syntactic forms which are not normative. Most types of tests usually provide exactly one normative form, whereas syntax translation tests necessarily provide multiple normative syntactic forms.

The property *:normativeSyntax* may take one of the following mutually different individuals as values:

- The individual *:FUNCTIONAL* indicates that all functional syntax input ontologies are normative.
- The individual *:OWLXML* indicates that all OWL XML syntax input ontologies are normative.
- The individual *:RDFXML* indicates that all RDF/XML syntax input ontologies are normative.

```
Declaration( ObjectProperty( :normativeSyntax ) )
PropertyRange( :normativeSyntax OneOf( :RDFXML :FUNCTIONAL :OWLXML ) )
DifferentIndividuals( :RDFXML :FUNCTIONAL :OWLXML )
```

```
SubClassOf( :TestCase MinCardinality( 1 :normativeSyntax ) )
```

3.2.4 Applicable Semantics

The *:semantics* object property indicates to which kind of OWL 2 semantics a semantic test case is applicable. The property can take the following mutually distinct individuals as possible values:

- The individual *:FULL* indicates that the test is applicable if the [OWL 2 RDF-Based Semantics] are used.
- The individual *:DL* indicates that the test is applicable if the [OWL 2 Direct Semantics] are used.

Each test should have one property assertion for each of the possible semantics: either a positive property assertion to confirm that the tests is applicable under this semantics, or a negative property assertion indicating it is not.

```
Declaration( ObjectProperty( :semantics ) )
DifferentIndividuals( :DL :FULL )
PropertyRange( :semantics OneOf( :DL :FULL ) )
```

If a test case is not applicable under one of the two semantics, then it is required that another test case is provided to highlight and illustrate the semantic difference (e.g. an entailment in OWL 2 Full might sometimes be a non-entailment in OWL 2 DL). The symmetric property *:alternativeSemanticsTest* is used to associate two test cases that are complementary in this sense.

```
Declaration( ObjectProperty( :alternativeSemanticsTest ) )
FunctionalProperty( :alternativeSemanticsTest )
SymmetricProperty( :alternativeSemanticsTest )

SubClassOf(
  IntersectionOf( :TestCase ComplementOf( HasValue( :semantics :FULL )
  SomeValuesFrom( :alternativeSemanticsTest HasValue( :semantics :FULL
)

SubClassOf(
  IntersectionOf( :TestCase ComplementOf( HasValue( :semantics :DL ) )
  SomeValuesFrom( :alternativeSemanticsTest HasValue( :semantics :DL )
)
```

3.2.5 Species

The *:species* property describes the syntactic conformance of the input ontology with respect to OWL 2 Full and OWL 2 DL. The property may take either of the following two mutually distinct individuals as values:

- The individual *:FULL* indicates that all input ontologies are OWL 2 Full ontology documents. This should be the case for all tests.
- The individual *:DL* indicates that all input ontologies are OWL 2 DL ontology documents.

Each test should either have a property assertion indicating the input ontology is an OWL DL ontology, or a negative property assertion indicating that it is not.

```
Declaration( ObjectProperty( :species ) )
PropertyRange( :species OneOf( :DL :FULL ) )
SubClassOf( :TestCase HasValue( :species :FULL ) )
```

If an input ontology is not an OWL 2 DL ontology document, the normative syntax must be RDF/XML:

```
SubClassOf(
    IntersectionOf( :TestCase ComplementOf( HasValue( :species :DL ) ) )
    HasValue( :normativeSyntax :RDFXML )
)
```

3.2.6 Profiles

The *:profile* object property describes the syntactic conformance of the input ontology with respect to the profiles of OWL 2. It may take one of the following mutually different individuals as values:

- The individual *:EL* indicates that all input ontologies are OWL 2 EL ontology documents.
- The individual *:QL* indicates that all input ontologies are OWL 2 QL ontology documents.
- The individual *:RL* indicates that all input ontologies are OWL 2 RL ontology documents.

Each test should have one property assertion for each of the profiles: either a positive property assertion to confirm that the tests conforms to the restrictions of the profile, or a negative property assertion indicating it does not.

```
Declaration( ObjectProperty( :profile ) )
PropertyRange( :profile OneOf( :EL :QL :RL ) )
```

```
        DifferentIndividuals( :EL :QL :RL )
```

The following axiom reflects the fact that OWL 2 EL and OWL 2 QL are syntactic profiles of OWL 2 DL.

```
        SubClassOf(
            SomeValuesFrom( :profile OneOf( :EL  :QL ) )
            HasValue( :species :DL )
        )
```

3.2.7 Imported Ontologies

The *:importedOntology* property associates a test case with an individual that describes an auxiliary ontology which is required to resolve import directives, as explained in Section 3.4 of [*OWL 2 Specification*]. The fact that import directives refer to *ontology locations* conflicts with the goal of maintaining test cases in single, location-independent files. Indeed, all contributed test cases would have to ensure imported ontologies to be available in the specified locations, and web access would be required to execute such tests.

Thus imported ontologies will be made available under the according URLs for all approved test cases (see Test Case Repositories). For contributed test cases, this may not be guaranteed, and it is generally desirable to execute all tests off-line based on a single manifest file. Test cases therefore also provide copies of the contents of all imported ontologies as part of their data, so that tools may use a simple location redirection mechanism as described in Section 3.2 of [*OWL 2 Specification*] when executing test cases.

Each imported ontology is represented by an auxiliary individual with multiple property values:

- one value for the object property *:importedOntologyURI*, specifying the location that the ontology should be in,
- one or more values for the properties *:fsInputOntology*, *:owlXMLInputOntology*, or *:rdfXMLInputOntology*, specifying the contents of the ontology, possibly in different syntactic forms, and
- one or more values for the property *:normativeSyntax* to define which of the given syntactic forms is to be considered normative.

Tools that execute tests off-line can simulate imports by assuming that a document containing any of the provided normative-syntax input ontologies is located at the given URI.

```
        Declaration( ObjectProperty( :importedOntology ) )
        Declaration( ObjectProperty( :importedOntologyURI ) )
```

```
SubClassOf(
    SomeValuesFrom( InverseOf(:importedOntology) :TestCase )
    IntersectionOf(
        ExactCardinality( 1 :importedOntologyURI )
        MinCardinality( 1 :inputOntology )
        MinCardinality( 1 :normativeSyntax )
    )
)
```

3.2.8 Status

The *:status* object property specifies the status of a test case according to the test case approval process. The status might thus be PROPOSED, ACCEPTED, or REJECTED.

```
Declaration( ObjectProperty( :status ) )
FunctionalProperty( :status )
PropertyRange( :status OneOf( :Proposed :Approved :Rejected ) )
DifferentIndividuals( :Proposed :Approved :Rejected )
```

3.2.9 Identifier

The *:identifier* data property should be used to associate a unique identifier with a test case. This identifier should conform to **irelative-ref** as defined in [*RFC-3987*] so that it may be appended to http://www.w3.org/2007/OWL/owlt/ to generate a URI for the test. Values of this property are of type *xsd:string*.

```
Declaration( DataProperty( :identifier ) )
PropertyRange( :identifier xsd:string )
```

3.2.10 Creator

A test can be related to a literal description (name) of its author using the *:creator* data property. Values of this property are of type *xsd:string*. A test can have multiple creators.

```
Declaration( DataProperty( :creator ) )
PropertyRange( :creator xsd:string )
```

3.2.11 Description

A literal containing a human-readable description can associated with a test using *:description* data property. Values of this property are of type *xsd:string*.

```
        Declaration( DataProperty( :description ) )
        PropertyRange( :description xsd:string )
```

3.2.12 Specification Reference

Tests that are specifically related to a particular (part of an) OWL 2 specification document may indicate this using the *:specRef* object property. The value of this property is the URL (possibly with section reference) of the referred specification.

```
        Declaration( ObjectProperty( :specRef ) )
```

Note that this property is only provided to specify concrete URL references. To describe the relationship of some test to the specification more verbosely, the description can be used.

3.2.13 Issue Reference

The *:issue* object property can be used to associate a test with a specific WG issue. The value of this property is the URL of the according page in the Working Group's issue tracker.

```
        Declaration( ObjectProperty( :issue ) )
```

3.2.14 Complete Test Ontology

```
    Namespace( = <http://www.w3.org/2007/OWL/testOntology#> )
    Namespace( xsd = <http://www.w3.org/2001/XMLSchema#> )
    Ontology(<http://www.w3.org/2007/OWL/testOntology>
        Label("The OWL 2 Test Ontology")

        Declaration( Class( :TestCase ) )

        Declaration( DataProperty( :identifier ) )
        PropertyRange( :identifier xsd:string )
        Declaration( DataProperty( :description ) )
        PropertyRange( :description xsd:string )
        Declaration( DataProperty( :creator ) )
        PropertyRange( :creator xsd:string )
```

```
Declaration( ObjectProperty( :specRef ) )
Declaration( ObjectProperty( :issue ) )

Declaration( DataProperty( :inputOntology ) )
PropertyRange( :inputOntology xsd:string )
Declaration( DataProperty( :premiseOntology ) )
Declaration( DataProperty( :conclusionOntology ) )
Declaration( DataProperty( :nonConclusionOntology ) )

SubPropertyOf( :premiseOntology :inputOntology )
SubPropertyOf( :conclusionOntology :inputOntology )
SubPropertyOf( :nonConclusionOntology :inputOntology )


Declaration( Class( :ProfileIdentificationTest ) )

SubClassOf( :ProfileIdentificationTest :TestCase )
SubClassOf( :ProfileIdentificationTest MinCardinality( 1 :inputOntology


Declaration( Class( :ConsistencyTest ) )
Declaration( Class( :InconsistencyTest ) )

SubClassOf( :ConsistencyTest :ProfileIdentificationTest )
SubClassOf( :InconsistencyTest :ProfileIdentificationTest )
SubClassOf( :ConsistencyTest MinCardinality( 1 :premiseOntology ) )
SubClassOf( :InconsistencyTest MinCardinality( 1 :premiseOntology ) )
DisjointClasses( :ConsistencyTest :InconsistencyTest )


Declaration( Class( :PositiveEntailmentTest ) )

SubClassOf( :PositiveEntailmentTest :ConsistencyTest )
SubClassOf( :PositiveEntailmentTest MinCardinality( 1 :conclusionOntolo


Declaration( Class( :NegativeEntailmentTest ) )

SubClassOf( :NegativeEntailmentTest :ConsistencyTest )
SubClassOf( :NegativeEntailmentTest MinCardinality( 1 :nonConclusionOnt


Declaration( ObjectProperty( :status ) )
FunctionalProperty( :status )
PropertyRange( :status OneOf( :Proposed :Approved :Rejected ) )
DifferentIndividuals( :Proposed :Approved :Rejected )
```

```
Declaration( ObjectProperty( :species ) )
PropertyRange( :species OneOf( :DL :FULL ) )
DifferentIndividuals( :DL :FULL )
SubClassOf( HasValue( :species :DL ) HasValue( :species :FULL ) )


Declaration( ObjectProperty( :profile ) )
PropertyRange( :profile OneOf( :EL :QL :RL ) )
DifferentIndividuals( :EL :QL :RL )


SubClassOf(
    SomeValuesFrom( :profile OneOf( :EL  :QL ) )
    HasValue( :species :DL )
)


Declaration( ObjectProperty( :normativeSyntax ) )
PropertyRange( :normativeSyntax OneOf( :RDFXML :FUNCTIONAL :OWLXML ) )
DifferentIndividuals( :RDFXML :FUNCTIONAL :OWLXML )

SubClassOf( :TestCase MinCardinality( 1 :normativeSyntax ) )

SubClassOf(
    IntersectionOf( :TestCase ComplementOf( HasValue( :species :DL ) )
    HasValue( :normativeSyntax :RDFXML )
)

Declaration( ObjectProperty( :semantics ) )
Declaration( ObjectProperty( :alternativeSemanticsTest ) )

PropertyRange( :semantics OneOf( :DL :FULL ) )
FunctionalProperty( :alternativeSemanticsTest )
SymmetricProperty( :alternativeSemanticsTest )

SubClassOf(
  IntersectionOf( :TestCase ComplementOf( HasValue( :semantics :FULL )
  SomeValuesFrom( :alternativeSemanticsTest HasValue( :semantics :FULL
)

SubClassOf(
  IntersectionOf( :TestCase ComplementOf( HasValue( :semantics :DL ) )
  SomeValuesFrom( :alternativeSemanticsTest HasValue( :semantics :DL )
)


Declaration( DataProperty( :fsInputOntology ) )
SubPropertyOf( :fsInputOntology :inputOntology )
```

```
Declaration( DataProperty( :owlXmlInputOntology ) )
SubPropertyOf( :owlXmlInputOntology :inputOntology )

Declaration( DataProperty( :rdfXmlInputOntology ) )
SubPropertyOf( :rdfXmlInputOntology :inputOntology )

DisjointProperties( :fsInputOntology :owlXmlInputOntology :rdfXmlInputO


Declaration( ObjectProperty( :importedOntology ) )
Declaration( ObjectProperty( :importedOntologyURI ) )

SubClassOf(
    SomeValuesFrom( InverseOf(:importedOntology) :TestCase )
    IntersectionOf(
        ExactCardinality( 1 :importedOntologyURI )
        MinCardinality( 1 :inputOntology )
        MinCardinality( 1 :normativeSyntax )
    )
)

# The following "intersection properties" have not been described in th

Declaration( DataProperty( :fsPremiseOntology ) )
Declaration( DataProperty( :fsConclusionOntology ) )
Declaration( DataProperty( :fsNonConclusionOntology ) )
SubPropertyOf( :fsPremiseOntology :premiseOntology )
SubPropertyOf( :fsPremiseOntology :fsInputOntology )
SubPropertyOf( :fsConclusionOntology :conclusionOntology )
SubPropertyOf( :fsConclusionOntology :fsInputOntology )
SubPropertyOf( :fsNonConclusionOntology :nonConclusionOntology )
SubPropertyOf( :fsNonConclusionOntology :fsInputOntology )

Declaration( DataProperty( :owlXmlPremiseOntology ) )
Declaration( DataProperty( :owlXmlConclusionOntology ) )
Declaration( DataProperty( :owlXmlNonConclusionOntology ) )
SubPropertyOf( :owlXmlPremiseOntology :premiseOntology )
SubPropertyOf( :owlXmlPremiseOntology :owlXmlInputOntology )
SubPropertyOf( :owlXmlConclusionOntology :conclusionOntology )
SubPropertyOf( :owlXmlConclusionOntology :owlXmlInputOntology )
SubPropertyOf( :owlXmlNonConclusionOntology :nonConclusionOntology )
SubPropertyOf( :owlXmlNonConclusionOntology :owlXmlInputOntology )

Declaration( DataProperty( :rdfXmlPremiseOntology ) )
Declaration( DataProperty( :rdfXmlConclusionOntology ) )
Declaration( DataProperty( :rdfXmlNonConclusionOntology ) )
SubPropertyOf( :rdfXmlPremiseOntology :premiseOntology )
```

```
        SubPropertyOf( :rdfXmlPremiseOntology :rdfXmlInputOntology )
        SubPropertyOf( :rdfXmlConclusionOntology :conclusionOntology )
        SubPropertyOf( :rdfXmlConclusionOntology :rdfXmlInputOntology )
        SubPropertyOf( :rdfXmlNonConclusionOntology :nonConclusionOntology )
        SubPropertyOf( :rdfXmlNonConclusionOntology :rdfXmlInputOntology )
    )
```

3.3 Test Case Repositories

A set of approved test cases is provided in the OWL 2 Test Case Repository [OWL 2 Test Cases]. These test cases have been collected based on the approval process described below, and are expected to remain static after the Working Group has finished.

Like any test set, the approved OWL tests are necessarily incomplete in that they cannot cover all relevant situations or possible implementation challenges. For this reason, an additional public test repository [Contributed Test Cases] is provided as a platform for collecting further test cases even after the termination of the Working Group. Since the Working Group does not control the approval process for those additional test cases, they may not be subjected to extensive review and may result in erroneous or misleading information. It is hoped that the additional repository will provide a valuable tool for the development of OWL after the finalization of the recommendation.

3.4 Approval Process Overview

This section outlines the process by means of which test cases have been selected for inclusion into the OWL 2 Test Case Repository [OWL 2 Test Cases].

- At the chair's discretion, individual tests or groups of tests are put to the Working Group in the weekly telecon or at a face-to-face meeting.
- The Working Group may approve, reject, or defer decision on a test.
    - If the Working Group approves a test, its status is changed to APPROVED. All approved and only approved tests are included in the test case repository [OWL 2 Test Cases].
    - If the Working Group rejects a test, its status is changed to REJECTED.
    - If the Working Group defers decision on a test, its status remains PROPOSED.
- At the chairs' discretion, the Working Group may review any previous decision regarding any test cases.

The Working Group has complete discretion to approve or reject tests independent of their conformance with this process or their conformance with the OWL Working Drafts.

3.5 Changes From WebOnt Tests

This section provides an overview of the differences of the OWL 2 test cases format and collection as compared to the test cases of the first OWL specification as developed by the WebOnt working group [*OWL Test Cases*].

3.5.1 Formatting Changes

As explained above, changes of the test case format are motivated by the desire to supply test cases within single stand-alone documents that meet the syntactic conformance criteria of OWL 2 DL. In order to avoid confusion with the earlier test case format, all elements of the ontology use new URIs based on a dedicated namespace. Many properties still reflect the general structure of test cases, as outlined in [*Test Metadata*], and are applied in the same sense. In addition, some new ontology elements were introduced to account for aspects that are specific to OWL 2 (e.g. the *:profile* property).

Besides the change in vocabulary, the main structural change compared to WebOnt test cases is the embedding of all relevant data in single files, instead of using multiple files for each involved ontology.

3.5.2 Changes to Test Types

"Profile Identification Tests" and "Syntax Translation Tests" did not exist in the WebOnt test suite.

"Tests for Incorrect Use of OWL Namespace" has been removed as a type. These tests were intended to highlight differences between the OWL RDF vocabulary and the DAML+OIL vocabulary. Time has reduced the motivation for such tests.

"True Tests", "OWL for OWL Tests", and "Import Entailment Tests" have been removed as types. These types were each specializations of entailment tests. To the extent that they are present in the current test suite, these tests are marked as positive entailment tests.

"Import Level Tests" has been removed as a type. This type is now included in the "Profile Identification Tests".

3.5.3 Changes to Process

Status of each test no longer includes "EXTRACREDIT" and "OBSOLETED".

4 Appendix: An Inconsistent Ontology

Consistency tests and inconsistency tests can be considered as entailment tests and non-entailment tests, respectively, by checking the entailment of an (arbitrary) inconsistent ontology. This appendix provides an ontology document d_{in} in the functional-style syntax that can be used for this purpose and that is compatible with all profiles. The corresponding ontology O_{in}=$Ont(d_{in})$ (see Section 2.2) is inconsistent with respect to both the direct semantics [*OWL 2 Direct Semantics*] and the RDF-Based semantics [*OWL 2 RDF-Based Semantics*]).

```
Namespace( owl =  <http://www.w3.org/2002/07/owl#> )
Ontology(<http://example.com/inconsistentOntology>
    SubClassOf( owl:Thing owl:Nothing )
)
```

5 Acknowledgments

The starting point for the development of OWL 2 was the OWL1.1 member submission, itself a result of user and developer feedback, and in particular of information gathered during the OWL Experiences and Directions (OWLED) Workshop series. The working group also considered postponed issues from the WebOnt Working Group.

This document is the product of the OWL Working Group (see below) whose members deserve recognition for their time and commitment. The editors extend special thanks to Bernardo Cuenca Grau (Oxford University), Michael Schneider (FZI) and Mike Smith (Clark & Parsia) for their thorough reviews.

The regular attendees at meetings of the OWL Working Group at the time of publication of this document were: Jie Bao (RPI), Diego Calvanese (Free University of Bozen-Bolzano), Bernardo Cuenca Grau (Oxford University), Martin Dzbor (Open University), Achille Fokoue (IBM Corporation), Christine Golbreich (Université de Versailles St-Quentin), Sandro Hawke (W3C/MIT), Ivan Herman (W3C/ERCIM), Rinke Hoekstra (University of Amsterdam), Ian Horrocks (Oxford University), Elisa Kendall (Sandpiper Software), Markus Krötzsch (FZI), Carsten Lutz (Universität Bremen), Boris Motik (Oxford University), Jeff Pan (University of Aberdeen), Bijan Parsia (University of Manchester), Peter F. Patel-Schneider (Bell Labs Research, Alcatel-Lucent), Alan Ruttenberg (Science Commons), Uli Sattler (University of Manchester), Michael Schneider (FZI), Mike Smith (Clark & Parsia), Evan Wallace (NIST), and Zhe Wu (Oracle Corporation). We would also like to thank past members of the working group: Jeremy Carroll, Jim Hendler and Vipul Kashyap.

6 References

[Contributed Test Cases]
Public Test Case Repository.

Editor's Note: The final location of the repository of user-contributed test cases is yet to be determined.

[OWL 2 Specification]
OWL 2 Web Ontology Language:Structural Specification and Functional-Style Syntax Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-syntax-20081202/. Latest version available at http://www.w3.org/TR/owl2-syntax/.

[OWL 2 Direct Semantics]
OWL 2 Web Ontology Language:Direct Semantics Boris Motik, Peter F. Patel-Schneider, Bernardo Cuenca Grau, eds. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-semantics-20081202/. Latest version available at http://www.w3.org/TR/owl2-semantics/.

[OWL 2 RDF-Based Semantics]
OWL 2 Web Ontology Language:RDF-Based Semantics Michael Schneider, editor. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-rdf-based-semantics-20081202/. Latest version available at http://www.w3.org/TR/owl2-rdf-based-semantics/.

[OWL 2 RDF Mapping]
OWL 2 Web Ontology Language:Mapping to RDF Graphs Peter F. Patel-Schneider, Boris Motik, eds. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-mapping-to-rdf-20081202/. Latest version available at http://www.w3.org/TR/owl2-mapping-to-rdf/.

[OWL 2 XML Syntax]
OWL 2 Web Ontology Language:XML Serialization Boris Motik, Peter Patel-Schneider, eds. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-xml-serialization-20081202/. Latest version available at http://www.w3.org/TR/owl2-xml-serialization/.

[OWL 2 Profiles]
OWL 2 Web Ontology Language:Profiles Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, Carsten Lutz, eds. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-profiles-20081202/. Latest version available at http://www.w3.org/TR/owl2-profiles/.

[OWL 2 Test Cases]
OWL 2 Test Case Repository.

Editor's Note: The final location of the repository of approved test cases is yet to be determined.

[OWL Test Cases]
OWL Web Ontology Language: Test Cases. Jeremy J. Carroll and Jos De Roo, eds., W3C Recommendation 10 February 2004, http://www.w3.org/TR/

2004/REC-owl-test-20040210/. Latest version available at http://www.w3.org/TR/owl-test/.

[RDF Syntax]
RDF/XML Syntax Specification (Revised). Dave Beckett, ed., W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Latest version available at http://www.w3.org/TR/rdf-syntax-grammar/.

[SROIQ]
The Even More Irresistible SROIQ. Ian Horrocks, Oliver Kutz, and Uli Sattler. In Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). AAAI Press, 2006.

[OWL Metamodeling]
On the Properties of Metamodeling in OWL. Boris Motik. *Journal of Logic and Computation*, 17(4):617-637, 2007.

[RFC 2119]
RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. Network Working Group, S. Bradner. Internet Best Current Practice, March 1997.

[RFC-3987]
RFC 3987: Internationalized Resource Identifiers (IRIs). M. Duerst, M. Suignard. IETF, January 2005, http://www.ietf.org/rfc/rfc3987.txt.

[Test Metadata]
Test Metadata. Patrick Curran and Karl Dubost, eds., W3C Working Group Note 14 September 2005, http://www.w3.org/TR/2005/NOTE-test-metadata-20050914/. Latest version available at http://www.w3.org/TR/test-metadata/.