W3C Editor's Draft

OWL 2 Web Ontology Language Conformance

W3C Editor's Draft 14 September 2009

This version:
http://www.w3.org/2007/OWL/draft/ED-owl2-conformance-20090914/
Latest editor's draft:
http://www.w3.org/2007/OWL/draft/owl2-conformance/
Previous version:
http://www.w3.org/2007/OWL/draft/ED-owl2-test-20090531/ (color-coded diff)
Editors:
Michael Smith, Clark & Parsia
Ian Horrocks, Oxford University
Markus Krötzsch, FZI Karlsruhe
Birte Glimm, Oxford University
Contributors: (in alphabetical order)
Sandro Hawke, W3C/MIT
Matthew Horridge, University of Manchester
Bijan Parsia, University of Manchester
Michael Schneider, FZI Research Center for Information Technology

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

Abstract

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. The OWL 2 Document Overview describes the overall state of OWL 2, and should be read before other OWL 2 documents.
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May Be Superseded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/.

Summary of Changes

This document has undergone only minor changes since the previous version of 11th June, 2009.

- Some conformance conditions relating to the OWL 2 datatype map were removed as the strengthening of the definition in Syntax renders them redundant.
- The role of "Extra Credit" test cases was clarified.

Please Comment By 12 October 2009

The OWL Working Group seeksformal review from members of the W3C Advisory Committee, via @@@TBD.

Others are welcome to continue to send reports of implementation experience, and other feedback, to public-owl-comments@w3.org (public archive). Reports of any success or difficulty with the test cases are encouraged. Open discussion among developers is welcome at public-owl-dev@w3.org (public archive).

No Endorsement

Publication as a Editor's 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

W3C Editor's Draft

the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of 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 "incomplete" in the sense that passing all the tests does not prove that a given system conforms to the OWL 2 specification; failing a test does, however, prove that the system does not conform to the 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* be able to publish them in the RDF/XML serialization if asked to do so (e.g., via HTTP

content negotiation), provided that the ontology can be so serialized. 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

For documents using the RDF/XML serialization, syntactic conformance is defined as follows:

An **OWL 2 Full ontology document** is any RDF/XML document [*RDF Syntax*].

An **OWL 2 DL ontology document** is an OWL 2 Full ontology document that can be successfully parsed using the canonical parsing process as defined in the OWL 2 Syntax specification [*OWL 2 Specification*] and 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*] to produce an instance of the OWL 2 ontology class satisfying all of the restrictions described in Section 3 of the OWL 2 Syntax specification [*OWL 2 Specification*].

An **OWL 2 EL ontology document** is an OWL 2 DL ontology document where the corresponding instance of the OWL 2 ontology class satisfies the definition of an OWL 2 EL ontology given in the OWL 2 Profiles specification [*OWL 2 Profiles*].

An **OWL 2 QL ontology document** is an OWL 2 DL ontology document where the corresponding instance of the OWL 2 ontology class satisfies the definition of an OWL 2 QL ontology given in the OWL 2 Profiles specification [*OWL 2 Profiles*].

An **OWL 2 RL ontology document** is an OWL 2 DL ontology document where the corresponding instance of the OWL 2 ontology class satisfies the definition of an OWL 2 RL ontology given in the OWL 2 Profiles specification [*OWL 2 Profiles*].

For documents using other serializations, conformance 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*], it can be successfully parsed using the canonical parsing process as defined in the OWL 2 Syntax specification [*OWL 2 Specification*], and the resulting instance of the OWL 2 ontology class satisfies all of the restrictions described in Section 3 of the OWL 2 Syntax specification [*OWL 2 Specification*]; it is an OWL 2 EL (respectively QL, RL) ontology document iff the resulting instance of the OWL 2 ontology class

satisfies the definition of an OWL 2 EL (respectively QL, RL) ontology given in the OWL 2 Profiles specification [*OWL 2 Profiles*].

Note that:

1. The conditions for document conformance are entirely syntactic and say nothing about semantics. For example, an OWL 2 DL ontology document is simply an OWL 2 Full ontology document that satisfies certain syntactic constraints, and it could be taken as input by tools that use either the Direct Semantics [*OWL 2 Direct Semantics*] or the RDF-Based Semantics [*OWL 2 RDF-Based Semantics*].
2. 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 the OWL 2 Datatype map. Unsupported datatypes cannot occur in conforming documents, so syntactic and semantic conditions on these datatypes are irrelevant and can be ignored.

2.2 Tool Conformance

An OWL 2 tool takes one or more OWL 2 ontology documents and checks some (syntactic or semantic) condition. A given tool is characterized by three largely independent parameters:

- the ontology documents it accepts (Full, DL, EL, QL or RL);
- the semantics it applies (Direct [*OWL 2 Direct Semantics*] or RDF-Based [*OWL 2 RDF-Based Semantics*]).

When the Direct Semantics are being applied, semantic conditions are defined with respect to ontology structures (i.e., instances of the **Ontology** class as defined in the OWL 2 Syntax specification [*OWL 2 Specification*]), and we denote with *Ont(d)* the ontology structure corresponding to an input ontology document *d*. When the RDF-Based semantics are being applied, semantic conditions are defined with respect to RDF graphs, and we denote with *Ont(d)* the RDF graph corresponding to an input ontology document *d*.

As noted above, any conformant OWL 2 tool *must* accept ontology documents using the RDF/XML serialization [*OWL 2 Mapping to RDF Graphs*]. Given an ontology document *d* in the RDF/XML serialization, for a tool applying the Direct Semantics, *Ont(d)* denotes the ontology structure obtained by applying to *d* the canonical parsing process as defined in the OWL 2 Syntax specification [*OWL 2 Specification*] using, in steps 2.2 and 3.3, 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*]; for a tool applying the RDF-Based

semantics, *Ont(d)* simply denotes the RDF graph corresponding to *d*, as defined in [*RDF Syntax*].

A conformant OWL 2 tool *may* also accept ontology documents using other serializations, for example Turtle [*Turtle*] or the XML Serialization [*OWL 2 XML Syntax*]. Alternative RDF serializations are treated in much the same way as RDF/XML, i.e., *Ont(d)* denotes either the ontology structure obtained from *d* via the procedure for mapping from RDF graphs to the structural specification or the RDF graph corresponding to *d*, depending on the semantics being applied. When using non-RDF serializations, e.g., the XML Serialization [*OWL 2 XML Syntax*], *Ont(d)* denotes either the ontology structure obtained from *d* using the canonical parsing process as defined in the OWL 2 Syntax specification [*OWL 2 Specification*], or the RDF graph obtained by applying the OWL 2 Mapping to RDF Graphs [*OWL 2 Mapping to RDF Graphs*] to the ontology structure obtained from *d*, depending on the semantics being applied.

The conformance conditions related to entailment checking and query answering are defined below. Similar conditions would apply to other OWL 2 tools. 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 ontology documents d_1 and d_2 and checks whether $Ont(d_1)$ entails $Ont(d_2)$ with respect to 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 $Ont(d_2)$ denote ontology structures that satisfy all of the restrictions on OWL 2 ontologies described in Section 3 of the OWL 2 Syntax specification [*OWL 2 Specification*]; when using the RDF-based Semantics, $Ont(d_1)$ and $Ont(d_2)$ denote RDF graphs. Additionally, an OWL 2 entailment checker:

- *must* provide a means to determine any limits it has on datatype lexical forms — they could, for example, be listed in supporting documentation (see Section 4 of the OWL 2 Syntax specification [*OWL 2 Specification*]); 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 datatype lexical forms that exceed any limits it has on datatype lexical forms — 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 OWL 2 Full ontology documents as input. It *must* return `True` only when $Ont(d_1)$ entails $Ont(d_2)$, and it *must* return `False` only when $Ont(d_1)$ does not entail $Ont(d_2)$. 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 $Ont(d_2)$, and it *must* return `False` only when $Ont(d_1)$ does not entail $Ont(d_2)$. 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 $Ont(d_2)$, and it *must* return `False` only when $Ont(d_1)$ does not entail $Ont(d_2)$. 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 $Ont(d_2)$, and it *must* return `False` only when $Ont(d_1)$ does not entail $Ont(d_2)$. It *should not* return `Unknown`.

An OWL 2 RL entailment checker is an OWL 2 entailment checker that takes OWL 2 Full ontology documents as input. It *must* return `True` only when $Ont(d_1)$ entails $Ont(d_2)$, and it *must* return `False` only when $Ont(d_1)$ does not entail $Ont(d_2)$. If applying the Direct Semantics, it *should not* return `Unknown`. If applying the RDF-Based Semantics, it *should not* return `Unknown` if it is possible to derive `True` using the OWL 2 RL/RDF rules; more formally, it *should not* return `Unknown` if $FO(Ont(d_1)) \cup R$ entails $FO(Ont(d_2))$ under the standard first-order semantics, where R denotes the OWL 2 RL/RDF rules, and $FO(Ont(d_i))$ denotes the first order 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`.

Note that it follows from Theorem PR1 of Profiles [*OWL 2 Profiles*] that it is always safe for an OWL 2 RL entailment checker using the RDF-Based Semantics to return `False` if:

- d_1 and d_2 are OWL 2 RL ontology documents;
- $Ont(d_1)$ and $Ont(d_2)$ satisfy the constraints described in Theorem PR1; and
- it is not possible to derive `True` using the OWL 2 RL/RDF rules (see above for a more formal definition of what this means).

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; it is **complete** if, given sufficient resources, it will always return `True` or `False` on syntactically-valid inputs.

2.2.2 Query Answering Tool

Query answering is closely related to entailment checking (see Section 2.5 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*]). 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 a query answering tool based on such an entailment checker *should* be both sound and complete. In the case of OWL 2 RL, a query answering tool based on an OWL 2 RL entailment checker *should* be sound; a tool based on an OWL 2 RL entailment checker using the Direct Semantics *should* also be complete; and a tool based on an OWL 2 RL entailment checker using the RDF-Based Semantics and the OWL 2 RL/RDF rules *should* also be complete if both O and Q are OWL 2 RL ontology documents, and $Ont(O)$ and $Ont(Q)$ 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 tools that process OWL 2 ontology documents, or that transform between various syntactic forms of OWL 2. 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 and Species Identification Tests

Profile and species 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 2 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 2 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. When using the direct semantics [*OWL 2 Direct Semantics*] the abstract structure is an OWL 2 ontology as defined in the OWL 2 Syntax specification [*OWL 2 Specification*]; when using the RDF-based semantics [*OWL 2 RDF-Based Semantics*] the abstract structure is an RDF graph. 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.

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 as if they were non-entailment tests: 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 transform inconsistency tests into entailment tests: if the ontology *Ont(d)* is inconsistent, then *Ont(d)* entails any inconsistent ontology — e.g., the inconsistent ontology O_{in} given in 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 2, 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 any OWL tool that can handle at least OWL 2 DL ontology documents. Thus the presented test case ontology and all test case documents using it conform to the definition of OWL 2 DL ontology documents. This design choice also motivates the use of dedicated IRIs (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 not been crafted this way and do not meet the above requirements. Details of the changes in the test case format as compared to WebOnt are found in Section 3.5.

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.

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 ) )
DataPropertyRange( :inputOntology xsd:string )

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

SubDataPropertyOf( :premiseOntology :inputOntology )
SubDataPropertyOf( :conclusionOntology :inputOntology )
SubDataPropertyOf( :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 ) )

SubDataPropertyOf( :fsInputOntology :inputOntology )
SubDataPropertyOf( :owlXmlInputOntology :inputOntology )
SubDataPropertyOf( :rdfXmlInputOntology :inputOntology )

DisjointDataProperties( :fsInputOntology :owlXmlInputOntology :rdfXmlI
```

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( :SyntaxTranslationTest ) )
Declaration( Class( :ConsistencyTest ) )
Declaration( Class( :InconsistencyTest ) )
Declaration( Class( :PositiveEntailmentTest ) )
Declaration( Class( :NegativeEntailmentTest ) )

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

SubClassOf( :ConsistencyTest DataMinCardinality( 1 :premiseOntology )
SubClassOf( :InconsistencyTest DataMinCardinality( 1 :premiseOntology
SubClassOf( :PositiveEntailmentTest DataMinCardinality( 1 :conclusionO
SubClassOf( :NegativeEntailmentTest DataMinCardinality( 1 :nonConclusi

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. syntax translation tests may provide multiple normative syntactic forms.

The property *:normativeSyntax* may take an instance of the *:Syntax* class as a value. The following mutually different individuals are members of the *:Syntax* class:

- The individual *:FUNCTIONAL* indicates that all functional syntax input ontologies associated with the test case are normative.
- The individual *:OWLXML* indicates that all OWL 2 XML syntax input ontologies associated with the test case are normative.
- The individual *:RDFXML* indicates that all RDF/XML syntax input ontologies associated with the test case are normative.

```
Declaration( Class( :Syntax ) )
ClassAssertion( :Syntax :RDFXML )
ClassAssertion( :Syntax :FUNCTIONAL )
ClassAssertion( :Syntax :OWLXML )
DifferentIndividuals( :RDFXML :FUNCTIONAL :OWLXML )

Declaration( ObjectProperty( :normativeSyntax ) )
ObjectPropertyRange( :normativeSyntax :Syntax )

SubClassOf( :TestCase ObjectMinCardinality( 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 *:RDF-BASED* indicates that the test is applicable if the [*OWL 2 RDF-Based Semantics*] are used.
- The individual *:DIRECT* 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( :DIRECT :RDF-BASED )
ObjectPropertyRange( :semantics ObjectOneOf( :DIRECT :RDF-BASED ) )
```

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 the RDF-based semantics might be a non-entailment in the direct semantics). The symmetric property *:alternativeSemanticsTest* is used to associate two test cases that are complementary in this sense.

```
Declaration( ObjectProperty( :alternativeSemanticsTest ) )
FunctionalObjectProperty( :alternativeSemanticsTest )
SymmetricObjectProperty( :alternativeSemanticsTest )

SubClassOf(
  ObjectIntersectionOf( :TestCase ObjectComplementOf( ObjectHasValue(
  ObjectSomeValuesFrom( :alternativeSemanticsTest ObjectHasValue( :sem
)

SubClassOf(
  ObjectIntersectionOf( :TestCase ObjectComplementOf( ObjectHasValue(
  ObjectSomeValuesFrom( :alternativeSemanticsTest ObjectHasValue( :sem
)
```

3.2.5 Species

The *:species* property describes the syntactic conformance of the input ontology documents with respect to OWL 2 Full ontology documents and OWL 2 DL ontology documents. 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 2 DL ontology, or a negative property assertion indicating that it is not.

```
Declaration( ObjectProperty( :species ) )
ObjectPropertyRange( :species ObjectOneOf( :DL :FULL ) )
DifferentIndividuals( :DL :FULL )
SubClassOf( :TestCase ObjectHasValue( :species :FULL ) )
```

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 ) )
ObjectPropertyRange( :profile ObjectOneOf( :EL :QL :RL ) )
DifferentIndividuals( :EL :QL :RL )
```

The following axiom reflects the fact that if an ontology conforms to the restrictions of OWL 2 EL, OWL 2 QL, or OWL 2 RL it also conforms to the restrictions of OWL 2 DL.

```
SubClassOf(
    ObjectSomeValuesFrom( :profile ObjectOneOf( :EL :QL :RL ) )
    ObjectHasValue( :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 *:importedOntologyIRI*, 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 IRI.

```
Declaration( ObjectProperty( :importedOntology ) )
Declaration( ObjectProperty( :importedOntologyIRI ) )

SubClassOf(
    ObjectSomeValuesFrom( InverseOf(:importedOntology) :TestCase )
    ObjectIntersectionOf(
        ObjectExactCardinality( 1 :importedOntologyIRI )
        DataMinCardinality( 1 :inputOntology )
        ObjectMinCardinality( 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, APPROVED, REJECTED, or EXTRACREDIT.

```
Declaration( ObjectProperty( :status ) )
FunctionalObjectProperty( :status )
```

```
ObjectPropertyRange( :status ObjectOneOf( :Proposed :Approved :Rejecte
DifferentIndividuals( :Proposed :Approved :Rejected :Extracredit )
```

3.2.9 Identifier

The *:identifier* data property should be used to associate a unique identifier with a test case.

```
Declaration( DataProperty( :identifier ) )
```

3.2.10 Creator

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

```
Declaration( DataProperty( :creator ) )
```

3.2.11 Description

A literal containing a human-readable description can associated with a test using *:description* data property.

```
Declaration( DataProperty( :description ) )
```

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#> )
Namespace( rdfs = <http://www.w3.org/2000/01/rdf-schema#> )
Ontology(<http://www.w3.org/2007/OWL/testOntology>
   Annotation( rdfs:label "The OWL 2 Test Ontology" )
   Annotation( rdfs:isDefinedBy <http://www.w3.org/TR/owl2-test/> )

   Declaration( Class( :TestCase ) )

   Declaration( DataProperty( :identifier ) )
   Declaration( DataProperty( :description ) )
   Declaration( DataProperty( :creator ) )
   Declaration( ObjectProperty( :specRef ) )
   Declaration( ObjectProperty( :issue ) )

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

   SubDataPropertyOf( :premiseOntology :inputOntology )
   SubDataPropertyOf( :conclusionOntology :inputOntology )
   SubDataPropertyOf( :nonConclusionOntology :inputOntology )


   Declaration( Class( :ProfileIdentificationTest ) )

   SubClassOf( :ProfileIdentificationTest :TestCase )
   SubClassOf( :ProfileIdentificationTest DataMinCardinality( 1 :inputOnt


   Declaration( Class( :SyntaxTranslationTest ) )

   SubClassOf( :SyntaxTranslationTest :TestCase )
```

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

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


Declaration( Class( :PositiveEntailmentTest ) )

SubClassOf( :PositiveEntailmentTest :ConsistencyTest )
SubClassOf( :PositiveEntailmentTest DataMinCardinality( 1 :conclusionO


Declaration( Class( :NegativeEntailmentTest ) )

SubClassOf( :NegativeEntailmentTest :ConsistencyTest )
SubClassOf( :NegativeEntailmentTest DataMinCardinality( 1 :nonConclusi


Declaration( ObjectProperty( :status ) )
FunctionalObjectProperty( :status )
ObjectPropertyRange( :status ObjectOneOf( :Proposed :Approved :Rejecte
DifferentIndividuals( :Proposed :Approved :Rejected :Extracredit )


Declaration( ObjectProperty( :species ) )
ObjectPropertyRange( :species ObjectOneOf( :DL :FULL ) )
DifferentIndividuals( :DL :FULL )
SubClassOf( ObjectHasValue( :species :DL ) ObjectHasValue( :species :F


Declaration( ObjectProperty( :profile ) )
ObjectPropertyRange( :profile ObjectOneOf( :EL :QL :RL ) )
DifferentIndividuals( :EL :QL :RL )


SubClassOf(
    ObjectSomeValuesFrom( :profile ObjectOneOf( :EL :QL :RL ) )
    ObjectHasValue( :species :DL )
)

Declaration( Class( :Syntax ) )
ClassAssertion( :Syntax :RDFXML )
ClassAssertion( :Syntax :FUNCTIONAL )
ClassAssertion( :Syntax :OWLXML )
DifferentIndividuals( :RDFXML :FUNCTIONAL :OWLXML )

Declaration( ObjectProperty( :normativeSyntax ) )
```

```
ObjectPropertyRange( :normativeSyntax :Syntax )

SubClassOf( :TestCase ObjectMinCardinality( 1 :normativeSyntax ) )

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

ObjectPropertyRange( :semantics ObjectOneOf( :DIRECT :RDF-BASED ) )
DifferentIndividuals( :DIRECT :RDF-BASED )
FunctionalObjectProperty( :alternativeSemanticsTest )
SymmetricObjectProperty( :alternativeSemanticsTest )

SubClassOf(
  ObjectIntersectionOf( :TestCase ObjectComplementOf( ObjectHasValue(
  ObjectSomeValuesFrom( :alternativeSemanticsTest ObjectHasValue( :sem
)

SubClassOf(
  ObjectIntersectionOf( :TestCase ObjectComplementOf( ObjectHasValue(
  ObjectSomeValuesFrom( :alternativeSemanticsTest ObjectHasValue( :sem
)


Declaration( DataProperty( :fsInputOntology ) )
SubDataPropertyOf( :fsInputOntology :inputOntology )

Declaration( DataProperty( :owlXmlInputOntology ) )
SubDataPropertyOf( :owlXmlInputOntology :inputOntology )

Declaration( DataProperty( :rdfXmlInputOntology ) )
SubDataPropertyOf( :rdfXmlInputOntology :inputOntology )

DisjointDataProperties( :fsInputOntology :owlXmlInputOntology :rdfXmlI


Declaration( ObjectProperty( :importedOntology ) )
Declaration( ObjectProperty( :importedOntologyIRI ) )

SubClassOf(
    ObjectSomeValuesFrom( InverseOf(:importedOntology) :TestCase )
    ObjectIntersectionOf(
        ObjectExactCardinality( 1 :importedOntologyIRI )
        DataMinCardinality( 1 :inputOntology )
        ObjectMinCardinality( 1 :normativeSyntax )
    )
)


Declaration( DataProperty( :fsPremiseOntology ) )
Declaration( DataProperty( :fsConclusionOntology ) )
Declaration( DataProperty( :fsNonConclusionOntology ) )
```

```
    SubDataPropertyOf( :fsPremiseOntology :premiseOntology )
    SubDataPropertyOf( :fsPremiseOntology :fsInputOntology )
    SubDataPropertyOf( :fsConclusionOntology :conclusionOntology )
    SubDataPropertyOf( :fsConclusionOntology :fsInputOntology )
    SubDataPropertyOf( :fsNonConclusionOntology :nonConclusionOntology )
    SubDataPropertyOf( :fsNonConclusionOntology :fsInputOntology )

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

    Declaration( DataProperty( :rdfXmlPremiseOntology ) )
    Declaration( DataProperty( :rdfXmlConclusionOntology ) )
    Declaration( DataProperty( :rdfXmlNonConclusionOntology ) )
    SubDataPropertyOf( :rdfXmlPremiseOntology :premiseOntology )
    SubDataPropertyOf( :rdfXmlPremiseOntology :rdfXmlInputOntology )
    SubDataPropertyOf( :rdfXmlConclusionOntology :conclusionOntology )
    SubDataPropertyOf( :rdfXmlConclusionOntology :rdfXmlInputOntology )
    SubDataPropertyOf( :rdfXmlNonConclusionOntology :nonConclusionOntology
    SubDataPropertyOf( :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 2 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*]. The working group may decide to approve some tests as EXTRACREDIT, indicating that implementors are not expected to implement the required functionality. Further details are found below.
    - 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.

Extra credit tests are approved test cases of status EXTRACREDIT. These tests have been found to be particularly hard to implement efficiently. Many extra credit tests are rather testing the performance of an implementation than highlighting a particular semantic interaction of OWL constructs. The name indicates that there is no expectation that any implementation will successfully run such tests and any that do gain extra credit.

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 IRIs 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 separate 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 "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 2.3) 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 has been produced by the OWL Working Group (see below), and its contents reflect extensive discussions within the Working Group as a whole. The editors extend special thanks to Bernardo Cuenca Grau (Oxford University), Peter F. Patel-Schneider (Bell Labs Research, Alcatel-Lucent), Jonathan Rees (Science Commons), Alan Ruttenberg (Science Commons) and Michael Schneider (FZI) 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 and LIRMM), 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), Deborah L. McGuinness (RPI), Boris Motik (Oxford University), Jeff Pan (University of Aberdeen), Bijan Parsia (University of Manchester), Peter F. Patel-Schneider (Bell Labs Research, Alcatel-Lucent), Sebastian Rudolph (FZI), Alan Ruttenberg (Science Commons), Uli Sattler (University of Manchester), Michael Schneider (FZI), Mike Smith (Clark & Parsia), Evan Wallace (NIST), Zhe Wu (Oracle Corporation), and Antoine Zimmermann (DERI Galway). We would also like to thank past members of the working group: Jeremy Carroll, Jim Hendler, 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 Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-syntax/.

[OWL 2 Direct Semantics]
OWL 2 Web Ontology Language: Direct Semantics Boris Motik, Peter F. Patel-Schneider, Bernardo Cuenca Grau, eds. W3C Editor's Draft, 14

September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-direct-semantics-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-direct-semantics/.

[OWL 2 RDF-Based Semantics]
OWL 2 Web Ontology Language: RDF-Based Semantics Michael Schneider, editor. W3C Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-rdf-based-semantics-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/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 Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-mapping-to-rdf-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-mapping-to-rdf/.

[OWL 2 XML Syntax]
OWL 2 Web Ontology Language: XML Serialization Boris Motik, Bijan Parsia, Peter Patel-Schneider, eds. W3C Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-xml-serialization-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/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 Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-profiles-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-profiles/.

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

[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/.

[Turtle]
Turtle - Terse RDF Triple Language. David Beckett and Tim Berners-Lee, W3C Team Submission 14 January 2008, http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/. Latest version available at http://www.w3.org/TeamSubmission/turtle/.