W3C Working Draft

OWL 2 Web Ontology Language: RDF-Based Semantics

W3C Working Draft 02 December 2008

This version:
http://www.w3.org/TR/2008/WD-owl2-rdf-based-semantics-20081202/
Latest version:
http://www.w3.org/TR/owl2-rdf-based-semantics/
Previous version:
http://www.w3.org/TR/2008/WD-owl2-rdf-based-semantics-20081008/
Editors:
Michael Schneider, FZI Research Center for Information Technology
Contributors:
Jeremy Carroll, HP (now at TopQuadrant)
Peter F Patel-Schneider, Bell Labs Research, Alcatel-Lucent

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 provides the RDF-compatible model-theoretic semantics for OWL 2, called "OWL 2 Full".

W3C Working Draft

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May Be Superseded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 (this document)
4. Conformance and Test Cases
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

Summary of Changes

This section lists significant changes since the First Public Working Draft of 08 October 2008.

- Added datatype "owl:rational", marking it "at risk" (WG resolution of Issue 87).
- The RDF syntax of self restrictions has been changed: The class owl:SelfRestriction has been replaced by the property owl:hasSelf (per WG resolution).
- Removed the semantic conditions for axiom annotations (WG resolution of Issue 144).
- Added semantic conditions inferring a union or intersection of datatypes into a datatype (following WG resolution of Issue 147).
- The URIs *owl:TopObjectProperty*, *owl:BottomObjectProperty*, *owl:TopDataProperty* and *owl:BottomDataProperty* have been renamed to their lower-case variants, respectively (per WG decision).
- The datatypes xsd:ID, xsd:IDREF and xsd:ENTITY have been removed (per WG resolution).
- Changed the semantic conditions for the n-ary value restrictions to infer the type of the properties $p_1,...,p_n$ (IODP) and the type of the class c (IDC).

- Corrected definitions of consistency and entailment: The vocabulary V was a global parameter of the definitions. Now the form of the definitions is close to the respective definitions in OWL 1.
- Corrected the semantic condition for sub property chains: missing premise "q in IP" in the second condition.
- Removed redundant statements in the consequent of the semantic conditions for negative property assertions.
- For D-Interpretations, the range of the mapping IL has been changed to IR instead of LV, with a reference to the RDF Semantics. This was a bug, since in both the RDF Semantics and in OWL 1 the range of IL has been IR.
- Splitted the table on "Parts of the Universe" in the "Semantic Conditions" section into a table defining the parts (now in the "Interpretations" section), and a table that specifies the semantic conditions for those parts.
- The definition of OWL 2 Full datatype maps now include the different facet-related sets that have formerly been part of the "abbreviations" table in the "Semantic Conditions" section.
- The nomenclature for datatype maps has been aligned with the one used in the RDF Semantics. In particular, the concept being called an "interpretation of a literal" is now being called a "datatype value", and the concept being called an "interpretation of a datatype" is now being called a "datatype" or the "value space" of a datatype, depending on whether the datatype itself or its class extension is meant.
- Replaced all applications of the URI-mapping 'IS(.)' by the more general interpretation function 'I(.)'. This usage is now in line with the usage in the RDF Semantics document. Also, there have formerly been applications of IS, where it was not guaranteed that the argument is a URI.
- Marked several sections as "Informative", as requestion by a previous review.
- Added to the "Ontologies" section some text about ontology headers and ontology versions, but removed every text referring to the semantic meaning of a OWL 2 Full ontology.
- Moved the "Ontologies" section from Section 5 to Section 3.
- Moved the discussion on axiomatic triples from the section on "Semantic Conditions" to a dedicated appendix.
- The "Introduction" section has been revised.
- The descriptions of the semantic condition tables have been revised.

This is a Recommendation-Track document.

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.

W3C Working Draft

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.

Contents

1 Introduction (Informative)

This document defines the RDF-compatible model-theoretic semantics of OWL 2, called "OWL 2 Full". The semantics given here is the OWL 2 semantic extension of RDFS [*RDF Semantics*]. Therefore, the semantic meaning given to an RDF graph by OWL 2 Full includes the meaning given to the graph by RDFS. Beyond that,

OWL 2 Full gives additional meaning to all the language features of OWL 2, by following the design principles that have been applied to the semantics of RDF.

OWL 2 Full accepts every well-formed RDF graph [*RDF*] as a syntactically valid OWL 2 Full ontology, and gives a precise semantic meaning to it. The semantic meaning is determined by the set of *OWL 2 Full semantic conditions*, which include and extend all the semantic conditions for RDF and RDFS specified in [*RDF Semantics*]. OWL 2 Full acts as a *vocabulary interpretation* for the RDF and the RDFS vocabularies, and for the *OWL 2 Full vocabulary*. The OWL 2 Full vocabulary is a set of URIs that occur in the sets of RDF triples, which define the RDF syntax of OWL 2 [*OWL 2 RDF Mapping*]. The OWL 2 Full semantic conditions specify exactly which triple sets are assigned a specific meaning, and what this meaning is.

OWL 2 Full interpretations are defined on the *OWL 2 Full universe*. The OWL 2 Full universe is identified with the RDFS universe, and comprises the set of all individuals. It is further divided into "parts", namely the classes, the properties, and the datatype values. Thus, the members of these parts are also individuals. Every class has a set of individuals associated with it, the so called "class extension", which is distinguished from the class itself. Analog, every property is associated with a "property extension", which consists of pairs of individuals. The classes subsume the datatypes, and the properties subsume the data properties, the annotation properties, and the ontology properties. Individuals may play different roles at the same time in an OWL 2 Full ontology. One individual can, for example, be both a class and a property, or both a data property and an annotation property.

Editor's Note: It has been proposed to have some figure here visualizing the parts hierarchy and possibly the IEXT/ICEXT concept as explained in the text above.

In OWL 2 Full ontologies, usually no care is needed to ensure that URI references are actually in the appropriate part of the OWL universe. These "localizing" assumptions will typically follow from applying the OWL 2 Full semantic conditions.

A strong relationship holds between OWL 2 Full and the Direct Semantics of OWL 2 [*OWL 2 Direct Semantics*]. OWL 2 Full is, in a certain sense, able to reflect all logical conclusions of the Direct Semantics, when applied to an OWL 2 DL ontology [*OWL 2 Structural Specification*] in RDF graph form. The precise relationship is stated by the OWL 2 correspondence theorem.

The content of this document is not meant to be self-contained, but builds on top of the RDF Semantics document [*RDF Semantics*] by only adding the OWL 2 specific aspects of the semantics. Hence, the complete definition of OWL 2 Full is actually given by the combination of these two documents.

The italicized keywords *must*, *must not*, *should*, *should not*, and *may* specify certain aspects of the normative behavior of OWL 2 tools, and are interpreted as specified in RFC 2119 [*RFC 2119*].

2 Vocabulary

The *OWL 2 Full vocabulary* is a set of URI references in the OWL namespace, `owl:`, which is given by the URI reference

http://www.w3.org/2002/07/owl#

Table 2.1 lists the OWL 2 Full vocabulary, which extends the RDF and RDFS vocabulary as specified by Sections 3.1 and 4.1 of [*RDF Semantics*]. Excluded are those URI references from the OWL namespace, which are mentioned in one of the other tables in this section.

Table 2.1: OWL 2 Full Vocabulary

```
owl:AllDifferent owl:AllDisjointClasses
owl:AllDisjointProperties owl:allValuesFrom owl:Annotation
owl:AnnotationProperty owl:assertionProperty
owl:AsymmetricProperty owl:Axiom owl:backwardCompatibleWith
owl:bottomDataProperty owl:bottomObjectProperty
owl:cardinality owl:Class owl:complementOf owl:DataRange
owl:datatypeComplementOf owl:DatatypeProperty
owl:deprecated owl:DeprecatedClass owl:DeprecatedProperty
owl:differentFrom owl:disjointUnionOf owl:disjointWith
owl:distinctMembers owl:equivalentClass
owl:equivalentProperty owl:FunctionalProperty owl:hasKey
owl:hasSelf owl:hasValue owl:imports owl:incompatibleWith
owl:intersectionOf owl:InverseFunctionalProperty
owl:inverseOf owl:IrreflexiveProperty owl:maxCardinality
owl:maxQualifiedCardinality owl:members owl:minCardinality
owl:minQualifiedCardinality owl:NamedIndividual
owl:NegativePropertyAssertion owl:Nothing owl:object
owl:ObjectProperty owl:onClass owl:onDataRange
owl:onDatatype owl:oneOf owl:onProperty owl:onProperties
owl:Ontology owl:OntologyProperty owl:predicate
owl:priorVersion owl:propertyChain owl:propertyDisjointWith
owl:qualifiedCardinality owl:ReflexiveProperty
owl:Restriction owl:sameAs owl:someValuesFrom
owl:sourceIndividual owl:subject owl:SymmetricProperty
owl:targetIndividual owl:targetValue owl:Thing
owl:topDataProperty owl:topObjectProperty
owl:TransitiveProperty owl:unionOf owl:versionInfo
owl:withRestrictions
```

Note: The use of the URI reference `owl:DataRange` has been deprecated as of OWL 2. The URI reference `rdfs:Datatype` *should* be used instead.

Table 2.2 lists the set of *datatypes* of OWL 2 Full. `rdf:XMLLiteral` is described in Section 3.1 of [*RDF Semantics*]. `rdf:text` is described in [*RDF:TEXT*]. All other datatypes are described in Section 4 of [*OWL 2 Structural Specification*].

Table 2.2: Datatypes of OWL 2 Full

```
xsd:anyURI xsd:base64Binary xsd:boolean xsd:byte
owl:dateTime xsd:decimal xsd:double xsd:float xsd:hexBinary
xsd:int xsd:integer xsd:language xsd:long xsd:Name
xsd:NCName xsd:negativeInteger xsd:NMTOKEN
xsd:nonNegativeInteger xsd:nonPositiveInteger
xsd:normalizedString xsd:positiveInteger owl:rational
owl:real owl:realPlus xsd:short xsd:string rdf:text
xsd:token xsd:unsignedByte xsd:unsignedInt xsd:unsignedLong
xsd:unsignedShort rdf:XMLLiteral
```

Feature At Risk #1: *owl:rational* support

Note: This feature is "at risk" and may be removed from this specification based on feedback. Please send feedback to public-owl-comments@w3.org.

The *owl:rational* datatype might be removed from OWL 2 if implementation experience reveals problems with supporting this datatype.

Feature At Risk #2: *owl:dateTime* name

The name *owl:dateTime* is currently a placeholder. XML Schema 1.1 Working Group will introduce a datatype for date-time with required timezone. Once this is done, *owl:dateTime* will be changed to whatever name XML Schema chooses. If the schedule of the XML Schema 1.1 Working Group slips the OWL 2 Working Group will consider possible alternatives.

Please send feedback to public-owl-comments@w3.org.

Table 2.3 lists the set of *datatype facets* of OWL 2 Full. Section 4 of [*OWL 2 Structural Specification*] describes the meaning of each facet, to which datatypes it can be applied, and which values it can take for a given datatype. The facet `rdf:langPattern` is further described in [*RDF:TEXT*].

Table 2.3: Datatype Facets of OWL 2 Full

```
rdf:langPattern xsd:length xsd:maxExclusive
xsd:maxInclusive xsd:maxLength xsd:minExclusive
xsd:minInclusive xsd:minLength xsd:pattern
```

3 Ontologies

Editor's Note: Should this section on ontologies and importing be kept in this semantics document? And if so, what should be said about the semantics of an RDF graph having imports statements? Obviously, such an RDF graph has a clear semantic meaning in OWL 2 Full without regarding the imports closure. Perhaps we should state that in such a case the whole imports closure SHOULD be regarded as the ontology instead of the single RDF graph, with the semantic meaning being the one of that ontology.

Every well-formed RDF graph [*RDF*] is a syntactically valid OWL 2 Full ontology. If a OWL 2 Full ontology *imports* other OWL 2 Full ontologies, then the whole *imports closure* of that ontology has to be taken into account.

Definition 3.1 (Import Closure): Let K be a collection of RDF graphs. K is *imports closed* iff for every triple in any element of K of the form `x owl:imports u` then K contains a graph that is referred to by u. The *imports closure* of a collection of RDF graphs is the smallest imports closed collection of RDF graphs containing the graphs.

A OWL 2 Full ontology *may* contain an ontology header, if the ontology's author wants to explicitly signal that an RDF graph is intended as a OWL 2 Full ontology. Such an ontology header *may* additionally contain information about the ontology's version. The OWL 2 Mapping to RDF [*OWL 2 RDF Mapping*] provides details about the syntax of ontology headers.

4 Interpretations

OWL 2 Full provides a *vocabulary interpretation* and *vocabulary entailment* (see Section 2.1 of [*RDF Semantics*]) for the RDF and RDFS vocabularies, and the OWL 2 Full vocabulary.

From the RDF Semantics [*RDF Semantics*], let V be a set of URI references and literals containing the RDF and RDFS vocabulary, and let D be a datatype map according to Section 5.1 of [*RDF Semantics*]. A D-interpretation I of V is a tuple

$$I = \langle IR, IP, IEXT, IS, IL, LV \rangle.$$

IR is the domain of discourse or universe, i.e., a nonempty set that contains the denotations of URI references and literals in V. IP is a subset of IR, the properties of I. LV is a subset of IR that covers at least the value spaces of all datatypes in D. IEXT is used to associate properties with their property extension, and is a mapping from IP to $P(IR \times IR)$, where *P* is the *powerset*. IS is a mapping from URI references in V to their denotations in IR. IL is a mapping from typed literals in V to their denotations in IR, which maps all well-typed literals to instances of LV

(Section 5.1 of [*RDF Semantics*] explains why the range of IL is actually IR instead of LV).

As detailed in [*RDF Semantics*], a D-interpretation has to meet additional semantic conditions, which constrain the set of RDF graphs that are true under this interpretation. An RDF graph *G* is said to be *satisfied* by a D-interpretation *I*, if *I(G) = true*.

The following definition specifies what a OWL 2 Full datatype map is. First, Table 4.1 defines sets that relate datatypes with their facets, and with the values a facet is allowed to take in combination with a certain datatype.

Table 4.1: Sets that relate datatypes, facets and facet values

Name of Set S	Definition
`IFP(d)`	`The set of all facets allowed for datatype d.`
`IFV(d,f)`	`The set of all facet values allowed for the combination of datatype d and facet f.`
`IFEXT(d,f,u)`	`The subset of the class extension of datatype d that results from applying facet f with facet value u to d.`

Editor's Note: The sets given in this table need to be exchanged for sets that have counter parts in OWL 2 DL, in order to proof the Correspondence Theorem. The semantic conditions and comprehension principles for datatype restrictions have then to be adjusted as well.

Editor's Note: The definitions in this table should be less informal. They should be similar to the definitions given in the OWL 2 DL Specification.

Definition 4.1 (OWL 2 Full Datatype Map): Let D be a datatype map as defined in Section 5.1 of [*RDF Semantics*]. D is a *OWL 2 Full datatype map*, if it contains at least all datatypes listed in Table 2.2, and if it defines the sets listed in Table 4.1 for each contained datatype.

The next definition specifies what a OWL 2 Full interpretation is.

Definition 4.2 (OWL 2 Full Interpretation): Let D be a OWL 2 Full datatype map, and let V be a vocabulary that includes the RDF and RDFS vocabularies, and the OWL 2 Full vocabulary together with all the datatype and facet names listed in Section 2. An *OWL 2 Full interpretation*, I = 〈 IR, IP, IEXT, IS, IL, LV 〉, of V with respect to D, is a D-interpretation of V that satisfies all the extra semantic conditions given in Section 5.

Table 4.2 defines the "parts" of the OWL 2 Full universe in terms of the mapping IEXT of an OWL 2 Full interpretation and by referring to the RDF, RDFS and OWL 2 Full vocabularies.

Table 4.2: Parts of the OWL 2 Full Universe

Name of Part S	Definition of S as {x ∈ IR | ⟨x,I(U)⟩ ∈ IEXT(I(rdf:type))} where URI U is	Explanation
IR	rdfs:Resource	individuals
LV	rdfs:Literal	datatype values
IX	owl:Ontology	ontologies
IC	rdfs:Class	classes
IDC	rdfs:Datatype	datatypes
IP	rdf:Property	properties
IODP	owl:DatatypeProperty	data properties
IOAP	owl:AnnotationProperty	annotation properties
IOXP	owl:OntologyProperty	ontology properties

Further, the mapping ICEXT from IC to P(IR) that associates classes with their class extension, is defined as

$$\mathrm{ICEXT}(c) = \{ x \in \mathrm{IR} \mid \langle x,c \rangle \in \mathrm{IEXT}(\mathrm{I}(\mathrm{rdf:type})) \}$$

for $c \in \mathrm{IC}$.

The following definitions specify what a *consistent* OWL 2 Full ontology is, and what it means that an OWL 2 Full ontology *entails* another OWL 2 Full Ontology.

Definition 4.3 (OWL 2 Full Consistency): Let K be a collection of RDF graphs, and let D be a OWL 2 Full datatype map. K is *OWL 2 Full consistent* with respect to D iff there is some OWL 2 Full interpretation with respect to D (of some vocabulary that includes the RDF and RDFS vocabularies, and the OWL 2 Full vocabulary together with all the datatype and facet names listed in Section 2) that satisfies all the RDF graphs in K.

Definition 4.4 (OWL 2 Full Entailment): Let K and Q be collections of RDF graphs, and let D be a OWL 2 Full datatype map. K *OWL 2 Full entails* Q with respect to D iff every OWL 2 Full interpretation with respect to D (of any vocabulary V that includes the RDF and RDFS vocabularies, and the OWL 2 Full vocabulary

together with all the datatype and facet names listed in Section 2) that satisfies all the RDF graphs in K also satisfies all the RDF graphs in Q.

5 Semantic Conditions

This section defines the semantic conditions of OWL 2 Full. The semantic conditions presented here are only those for the specific features of OWL 2. The complete set of semantic conditions for OWL 2 Full is the combination of the semantic conditions presented here and the semantic conditions given for Simple Entailment, RDF, RDFS and D-Entailment in [*RDF Semantics*].

Table 5.1 specifies semantic conditions for the different parts of the OWL 2 Full universe, as defined in Section 4. Table 5.2 and Table 5.3 list semantic conditions for the classes and the properties of the OWL 2 Full vocabulary. The remaining tables in this section specify the OWL 2 Full semantic conditions for the different language features of OWL 2.

Most semantic conditions are "iff" conditions, which completely specify the semantics of the respective language feature. For some language features, however, there are only "if-then" conditions in order to avoid certain semantic paradoxes and other problems with the semantics. Several language features with "iff" conditions, namely Sub Property Chains in Table 5.9, N-ary Axioms in Table 5.11, and Negative Property Assertions in Table 5.15, have a multi-triple representation in RDF, where the different triples share a common "root node" *x*. In order to treat this specific syntactic aspect technically, the "iff" conditions of these language features have been split into two "if-then" conditions, and the right-to-left "if" condition contains an additional premise of the form "$\exists x \in IR$", which has the single purpose to provide the needed "root node" *x*.

Conventions used in this section:

Several conventions are used when presenting logic expressions in the below tables.

Having a *comma* between two assertions in a semantic condition, as in

$c \in IC$, $p \in IP$

means a logical "and".

If no scope is explicitly given for a variable *x*, as in "$\forall x{:}\ldots$" or in "$\{x|\ldots\}$", then x is unconstrained, which means that $x \in IR$.

An expression of the form "l sequence of $u_1,\ldots, u_n \in S$" means that l represents a list of n elements, all of them being instances of the class S. Precisely, $u_1 \in S,\ldots$, $u_n \in S$, and there exist $x_1 \in IR,\ldots, x_n \in IR$, such that

I(l) ∈ ICEXT(I(rdf:List)),
I(l) = I(x_1),
⟨x_1,u_1⟩ ∈ IEXT(I(rdf:first)), ⟨x_1,x_2⟩ ∈ IEXT(I(rdf:rest)),
…,
⟨x_n,u_n⟩ ∈ IEXT(I(rdf:first)), ⟨x_n,I(rdf:nil)⟩ ∈ IEXT(I(rdf:rest)).

The following names for certain sets are used as convenient abbreviations throughout this and the following sections:

- `ISEQ`: The set of all sequences. This set equals the class extension of rdf:List.
- `INNI`: The set of all non-negative integers. This set equals the value space of xsd:NonNegativeInteger, but is also contained in the value spaces of other numerical datatypes, such as xsd:integer.

The semantic conditions in the following tables sometimes do not explicitly list typing statements in their consequent that one would normally expect. For example, the semantic condition for owl:allValuesFrom restrictions in Table 5.6 does not list the statement *x* ∈ *ICEXT(I(`owl:Restriction`))* on its right hand side. Consequents are generally not mentioned, if they can already be deduced by means of the semantic conditions given in Table 5.2 and Table 5.3, occasionally in connection with Table 5.1. In the example above, the omitted consequent can be obtained from the third column of the entry for `owl:allValuesFrom` in Table 5.3, which determines that IEXT(I(owl:allValuesFrom)) ⊆ ICEXT(I(owl:Restriction)) × IC.

Table 5.1 lists the semantic conditions for the parts of the OWL 2 Full universe, as defined by Table 4.2 in Section 4. The semantic conditions say how the parts are related to other parts, and they further specify the semantics for the instances of some of the parts.

Table 5.1: Semantic Conditions for the Parts of the OWL 2 Full Universe

Name of Part S	Conditions on S	Conditions on Instances x of S
IR	S ≠ Ø	
LV	S ⊆ IR	
IX	S ⊆ IR	
IC	S ⊆ IR	ICEXT(x) ⊆ IR
IDC	S ⊆ IC	ICEXT(x) ⊆ LV
IP	S ⊆ IR	IEXT(x) ⊆ IR × IR
IODP	S ⊆ IP	IEXT(x) ⊆ IR × LV

IOAP	S ⊆ IP	IEXT(x) ⊆ IR × IR
IOXP	S ⊆ IP	IEXT(x) ⊆ IX × IX

Table 5.2 lists the semantic conditions for the classes of the OWL 2 Full vocabulary, and certain classes from RDF and RDFS. It tells the sort of class, and specifies the part of the OWL 2 Full universe the extension of each class belongs to. As a specific note: For `owl:NamedIndividual` that there is no way in OWL 2 Full to restrict the set of individuals to only those being named by a URI, hence the extension of this class has been specified to equal the whole domain.

Not included in this table are the *datatypes* of OWL 2 Full, as given in Table 2.2. For a datatype URI `U`, the following semantic conditions hold: `I(U) ∈ IDC`, and `ICEXT(I(U)) ⊆ LV`.

Table 5.2: Semantic Conditions for Classes

Vocabulary URI U	I(U)	ICEXT(I(U))
owl:AllDifferent	∈ IC	⊆ IR
owl:AllDisjointClasses	∈ IC	⊆ IR
owl:AllDisjointProperties	∈ IC	⊆ IR
owl:Annotation	∈ IC	⊆ IR
owl:AnnotationProperty	∈ IC	= IOAP
owl:AsymmetricProperty	∈ IC	⊆ IP
owl:Axiom	∈ IC	⊆ IR
rdfs:Class	∈ IC	= IC
owl:Class	∈ IC	= IC
owl:DataRange	∈ IC	= IDC
rdfs:Datatype	∈ IC	= IDC
owl:DatatypeProperty	∈ IC	= IODP
owl:DeprecatedClass	∈ IC	⊆ IC
owl:DeprecatedProperty	∈ IC	⊆ IP
owl:FunctionalProperty	∈ IC	⊆ IP
owl:InverseFunctionalProperty	∈ IC	⊆ IP

owl:IrreflexiveProperty	∈ IC	⊆ IP
rdfs:Literal	∈ IDC	= LV
owl:NamedIndividual	∈ IC	= IR
owl:NegativePropertyAssertion	∈ IC	⊆ IR
owl:Nothing	∈ IC	= ∅
owl:ObjectProperty	∈ IC	= IP
owl:Ontology	∈ IC	= IX
owl:OntologyProperty	∈ IC	= IOXP
rdf:Property	∈ IC	= IP
owl:ReflexiveProperty	∈ IC	⊆ IP
rdfs:Resource	∈ IC	= IR
owl:Restriction	∈ IC	⊆ IC
owl:SymmetricProperty	∈ IC	⊆ IP
owl:Thing	∈ IC	= IR
owl:TransitiveProperty	∈ IC	⊆ IP

Table 5.3 lists the semantic conditions for the properties of the OWL 2 Full vocabulary and certain properties from RDFS. It tells the sort of property, and specifies the domain and range for each property. As specific notes: `owl:topObjectProperty` relates every two individuals in the universe to each other. Likewise, `owl:topDataProperty` relates every individual to every datavalue. `owl:bottomObjectProperty` and `owl:bottomDataProperty` do not relate any individuals to each other at all. The ranges of the properties `owl:deprecated` and `owl:hasSelf` are not restricted to be boolean values, so it is possible for these properties to have objects of arbitrary type.

Not included in this table are the *datatype facets* of OWL 2 Full, as given in Table 2.3. For a facet URI `U`, the following semantic conditions hold: `I(U) ∈ IP`, and `IEXT(I(U)) ⊆ IR × LV`.

Table 5.3: Semantic Conditions for Properties

Vocabulary URI U	I(U)	IEXT(I(U))
owl:allValuesFrom	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IC

owl:assertionProperty	∈ IP	⊆ ICEXT(I(owl:NegativePropertyAssertion)) × IP
owl:backwardCompatibleWith	∈ IOXP	⊆ IX × IX
owl:bottomDataProperty	∈ IODP	= Ø
owl:bottomObjectProperty	∈ IP	= Ø
owl:cardinality	∈ IP	⊆ ICEXT(I(owl:Restriction)) × INNI
rdfs:comment	∈ IOAP	⊆ IR × LV
owl:complementOf	∈ IP	⊆ IC × IC
owl:datatypeComplementOf	∈ IP	⊆ IDC × IDC
owl:deprecated	∈ IOAP	⊆ IR × IR
owl:differentFrom	∈ IP	⊆ IR × IR
owl:disjointUnionOf	∈ IP	⊆ IC × ISEQ
owl:disjointWith	∈ IP	⊆ IC × IC
owl:distinctMembers	∈ IP	⊆ ICEXT(I(owl:AllDifferent)) × ISEQ
owl:equivalentClass	∈ IP	⊆ IC × IC
owl:equivalentProperty	∈ IP	⊆ IP × IP
owl:hasKey	∈ IP	⊆ IC × ISEQ

owl:hasSelf	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IR
owl:hasValue	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IR
owl:imports	∈ IOXP	⊆ IX × IX
owl:incompatibleWith	∈ IOXP	⊆ IX × IX
owl:intersectionOf	∈ IP	⊆ IC × ISEQ
owl:inverseOf	∈ IP	⊆ IP × IP
rdfs:isDefinedBy	∈ IOAP	⊆ IR × IR
rdfs:label	∈ IOAP	⊆ IR × LV
owl:maxCardinality	∈ IP	⊆ ICEXT(I(owl:Restriction)) × INNI
owl:maxQualifiedCardinality	∈ IP	⊆ ICEXT(I(owl:Restriction)) × INNI
owl:members	∈ IP	⊆ IR × ISEQ
owl:minCardinality	∈ IP	⊆ ICEXT(I(owl:Restriction)) × INNI
owl:minQualifiedCardinality	∈ IP	⊆ ICEXT(I(owl:Restriction)) × INNI
owl:object	∈ IP	⊆ IR × IR
owl:onClass	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IC
owl:onDataRange	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IDC
owl:onDatatype	∈ IP	⊆ IDC × IDC

owl:oneOf	∈ IP	⊆ IC × ISEQ
owl:onProperty	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IP
owl:onProperties	∈ IP	⊆ ICEXT(I(owl:Restriction)) × ISEQ
owl:predicate	∈ IP	⊆ IR × IP
owl:priorVersion	∈ IOXP	⊆ IX × IX
owl:propertyChain	∈ IP	⊆ IP × ISEQ
owl:propertyDisjointWith	∈ IP	⊆ IP × IP
owl:qualifiedCardinality	∈ IP	⊆ ICEXT(I(owl:Restriction)) × INNI
owl:sameAs	∈ IP	⊆ IR × IR
rdfs:seeAlso	∈ IOAP	⊆ IR × IR
owl:someValuesFrom	∈ IP	⊆ ICEXT(I(owl:Restriction)) × IC
owl:sourceIndividual	∈ IP	⊆ ICEXT(I(owl:NegativePropertyAssertion)) × IR
owl:subject	∈ IP	⊆ IR × IR
owl:targetIndividual	∈ IP	⊆ ICEXT(I(owl:NegativePropertyAssertion)) × IR
owl:targetValue	∈ IP	⊆ ICEXT(I(owl:NegativePropertyAssertion)) × LV
owl:topDataProperty	∈ IODP	= IR × LV

owl:topObjectProperty	∈ IP	= IR × IR
owl:unionOf	∈ IP	⊆ IC × ISEQ
owl:versionInfo	∈ IOAP	⊆ IR × IR
owl:withRestrictions	∈ IP	⊆ IDC × ISEQ

Table 5.4 lists the semantic conditions for boolean class expressions, including complements, intersections, and unions of classes. An intersection or union of a collection of datatypes is itself a datatype. While a complement of a class is created w.r.t. to the whole domain, a datatype complement is created for a datatype w.r.t. the set of data values only, and results itself in a datatype.

Table 5.4: Semantic Conditions for Boolean Class Expressions

⟨c,d⟩ ∈ IEXT(I(owl:complementOf))	**iff**	c, d ∈ IC ICEXT(c) = IR \ ICEXT(d)
⟨c,d⟩ ∈ IEXT(I(owl:datatypeComplementOf))		c, d ∈ IDC, ICEXT(c) = LV \ ICEXT(d)
if l sequence of d_1,…, d_n ∈ IR then		
⟨c,l⟩ ∈ IEXT(I(owl:intersectionOf))	**iff**	c, d_1,…, d_n ∈ IC, ICEXT(c) = ICEXT(d_1) ∩…∩ ICEXT(d_n)
⟨c,l⟩ ∈ IEXT(I(owl:unionOf))		c, d_1,…, d_n ∈ IC, ICEXT(c) = ICEXT(d_1) ∪…∪ ICEXT(d_n)
if		**then**
l sequence of d_1,…, d_n ∈ IDC, n ≥ 1, ⟨c,l⟩ ∈ IEXT(I(owl:intersectionOf))		c ∈ IDC
l sequence of d_1,…, d_n ∈ IDC, n ≥ 1, ⟨c,l⟩ ∈ IEXT(I(owl:unionOf))		c ∈ IDC

Table 5.5 lists the semantic conditions for enumerations, i.e. classes that consist of an explicitly given finite set of instances. In particular, an enumeration entirely consisting of datatype values is a datatype.

Table 5.5: Semantic Conditions for Enumerations

if l sequence of $u_1,\ldots, u_n \in$ IR then		
$\langle c,l\rangle \in$ IEXT(I(owl:oneOf))	**iff**	$c \in$ IC, ICEXT(c) = { $u_1,\ldots, u_n$ }
if		**then**
l sequence of $u_1,\ldots, u_n \in$ LV, $n \geq 1$, $\langle c,l\rangle \in$ IEXT(I(owl:oneOf))		$c \in$ IDC

Table 5.6 lists the semantic conditions for property restrictions, including value restrictions, cardinality restrictions, and self restrictions. There are also semantic conditions for value restrictions dealing with n-ary datatypes. Note that the semantic condition for self restrictions does not entail the right hand side of a `owl:hasSelf` assertion to be a boolean value, so it is possible to have right hand sides of arbitrary type.

Table 5.6: Semantic Conditions for Property Restrictions

if	**then**
$\langle x,u\rangle \in$ IEXT(I(owl:hasSelf)), $\langle x,p\rangle \in$ IEXT(I(owl:onProperty))	ICEXT(x) = {y | $\langle y,y\rangle \in$ IEXT(p)}
$\langle x,c\rangle \in$ IEXT(I(owl:allValuesFrom)), $\langle x,p\rangle \in$ IEXT(I(owl:onProperty))	ICEXT(x) = {y | $\forall z : \langle y,z\rangle \in$ IEXT(p) $\rightarrow z \in$ ICEXT(c)}
l sequence of $p_1,\ldots, p_n \in$ IR, $\langle x,c\rangle \in$ IEXT(I(owl:allValuesFrom)), $\langle x,l\rangle \in$ IEXT(I(owl:onProperties))	$p_1,\ldots, p_n \in$ IODP, $c \in$ IDC, ICEXT(x) = {y | $\forall z_1,\ldots,z_n \in$ LV : $\langle y,z_1\rangle \in$ IEXT(p_1) $\wedge\ldots\wedge \langle y,z_n\rangle \in$ IEXT(p_n) $\rightarrow \langle z_1,\ldots,z_n\rangle \in$ ICEXT(c)}
$\langle x,c\rangle \in$ IEXT(I(owl:someValuesFrom)), $\langle x,p\rangle \in$ IEXT(I(owl:onProperty))	ICEXT(x) = {y | $\exists z : \langle y,z\rangle \in$ IEXT(p) $\wedge z \in$ ICEXT(c)}
l sequence of $p_1,\ldots, p_n \in$ IR, $\langle x,c\rangle \in$ IEXT(I(owl:someValuesFrom)), $\langle x,l\rangle \in$ IEXT(I(owl:onProperties))	$p_1,\ldots, p_n \in$ IODP, $c \in$ IDC, ICEXT(x) = {y | $\exists z_1,\ldots,z_n \in$ LV : $\langle y,z_1\rangle \in$ IEXT(p_1)

	$\wedge \ldots \wedge \langle y, z_n \rangle \in IEXT(p_n) \wedge \langle z_1, \ldots, z_n \rangle \in ICEXT(c)\}$
$\langle x,u \rangle \in IEXT(I(owl:hasValue)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \langle y,u \rangle \in IEXT(p)\}$
$\langle x,n \rangle \in IEXT(I(owl:cardinality)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \#\{z \mid \langle y,z \rangle \in IEXT(p)\} = n\}$
$\langle x,n \rangle \in IEXT(I(owl:minCardinality)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \#\{z \mid \langle y,z \rangle \in IEXT(p)\} \geq n\}$
$\langle x,n \rangle \in IEXT(I(owl:maxCardinality)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \#\{z \mid \langle y,z \rangle \in IEXT(p)\} \leq n\}$
$\langle x,n \rangle \in IEXT(I(owl:qualifiedCardinality)), \langle x,c \rangle \in IEXT(I(owl:onClass)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \#\{z \mid \langle y,z \rangle \in IEXT(p) \wedge z \in ICEXT(c)\} = n\}$
$\langle x,n \rangle \in IEXT(I(owl:qualifiedCardinality)), \langle x,c \rangle \in IEXT(I(owl:onDataRange)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$p \in IODP, ICEXT(x) = \{y \mid \#\{z \in LV \mid \langle y,z \rangle \in IEXT(p) \wedge z \in ICEXT(c)\} = n\}$
$\langle x,n \rangle \in IEXT(I(owl:minQualifiedCardinality)), \langle x,c \rangle \in IEXT(I(owl:onClass)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \#\{z \mid \langle y,z \rangle \in IEXT(p) \wedge z \in ICEXT(c)\} \geq n\}$
$\langle x,n \rangle \in IEXT(I(owl:minQualifiedCardinality)), \langle x,c \rangle \in IEXT(I(owl:onDataRange)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$p \in IODP, ICEXT(x) = \{y \mid \#\{z \in LV \mid \langle y,z \rangle \in IEXT(p) \wedge z \in ICEXT(c)\} \geq n\}$
$\langle x,n \rangle \in IEXT(I(owl:maxQualifiedCardinality)), \langle x,c \rangle \in IEXT(I(owl:onClass)), \langle x,p \rangle \in IEXT(I(owl:onProperty))$	$ICEXT(x) = \{y \mid \#\{z \mid \langle y,z \rangle \in IEXT(p) \wedge z \in ICEXT(c)\} \leq n\}$
$\langle x,n \rangle \in IEXT(I(owl:maxQualifiedCardinality)),$	$p \in IODP, ICEXT(x) = \{y \mid \#\{z \in LV \mid \langle y,z \rangle \in IEXT(p)$

⟨x,c⟩ ∈ IEXT(I(owl:onDataRange)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))	∧ z ∈ ICEXT(c)} ≤ n}

Editor's Note: There is an open issue with OWL 2 Full's support of n-ary datatypes. I suppose that we need to introduce a new sort of extension, a "datarange extension of arity n", which associates an individual with a set of n-tuples of individuals. Affected parts of the document would be: Interpretations, the Semantic Conditions and the Comprehension Principles for the Datatype Complement, and the Restrictions on N-ary Datatypes ("owl:onProperties"). Should this be the case, then the latter semantic conditions have to be considered broken at the moment, since n-tuples of individuals then cannot be instances of class extensions.

However, it has been proposed that there is no need to introduce such an additional sort of extension.

Table 5.7 lists the semantic conditions for datatype restrictions, which are specified for a datatype, and for a set of facets and facet values. Note that if no facet is applied to a given datatype, then the resulting datatype will be equivalent to the original datatype. Note further that the semantic conditions are specified in a way that applying a facet to a datatype, for which it is not defined, will lead to an unsatisfiable ontology. Likewise, adding an inapplicable facet value to a certain combination of a datatype a facet will lead to an unsatisfiable ontology. As a consequence, a datatype restriction with one or more specified facets will lead to an unsatisfiable ontology if applied to a datatype for which no facets are defined (usually a set of facets only exists for datatypes contained in the datatype map).

Table 5.7: Semantic Conditions for Datatype Restrictions

if	then
l sequence of $y_1,\ldots, y_n \in IR$, $f_1,\ldots, f_n \in IP$, ⟨c,d⟩ ∈ IEXT(I(owl:onDatatype)), ⟨c,l⟩ ∈ IEXT(I(owl:withRestrictions)), $\langle y_1,u_1\rangle \in IEXT(f_1)$, …, $\langle y_n,u_n\rangle \in IEXT(f_n)$	c, d ∈ IDC, $f_i \in IFP(d)$ for $1\leq i\leq n$, $u_i \in IFV(d,f_i)$ for $1\leq i\leq n$, $ICEXT(c) = ICEXT(d) \cap IFEXT(d,f_1,u_1) \cap\ldots\cap IFEXT(d,f_n,u_n)$

Table 5.8 *extends* the semantic conditions for the RDFS vocabulary. The original semantics for the language features regarded here are specified in [*RDF Semantics*], and they only provide for "if-then" semantic conditions, while OWL 2 Full specifies stronger "iff" semantic conditions. Note that only the additional semantic conditions are given here and that the other conditions on the RDF and RDFS vocabularies are retained.

Table 5.8: Extended Semantic Conditions for the RDFS Vocabulary

$\langle c,d \rangle \in$ IEXT(I(rdfs:subClassOf))		$c, d \in IC$, ICEXT(c) $\subseteq$ ICEXT(d)
$\langle p,q \rangle \in$ IEXT(I(rdfs:subPropertyOf))		$p, q \in IP$, IEXT(p) $\subseteq$ IEXT(q)
$\langle p,c \rangle \in$ IEXT(I(rdfs:domain))	**iff**	$p \in IP$, $c \in IC$, $\forall x,y : \langle x,y \rangle \in$ IEXT(p) $\rightarrow x \in$ ICEXT(c)
$\langle p,c \rangle \in$ IEXT(I(rdfs:range))		$p \in IP$, $c \in IC$, $\forall x,y : \langle x,y \rangle \in$ IEXT(p) $\rightarrow y \in$ ICEXT(c)

Table 5.9 lists the semantic conditions for sub property chains. The semantics have been specified in a way to allow a sub property chain axiom to be satisfiable without requiring the existence of a property that represents the property chain. In particular, the property on the left hand side of the sub property assertion does not necessarily represent the property chain.

Table 5.9: Semantic Conditions for Sub Property Chains

if	**then**
l sequence of $p_1,\ldots, p_n \in IR$, $\langle x,q \rangle \in$ IEXT(I(rdfs:subPropertyOf)), $\langle x,l \rangle \in$ IEXT(I(owl:propertyChain))	$p_1,\ldots, p_n \in IP$, $q \in IP$, $\forall y_0,\ldots,y_n : \langle y_0,y_1 \rangle \in$ IEXT(p_1) $\wedge\ldots\wedge \langle y_{n-1},y_n \rangle \in$ IEXT(p_n) $\rightarrow \langle y_0,y_n \rangle \in$ IEXT(q)
if	**then exists x $\in$ IR**
l sequence of $p_1,\ldots, p_n \in IP$, $q \in IP$, $\forall y_0,\ldots,y_n : \langle y_0,y_1 \rangle \in$ IEXT(p_1) $\wedge\ldots\wedge \langle y_{n-1},y_n \rangle \in$ IEXT(p_n) $\rightarrow \langle y_0,y_n \rangle \in$ IEXT(q)	$\langle x,q \rangle \in$ IEXT(I(rdfs:subPropertyOf)), $\langle x,l \rangle \in$ IEXT(I(owl:propertyChain))

Table 5.10 lists the semantic conditions for equal and different individuals, equivalent and disjoint classes, and equivalent and disjoint properties. Also treated here are disjoint union axioms.

Table 5.10: Semantic Conditions for Equivalence and Disjointness Axioms

$\langle u,w\rangle \in IEXT(I(owl:sameAs))$	**iff**	$u = w$
$\langle u,w\rangle \in IEXT(I(owl:differentFrom))$		$u \neq w$
$\langle c,d\rangle \in IEXT(I(owl:equivalentClass))$		$c, d \in IC, ICEXT(c) = ICEXT(d)$
$\langle c,d\rangle \in IEXT(I(owl:disjointWith))$		$c, d \in IC, ICEXT(c) \cap ICEXT(d) = \emptyset$
$\langle p,q\rangle \in IEXT(I(owl:equivalentProperty))$		$p, q \in IP, IEXT(p) = IEXT(q)$
$\langle p,q\rangle \in IEXT(I(owl:propertyDisjointWith))$		$p, q \in IP, IEXT(p) \cap IEXT(q) = \emptyset$
if l sequence of $d_1,\ldots, d_n \in IR$ then		
$\langle c,l\rangle \in IEXT(I(owl:disjointUnionOf))$	**iff**	$c, d_1,\ldots, d_n \in IC, ICEXT(c) = ICEXT(d_1) \cup\ldots\cup ICEXT(d_n), ICEXT(d_i) \cap ICEXT(d_k) = \emptyset$ for $1 \leq i \neq k \leq n$

Table 5.11 lists the semantic conditions for n-ary axioms on different individuals, disjoint classes, and disjoint properties. Note that there are two alternative ways to specify `owl:AllDifferent` axioms, both of them having the same model-theoretic meaning.

Table 5.11: Semantic Conditions for N-ary Axioms

if	then
l sequence of $u_1,\ldots, u_n \in IR$, $x \in ICEXT(I(owl:AllDifferent))$, $\langle x,l\rangle \in IEXT(I(owl:distinctMembers))$	$u_i \neq u_k$ for $1 \leq i \neq k \leq n$
l sequence of $u_1,\ldots, u_n \in IR$, $x \in ICEXT(I(owl:AllDifferent))$, $\langle x,l\rangle \in IEXT(I(owl:members))$	$u_i \neq u_k$ for $1 \leq i \neq k \leq n$
l sequence of $c_1,\ldots, c_n \in IR$, $x \in$	$c_1,\ldots, c_n \in IC$, $ICEXT(c_i) \cap ICEXT(c_k) = \emptyset$ for $1 \leq i \neq k \leq n$

ICEXT(I(owl:AllDisjointClasses)), ⟨x,l⟩ ∈ IEXT(I(owl:members))	
l sequence of $p_1,\ldots, p_n \in IR$, x ∈ ICEXT(I(owl:AllDisjointProperties)), ⟨x,l⟩ ∈ IEXT(I(owl:members))	$p_1,\ldots, p_n \in IP$, $IEXT(p_i) \cap IEXT(p_k) = \emptyset$ for $1 \leq i \neq k \leq n$
if	**then exists x ∈ IR**
l sequence of $u_1,\ldots, u_n \in IR$, $u_i \neq u_k$ for $1 \leq i \neq k \leq n$	x ∈ ICEXT(I(owl:AllDifferent)), ⟨x,l⟩ ∈ IEXT(I(owl:distinctMembers))
l sequence of $u_1,\ldots, u_n \in IR$, $u_i \neq u_k$ for $1 \leq i \neq k \leq n$	x ∈ ICEXT(I(owl:AllDifferent)), ⟨x,l⟩ ∈ IEXT(I(owl:members))
l sequence of $c_1,\ldots, c_n \in IC$, $ICEXT(c_i) \cap ICEXT(c_k) = \emptyset$ for $1 \leq i \neq k \leq n$	x ∈ ICEXT(I(owl:AllDisjointClasses)), ⟨x,l⟩ ∈ IEXT(I(owl:members))
l sequence of $p_1,\ldots, p_n \in IP$, $IEXT(p_i) \cap IEXT(p_k) = \emptyset$ for $1 \leq i \neq k \leq n$	x ∈ ICEXT(I(owl:AllDisjointProperties)), ⟨x,l⟩ ∈ IEXT(I(owl:members))

Table 5.12 lists the semantic conditions for inverse property axioms.

Table 5.12: Semantic Conditions for Inverse Property Axioms

⟨p,q⟩ ∈ IEXT(I(owl:inverseOf))	**iff**	p, q ∈ IP, IEXT(p) = {⟨x,y⟩ | ⟨y,x⟩ ∈ IEXT(q)}

Table 5.13 lists the semantic conditions for property characteristics, i.e. functionality and inverse functionality, reflexivity and irreflexivity, symmetry and asymmetry, and transitivity of properties.

Table 5.13: Semantic Conditions for Property Characteristics

p ∈ ICEXT(I(owl:FunctionalProperty))	iff	p ∈ IP, ∀x,y,z : ⟨x,y⟩, ⟨x,z⟩ ∈ IEXT(p) → y = z
p ∈ ICEXT(I(owl:InverseFunctionalProperty))		p ∈ IP, ∀x,y,z : ⟨y,x⟩, ⟨z,x⟩ ∈ IEXT(p) → y = z
p ∈ ICEXT(I(owl:ReflexiveProperty))		p ∈ IP, ∀x : ⟨x,x⟩ ∈ IEXT(p)
p ∈ ICEXT(I(owl:IrreflexiveProperty))		p ∈ IP, ∀x : ⟨x,x⟩ ∉ IEXT(p)
p ∈ ICEXT(I(owl:SymmetricProperty))		p ∈ IP, ∀x,y : ⟨x,y⟩ ∈ IEXT(p) → ⟨y,x⟩ ∈ IEXT(p)
p ∈ ICEXT(I(owl:AsymmetricProperty))		p ∈ IP, ∀x,y : ⟨x,y⟩ ∈ IEXT(p) → ⟨y,x⟩ ∉ IEXT(p)
p ∈ ICEXT(I(owl:TransitiveProperty))		p ∈ IP, ∀x,y,z : ⟨x,y⟩, ⟨y,z⟩ ∈ IEXT(p) → ⟨x,z⟩ ∈ IEXT(p)

Table 5.14 lists the semantic conditions for Keys. Keys are an alternative to inverse functional properties (see Table 5.13). They provide for compound keys, and they allow to specify the class of individuals for which a property plays the role of a key feature.

Table 5.14: Semantic Conditions for Keys

if I sequence of $p_1,\ldots, p_n \in$ IR then		
⟨c,l⟩ ∈ IEXT(I(owl:hasKey))	iff	c ∈ IC, $p_1,\ldots, p_n \in$ IP, $\forall x,y,z_1,\ldots,z_n$: x, y ∈ ICEXT(c), ⟨x,z_i⟩, ⟨y,z_i⟩ ∈ IEXT(p_i), 1 ≤

		i ≤ n → x = y

Table 5.15 lists the semantic conditions for negative property assertions. They allow to state that an individual *u* does *not* stand in a relationship *p* with another individual *w*. The second form based on `owl:targetValue` is more specific than the first form based on `owl:targetIndividual` in that it is restricted to the case of negative *data* property assertions. Note that the second form will coerce the target individual of a negative property assertion into a data value, due to the range defined for the property `owl:targetValue` in Table 5.3.

Table 5.15: Semantic Conditions for Negative Property Assertions

if	then
⟨x,u⟩ ∈ IEXT(I(owl:sourceIndividual)), ⟨x,p⟩ ∈ IEXT(I(owl:assertionProperty)), ⟨x,w⟩ ∈ IEXT(I(owl:targetIndividual))	⟨u,w⟩ ∉ IEXT(p)
⟨x,u⟩ ∈ IEXT(I(owl:sourceIndividual)), ⟨x,p⟩ ∈ IEXT(I(owl:assertionProperty)), ⟨x,w⟩ ∈ IEXT(I(owl:targetValue))	p ∈ IODP, ⟨u,w⟩ ∉ IEXT(p)
if	**then exists x ∈ IR**
u ∈ IR, p ∈ IP, w ∈ IR, ⟨u,w⟩ ∉ IEXT(p)	⟨x,u⟩ ∈ IEXT(I(owl:sourceIndividual)), ⟨x,p⟩ ∈ IEXT(I(owl:assertionProperty)), ⟨x,w⟩ ∈ IEXT(I(owl:targetIndividual))
u ∈ IR, p ∈ IODP, w ∈ LV, ⟨u,w⟩ ∉ IEXT(p)	⟨x,u⟩ ∈ IEXT(I(owl:sourceIndividual)), ⟨x,p⟩ ∈ IEXT(I(owl:assertionProperty)), ⟨x,w⟩ ∈ IEXT(I(owl:targetValue))

6 Relationship to OWL 2 DL

This section is concerned with a strong relationship that holds between OWL 2 Full and the Direct Semantics of OWL 2 [*OWL 2 Direct Semantics*].

6.1 A Difference to the Direct Semantics (Informative)

One design goal of OWL 2 has been that OWL 2 Full should reflect every logical consequence of the Direct Semantics of OWL 2 [*OWL 2 Direct Semantics*], as long as this consequence and all its premises can be represented as valid OWL 2 DL ontologies in RDF graph form. However, a fundamental semantic difference exists between the Direct Semantics and OWL 2 Full, which complicates a comparison of their semantic expressiveness. The Direct Semantics treats classes as *sets*, i.e. subsets of the universe. Classes in OWL 2 Full, however, are *individuals* in the universe, which have such a set associated to them as their class extension. Hence, under OWL 2 Full, all classes are instances of the universe, but this cannot generally be assumed under the Direct Semantics. An analog distinction holds for properties.

An effect of this difference is that certain logical conclusions of OWL 2 DL do not become "visible" under OWL 2 Full, although they are reflected by OWL 2 Full at a set theoretical level. For example, consider the following two RDF graphs G_1 and G_2 (RDF graphs are presented here in the style used in [*OWL 2 RDF Mapping*]):

G_1 := {

```
ex:C rdf:type owl:Class .
ex:D rdf:type owl:Class .
ex:C rdfs:subClassOf ex:D .
```

}

G_2 := {

```
ex:C rdf:type owl:Class .
ex:D rdf:type owl:Class .
_:x owl:intersectionOf (SEQ ex:C ex:D) .
_:x rdfs:subClassOf ex:D .
```

}

Both graphs are OWL 2 DL ontologies in RDF graph form, and G_1 entails G_2 under the Direct Semantics. However, under OWL 2 Full this entailment does not hold. Actually, OWL 2 Full interprets G_1 in a way such that the set theoretical relationship

$$ICEXT(I(ex:C)) \cap ICEXT(I(ex:D)) \subseteq ICEXT(I(ex:D))$$

can be concluded. But since OWL 2 Full distinguishes between classes as individuals and their class extensions being the actual sets, G_2 is not entailed, unless there exists some additional "helper" individual w, having the set S, defined by

S := ICEXT(w) = ICEXT(I(ex:C)) ∩ ICEXT(I(ex:D))

as its class extension. Whether such a helper individual exists or not has no effect on the answer to the question, whether the basic logical conclusion at the set theoretical level holds or not. The individual is, however, required to represent this conclusion as the RDF graph G_2.

The following subsection introduces a set of *"comprehension principles"*, which have the purpose to provide the missing "helper" individuals.

6.2 Comprehension Principles

This section lists the set of comprehension principles of OWL 2 Full. These comprehension principles are *not* part of the set of semantic conditions given in Section 5, and therefore do not need to be met by a OWL 2 Full interpretation as defined in Section 4. They are, however, needed for the *OWL 2 correspondence theorem* (see Section 6.3) to hold, since the correspondence theorem compares OWL 2 Full and the Direct Semantics solely based on entailments.

Table 6.1 lists the comprehension principles for sequences, i.e. RDF lists build from any finite combination of individuals.

Table 6.1: Comprehension Principles for Sequences

if	then exists $x_1,\ldots, x_n \in$ IR
$u_1,\ldots, u_n \in$ IR	$\langle x_1,u_1\rangle \in$ IEXT(I(rdf:first)), $\langle x_1,x_2\rangle \in$ IEXT(I(rdf:rest)), …, $\langle x_n,u_n\rangle \in$ IEXT(I(rdf:first)), $\langle x_n,$I(rdf:nil)$\rangle \in$ IEXT(I(rdf:rest))

Table 6.2 lists the comprehension principles for boolean class expressions, including complements, intersections, and unions of classes.

Table 6.2: Comprehension Principles for Boolean Class Expressions

if	then exists x ∈ IR
c ∈ IC	⟨x,c⟩ ∈ IEXT(I(owl:complementOf))
c ∈ IDC	⟨x,c⟩ ∈ IEXT(I(owl:datatypeComplementOf))

l sequence of $c_1,\ldots, c_n$ ∈ IC	⟨x,l⟩ ∈ IEXT(I(owl:intersectionOf))
l sequence of $c_1,\ldots, c_n$ ∈ IC	⟨x,l⟩ ∈ IEXT(I(owl:unionOf))

Table 6.3 lists the comprehension principles for enumerations, i.e. classes that consist of an explicitly given finite set of instances.

Table 6.3: Comprehension Principles for Enumerations

if	then exists x ∈ IR
l sequence of $u_1,\ldots, u_n$ ∈ IR	⟨x,l⟩ ∈ IEXT(I(owl:oneOf))

Table 6.4 lists the comprehension principles for property restrictions, including value restrictions, cardinality restrictions, and self restrictions. There are also comprehension principles for value restrictions dealing with n-ary datatypes.

Table 6.4: Comprehension Principles for Property Restrictions

if	then exists x ∈ IR
p ∈ IP	⟨x,I("true"^^xsd:boolean)⟩ ∈ IEXT(I(owl:hasSelf)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
c ∈ IC, p ∈ IP	⟨x,c⟩ ∈ IEXT(I(owl:allValuesFrom)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
c ∈ IDC, l sequence of $p_1,\ldots,$ p_n ∈ IODP	⟨x,c⟩ ∈ IEXT(I(owl:allValuesFrom)), ⟨x,l⟩ ∈ IEXT(I(owl:onProperties))
c ∈ IC, p ∈ IP	⟨x,c⟩ ∈ IEXT(I(owl:someValuesFrom)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
c ∈ IDC, l sequence of $p_1,\ldots,$ p_n ∈ IODP	⟨x,c⟩ ∈ IEXT(I(owl:someValuesFrom)), ⟨x,l⟩ ∈ IEXT(I(owl:onProperties))
u ∈ IR, p ∈ IP	⟨x,u⟩ ∈ IEXT(I(owl:hasValue)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, p ∈ IP	⟨x,n⟩ ∈ IEXT(I(owl:cardinality)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, p ∈ IP	⟨x,n⟩ ∈ IEXT(I(owl:minCardinality)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))

n ∈ INNI, p ∈ IP	⟨x,n⟩ ∈ IEXT(I(owl:maxCardinality)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, c ∈ IC, p ∈ IP	⟨x,n⟩ ∈ IEXT(I(owl:qualifiedCardinality)), ⟨x,c⟩ ∈ IEXT(I(owl:onClass)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, c ∈ IDC, p ∈ IODP	⟨x,n⟩ ∈ IEXT(I(owl:qualifiedCardinality)), ⟨x,c⟩ ∈ IEXT(I(owl:onDataRange)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, c ∈ IC, p ∈ IP	⟨x,n⟩ ∈ IEXT(I(owl:minQualifiedCardinality)), ⟨x,c⟩ ∈ IEXT(I(owl:onClass)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, c ∈ IDC, p ∈ IODP	⟨x,n⟩ ∈ IEXT(I(owl:minQualifiedCardinality)), ⟨x,c⟩ ∈ IEXT(I(owl:onDataRange)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, c ∈ IC, p ∈ IP	⟨x,n⟩ ∈ IEXT(I(owl:maxQualifiedCardinality)), ⟨x,c⟩ ∈ IEXT(I(owl:onClass)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))
n ∈ INNI, c ∈ IDC, p ∈ IODP	⟨x,n⟩ ∈ IEXT(I(owl:maxQualifiedCardinality)), ⟨x,c⟩ ∈ IEXT(I(owl:onDataRange)), ⟨x,p⟩ ∈ IEXT(I(owl:onProperty))

Table 6.5 lists the comprehension principles for datatype restrictions, which are specified for a datatype, and for a set of facets and facet values.

Table 6.5: Comprehension Principles for Datatype Restrictions

if	then exists x ∈ IR, l sequence of $y_1,\ldots,y_n$ ∈ IR
c ∈ IDC, $f_1,\ldots,f_n$ facets, $u_1,\ldots,u_n$ ∈ LV	⟨x,c⟩ ∈ IEXT(I(owl:onDatatype)), ⟨x,l⟩ ∈ IEXT(I(owl:withRestrictions)), $\langle y_1,u_1\rangle$ ∈ IEXT(f_1), …, $\langle y_n,u_n\rangle$ ∈ IEXT(f_n)

6.3 Correspondence Theorem

This section presents the OWL 2 correspondence theorem.

Theorem 6.1 (Correspondence Theorem): Let D be a OWL 2 Full datatype map, and let K and Q be collections of valid OWL 2 DL ontologies in RDF graph form that are imports closed, and without annotations occurring in Q. Let F(K) and F(Q) be the collections of OWL 2 DL ontologies in Functional Syntax that result from applying the reverse RDF mapping [*OWL 2 RDF Mapping*] to K and Q, respectively. If F(K) entails F(Q) with respect to the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] and with respect to D, then K entails Q with respect to OWL 2 Full extended by the comprehension principles, and with respect to D.

> **Editor's Note:** In the given form, the theorem is trivially true, since combining the set of all OWL 2 Full semantic conditions with all the comprehension principles leads to inconsistency. Further, in this form the theorem does not give very useful hints to implementers on how to best exploit the set of comprehension principles in order to build good entailment checkers that also cover all of OWL 2 DL. Currently, an alternative formulation of the theorem is under investigation.

A sketch of a proof for this theorem is given in Appendix B.

7 Appendix A: Axiomatic Triples (Informative)

The RDF Semantics document [*RDF Semantics*] defines so called "axiomatic triples" for the RDF and RDFS vocabularies. Examples of axiomatic triples are:

```
rdf:type rdf:type rdf:Property .
rdf:type rdfs:domain rdfs:Resource .
rdf:type rdfs:range rdfs:Class .
```

Axiomatic triples are used to give certain basic semantic meaning to all the URIs in the RDF and RDFS vocabularies. This semantic meaning is meant to be "axiomatic", in the sense that it holds for *every* ontology of the regarded language, including the empty ontology.

Typically, as shown by the examples above, axiomatic triples are used in the RDF Semantics to specify the part of the universe that the denotation of a vocabulary URI belongs to. In the case of properties of the regarded vocabulary, also the domains and the ranges are specified. These kinds of axiomatic triples can be equivalently restated in the form of semantic conditions that have neither premises nor bound variables. Using the names of the different parts of the universe, defined by Table 4.2, the example axiomatic triples above can be restated as:

$$I(\text{rdf:type}) \in IP ,$$
$$IEXT(I(\text{rdf:type})) \subseteq IR \times IC .$$

Unlike the RDF Semantics, OWL 2 Full does not provide an explicit list of axiomatic triples. It might not be possible to give a definition of OWL 2 Full that captures all intended "axiomatic aspects" of the language in the form of sets of RDF triples, just as it is not possible to define the whole semantics of OWL 2 Full in the form of a set of RDF entailment rules. However, Section 5 contains sets of semantic conditions that are "axiomatic" in the sense described above. Most of these semantic conditions actually have a form similar to those semantic conditions, which resulted from equivalently restating the example axiomatic triples above.

The semantic conditions given in Table 5.2 for "Classes" can be regarded as a set of OWL 2 Full axiomatic triples for classes: For each URI U occurring in the first column of the table, if the second column contains an entry "I(U) ∈ S" for some set S, then this entry corresponds to some RDF triple of the form "U rdf:type C", where C is the URI of some class with ICEXT(I(C)) = S. In this table, S will always be either the set IC of all classes, or some subset of IC. Hence, in a corresponding RDF triple the URI C will typically be one of "rdfs:Class" or "owl:Class" (S=IC in both cases), or "rdfs:Datatype" (S=IDC).

For example, the semantic condition for the URI "owl:FunctionalProperty", given by

I(owl:FunctionalProperty) ∈ IC

has the corresponding RDF triple

owl:FunctionalProperty rdf:type rdfs:Class .

Further, for each URI U in the first column, if the third column contains an entry "ICEXT(I(U)) ⊆ S" (or "ICEXT(I(U)) = S") for some set S, then this entry corresponds to some RDF triple of the form "U rdfs:subClassOf C" (or "U owl:equivalentClass C"), where C is the URI of some class with ICEXT(I(C)) = S.

For example, the semantic condition

ICEXT(I(owl:FunctionalProperty)) ⊆ IP

has the corresponding RDF triple

owl:FunctionalProperty rdfs:subClassOf rdf:Property .

Additionally, the conditions on the sets given in Table 5.1 have to be taken into account. In particular, if an entry of the first column states "$S_1 \subseteq S_2$" for some sets S_1 and S_2, then this corresponds to some RDF triple C_1 owl:subClassOf C_2, where C_1 and C_2 are the URIs of some classes with ICEXT(I(C_1)) = S_1 and ICEXT(I(C_2)) = S_2, respectively, according to Table 5.2.

Note that some of the RDF triples received in this way already follow from the RDFS semantics [*RDF Semantics*].

The semantic conditions given in Table 5.3 for "Properties" can be regarded as a set of OWL 2 Full axiomatic triples for properties: For each URI U occurring in the first column of the table, if the second column contains an entry "I(U) ∈ S" for some set S, then this entry corresponds to some RDF triple of the form "U rdf:type C", where C is the URI of some class with ICEXT(I(C)) = S. In this table, S will always be either the set IP of all properties, or some subset of IP. Hence, in a corresponding RDF triple the URI C will typically be one of "rdf:Property" or "owl:ObjectProperty" (S=IP in both cases), "owl:DatatypeProperty" (S=IODP), "owl:AnnotationProperty" (S=IOAP), or "owl:OntologyProperty" (S=IOXP).

For example, the semantic condition for the URI "owl:disjointWith", given by

I(owl:disjointWith) ∈ IP

has the corresponding RDF triple

owl:disjointWith rdf:type rdf:Property .

Further, for each URI U in the first column, if the third column contains an entry "IEXT(I(U)) ⊆ S_1 × S_2" for some sets S_1 and S_2, then this entry corresponds to some RDF triples of the forms "U rdfs:domain C_1" and "U rdfs:range C_2", where C_1 and C_2 are the URIs of some classes with ICEXT(I(C_1)) = S_1 and ICEXT(I(C_2)) = S_2, respectively.

For example, the semantic condition

IEXT(I(owl:disjointWith)) ⊆ IC × IC

has the corresponding RDF triples

owl:disjointWith rdfs:domain rdfs:Class ,
owl:disjointWith rdfs:range rdfs:Class .

Exceptions are the semantic conditions "IEXT(I(owl:topObjectProperty)) = IR × IR" and "IEXT(I(owl:topDataProperty)) = IR × LV", for which there are no corresponding domain and range triples.

These axiomatic triples are "simple" in the following sense: For every set *S* mentioned in the second and the third column of Table 5.2 for "Classes", there exists a URI *C* of some class in the vocabularies for RDF and RDFS, or those given in Section 2, for which *S = ICEXT(I(C))*. For every set *S* mentioned in the second column of Table 5.3 for "Properties", and as the left or right hand side of a Cartesian product in the third column of the table, there exists a URI *C* of some class in the vocabularies for RDF and RDFS or those given in Section 2, for which *S = ICEXT(I(C))*.

8 Appendix B: Proof of the Correspondence Theorem (Informative)

Editor's Note: TODO: The proof still needs to be constructed.

9 Appendix C: Changes (Informative)

9.1 Changes since First Public Working Draft

This section lists significant changes since the First Public Working Draft.

- Added datatype "owl:rational", marking it "at risk" (WG resolution of Issue 87).
- The RDF syntax of self restrictions has been changed: The class owl:SelfRestriction has been replaced by the property owl:hasSelf (per WG resolution).
- Removed the semantic conditions for axiom annotations (WG resolution of Issue 144).
- Added semantic conditions inferring a union or intersection of datatypes into a datatype (following WG resolution of Issue 147).
- The URIs *owl:TopObjectProperty*, *owl:BottomObjectProperty*, *owl:TopDataProperty* and *owl:BottomDataProperty* have been renamed to their lower-case variants, respectively (per WG decision).
- The datatypes xsd:ID, xsd:IDREF and xsd:ENTITY have been removed (per WG resolution).
- Changed the semantic conditions for the n-ary value restrictions to infer the type of the properties $p_1,...,p_n$ (IODP) and the type of the class c (IDC).
- Corrected definitions of consistency and entailment: The vocabulary V was a global parameter of the definitions. Now the form of the definitions is close to the respective definitions in OWL 1.
- Corrected the semantic condition for sub property chains: missing premise "q in IP" in the second condition.
- Removed redundant statements in the consequent of the semantic conditions for negative property assertions.
- For D-Interpretations, the range of the mapping IL has been changed to IR instead of LV, with a reference to the RDF Semantics. This was a bug, since in both the RDF Semantics and in OWL 1 the range of IL has been IR.
- Split the table on "Parts of the Universe" in the "Semantic Conditions" section into a table defining the parts (now in the "Interpretations" section), and a table that specifies the semantic conditions for those parts.
- The definition of OWL 2 Full datatype maps now include the different facet-related sets that have formerly been part of the "abbreviations" table in the "Semantic Conditions" section.

- The nomenclature for datatype maps has been aligned with the one used in the RDF Semantics. In particular, the concept being called an "interpretation of a literal" is now being called a "datatype value", and the concept being called an "interpretation of a datatype" is now being called a "datatype" or the "value space" of a datatype, depending on whether the datatype itself or its class extension is meant.
- Replaced all applications of the URI-mapping 'IS(.)' by the more general interpretation function 'I(.)'. This usage is now in line with the usage in the RDF Semantics document. Also, there have formerly been applications of IS, where it was not guaranteed that the argument is a URI.
- Marked several sections as "Informative", as requested by a previous review.
- Added to the "Ontologies" section some text about ontology headers and ontology versions, but removed every text referring to the semantic meaning of a OWL 2 Full ontology.
- Moved the "Ontologies" section from Section 5 to Section 3.
- Moved the discussion on axiomatic triples from the section on "Semantic Conditions" to a dedicated appendix.
- The "Introduction" section has been revised.
- The descriptions of the semantic condition tables have been revised.

9.2 Differences to OWL Full

This section lists significant differences between OWL 2 Full and the original version of OWL Full, as defined in Section 5 of [*OWL Semantics and Abstract Syntax*].

Editor's Note: This section needs to be completed before publication as a Last Call working draft. The following items are currently under consideration:

- Role of the Comprehension Principles
- Changes to the correspondence theorem
- Appendix on "Axiomatic triples"
- Imports closure definition
- more specific definition of owl:DataRange
- deprecated URIs (owl:DataRange, …)
- data-version of oneOf semantic condition now requires lists of length ≥ 1
- Bug fixes
  - "sequence-based" constructs
  - missing semantic condition for AllDifferent
- Editorial changes
  - Name and usage of the interpretation function and its components ('IEXT' instead of EXT_I; using always I(.) instead of specific mappings, as in RDF Semantics)
  - Naming convention for "Parts of the Universe" (RDFS takes precedence over OWL 1 Full, but keep OWL 1 Full names in every other case; reduced set of abbreviations)

10 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 Jie Bao (RPI), Peter F. Patel-Schneider (Bell Labs Research, Alcatel-Lucent) and Zhe Wu (Oracle Corporation) 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.

11 References

[OWL 2 Conformance and Test Cases]
OWL 2 Web Ontology Language:Conformance and Test Cases Michael Smith, Ian Horrocks, Markus Krötzsch, eds. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-owl2-test-20081202/. Latest version available at http://www.w3.org/TR/owl2-test/.

[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 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 Structural 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 Semantics and Abstract Syntax]
OWL Web Ontology Language: Semantics and Abstract Syntax. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, eds., W3C Recommendation, 10 February 2004.

[RDF]
Resource Description Framework (RDF): Concepts and Abstract Syntax. Graham Klyne and Jeremy J. Carroll, eds., W3C Recommendation, 10 February 2004.

[RDF Semantics]
RDF Semantics. Patrick Hayes, ed., W3C Recommendation, 10 February 2004.

[RDF:TEXT]
OWL 2 Web Ontology Language:rdf:text: A Datatype for Internationalized Text Jie Bao, Axel Polleres, Boris Motik. W3C Working Draft, 02 December 2008, http://www.w3.org/TR/2008/WD-rdf-text-20081202/. Latest version available at http://www.w3.org/TR/rdf-text/.

[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.