W3C Recommendation

OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition)

W3C Recommendation 11 December 2012

This version:
http://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/
Latest version (series 2):
http://www.w3.org/TR/owl2-rdf-based-semantics/
Latest Recommendation:
http://www.w3.org/TR/owl-rdf-based-semantics
Previous version:
http://www.w3.org/TR/2012/PER-owl2-rdf-based-semantics-20121018/
Editors:
Michael Schneider, FZI Research Center for Information Technology
Contributors: (in alphabetical order)
Jeremy Carroll, HP (now at TopQuadrant)
Ivan Herman, W3C/ERCIM
Peter F. Patel-Schneider, Nuance Communications

Please refer to the **errata** for this document, which may include some normative corrections.

A color-coded version of this document showing changes made since the previous version is also available.

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

See also translations.

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 defines the RDF-compatible model-theoretic semantics of OWL 2.

W3C Recommendation

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May Be Superseded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

There have been no substantive changes since the previous version. For details on the minor changes see the change log and color-coded diff.

Please Send Comments

Please send any comments to public-owl-comments@w3.org (public archive). Although work on this document by the OWL Working Group is complete, comments may be addressed in the errata or in future revisions. Open discussion among developers is welcome at public-owl-dev@w3.org (public archive).

Endorsed By W3C

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

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.

Table of Contents

W3C Recommendation

1 Introduction (Informative)

This document defines the RDF-compatible model-theoretic semantics of OWL 2, referred to as the "*OWL 2 RDF-Based Semantics*". The OWL 2 RDF-Based Semantics gives a formal meaning to every *RDF graph* [*RDF Concepts*] and is fully compatible with the *RDF Semantics specification* [*RDF Semantics*]. The specification provided here is the successor to the original *OWL 1 RDF-Compatible Semantics* specification [*OWL 1 RDF-Compatible Semantics*].

Technically, the OWL 2 RDF-Based Semantics is defined as a *semantic extension* of "D-Entailment" (RDFS with datatype support), as specified in the RDF Semantics [*RDF Semantics*]. In other words, the meaning given to an RDF graph by the OWL 2 RDF-Based Semantics includes the meaning provided by the semantics of RDFS with datatypes, and additional meaning is specified for all the language constructs of OWL 2, such as Boolean connectives, sub property chains and qualified cardinality restrictions (see the OWL 2 Structural Specification [*OWL 2 Specification*] for further information on all the language constructs of OWL 2). The definition of the semantics for the extra constructs follows the design principles as applied to the RDF Semantics.

The content of this document is not meant to be self-contained but builds on top of the RDF Semantics document [*RDF Semantics*] by adding those aspects that are specific to OWL 2. Hence, the complete definition of the OWL 2 RDF-Based Semantics is given by the *combination* of both the RDF Semantics document and the document at hand. In particular, the terminology used in the RDF Semantics is reused here except for cases where a conflict exists with the rest of the OWL 2 specification.

The remainder of this section provides an overview of some of the distinguishing features of the OWL 2 RDF-Based Semantics and outlines the document's structure and content.

In Section 2, the *syntax* over which the OWL 2 RDF-Based Semantics is defined is the set of all *RDF graphs* [*RDF Concepts*]. The OWL 2 RDF-Based Semantics provides a precise formal meaning for every RDF graph. The language that is determined by RDF graphs being interpreted using the OWL 2 RDF-Based Semantics is called "*OWL 2 Full*". In this document, RDF graphs are also called "*OWL 2 Full ontologies*", or simply "*ontologies*", unless there is risk of confusion.

The OWL 2 RDF-Based Semantics interprets the RDF and RDFS *vocabularies* [*RDF Semantics*] and the *OWL 2 RDF-Based vocabulary* together with an extended set of *datatypes* and their constraining *facets* (see Section 3).

OWL 2 RDF-Based interpretations (Section 4) are defined on a *universe* (see Section 1.3 of the RDF Semantics specification [*RDF Semantics*] for an overview of the basic intuition of model-theoretic semantics). The universe is divided into *parts*, namely *individuals*, *classes*, and *properties*, which are identified with their RDF counterparts (see Figure 1). The part of individuals equals the whole universe. This means that all classes and properties are also individuals in their own right. Further, every name interpreted by an OWL 2 RDF-Based interpretation denotes an individual.

The three basic parts are divided into further parts as follows. The part of individuals subsumes the part of *data values*, which comprises the denotations of all literals. Also subsumed by the individuals is the part of *ontologies*. The part of classes subsumes the part of *datatypes*, which are classes consisting entirely of data values. Finally, the part of properties subsumes the parts of *object properties*, *data properties*, *ontology properties* and *annotation properties*. The part of object properties equals the whole part of properties, and therefore all other kinds of properties are also object properties.

For *annotations properties* note that annotations are not "semantic-free" under the OWL 2 RDF-Based Semantics. Just like every other triple or set of triples occurring in an RDF graph, an annotation is assigned a truth value by any given OWL 2 RDF-Based interpretation. Hence, although annotations are meant to be "semantically weak", i.e., their formal meaning does not significantly exceed that originating from the RDF Semantics specification, adding an annotation may still change the meaning of an ontology. A similar discussion holds for statements that are built from *ontology properties*, such as `owl:imports`, which are used to define relationships between two ontologies.

Every class represents a specific set of individuals, called the *class extension* of the class: an individual a is an instance of a class C, if a is a member of the class extension ICEXT(C). Since a class is itself an individual under the OWL 2 RDF-Based Semantics, classes are distinguished from their respective class extensions. This distinction allows, for example, that a class may be an instance of itself by being a member of its own class extension. Also, two classes may be equivalent by sharing the same class extension, although being different individuals, e.g., they do not need to share the same properties. Similarly, every property has an associated *property extension* that consists of pairs of individuals: an individual a_1 has a relationship to an individual a_2 with respect to a property p if the pair (a_1 , a_2) is a member of the property extension IEXT(p). Again, properties are distinguished from their property extensions. In general, if there are no further constraints, an arbitrary extension may be associated with a given class or property, and two interpretations may associate distinct extensions with the same class or property.

Individuals may *play different* "*roles*". For example, an individual can be both a data property and an annotation property, since the different parts of the universe of an OWL 2 RDF-Based interpretation are not required to be mutually disjoint, or an individual can be both a class and a property by associating both a class extension and a property extension with it. In the latter case there will be no specific relationship between the class extension and the property extension of such an individual without further constraints. For example, the same individual can have an empty class extension while having a nonempty property extension.

The main part of the OWL 2 RDF-Based Semantics is Section 5, which specifies a formal meaning for all the OWL 2 language constructs by means of the *OWL 2 RDF-Based semantic conditions*. These semantic conditions extend all the semantic conditions given in the RDF Semantics [*RDF Semantics*]. The OWL 2 RDF-Based semantic conditions effectively determine which sets of RDF triples are assigned a specific meaning and what this meaning is. For example, semantic conditions exist that allow one to interpret the triple "*C* `owl:disjointWith` *D*" to mean that the denotations of the IRIs *C* and *D* have disjoint class extensions.

There is usually no need to provide *localizing information* (e.g., by means of "typing triples") for the IRIs occurring in an ontology. As for the RDF Semantics, the OWL 2 RDF-Based semantic conditions have been designed to ensure that the denotation of any IRI will be in the appropriate part of the universe. For example, the RDF triple "*C* `owl:disjointWith` *D*" is sufficient to deduce that the denotations of the IRIs *C* and *D* are actually *classes*. It is not necessary to explicitly add additional typing triples "*C* `rdf:type rdfs:Class`" and "*D* `rdf:type rdfs:Class`" to the ontology.

In the RDF Semantics, this kind of "automatic localization" was to some extent achieved by so called *"axiomatic triples"* [*RDF Semantics*], such as "`rdf:type rdf:type rdf:Property`" or "`rdf:type rdfs:domain rdfs:Resource`". However, there is no explicit normative collection of additional axiomatic triples for the OWL 2 RDF-Based Semantics;

instead, the specific axiomatic aspects of the OWL 2 RDF-Based Semantics are determined by a subset of the OWL 2 RDF-Based semantic conditions. Section 6 discusses axiomatic triples in general and provides an example set of axiomatic triples that is compatible with the OWL 2 RDF-Based Semantics.

Section 7 compares the OWL 2 RDF-Based Semantics with the *OWL 2 Direct Semantics* [*OWL 2 Direct Semantics*]. While the OWL 2 RDF-Based Semantics is based on the RDF Semantics specification [*RDF Semantics*], the OWL 2 Direct Semantics is a *description logic* style semantics. Several fundamental differences exist between the two semantics, but there is also a strong relationship basically stating that the OWL 2 RDF-Based Semantics is able to reflect all logical conclusions of the OWL 2 Direct Semantics. This means that the OWL 2 Direct Semantics can in a sense be regarded as a semantics subset of the OWL 2 RDF-Based Semantics. The precise relationship is given by the *OWL 2 correspondence theorem*.

Significant effort has been spent in keeping the design of the OWL 2 RDF-Based Semantics as close as possible to that of the original specification of the *OWL 1 RDF-Compatible Semantics* [*OWL 1 RDF-Compatible Semantics*]. While this aim was achieved to a large degree, the OWL 2 RDF-Based Semantics actually deviates from its predecessor in several aspects. In most cases, this is because of serious technical problems that would have arisen from a conservative semantic extension. One important change is that while so called "*comprehension conditions*" for the OWL 2 RDF-Based Semantics (see Section 8) still exist, these are *not* part of the normative set of semantic conditions anymore. The OWL 2 RDF-Based Semantics also corrects several errors of OWL 1. A list of differences between the two languages is given in Section 9.

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*].

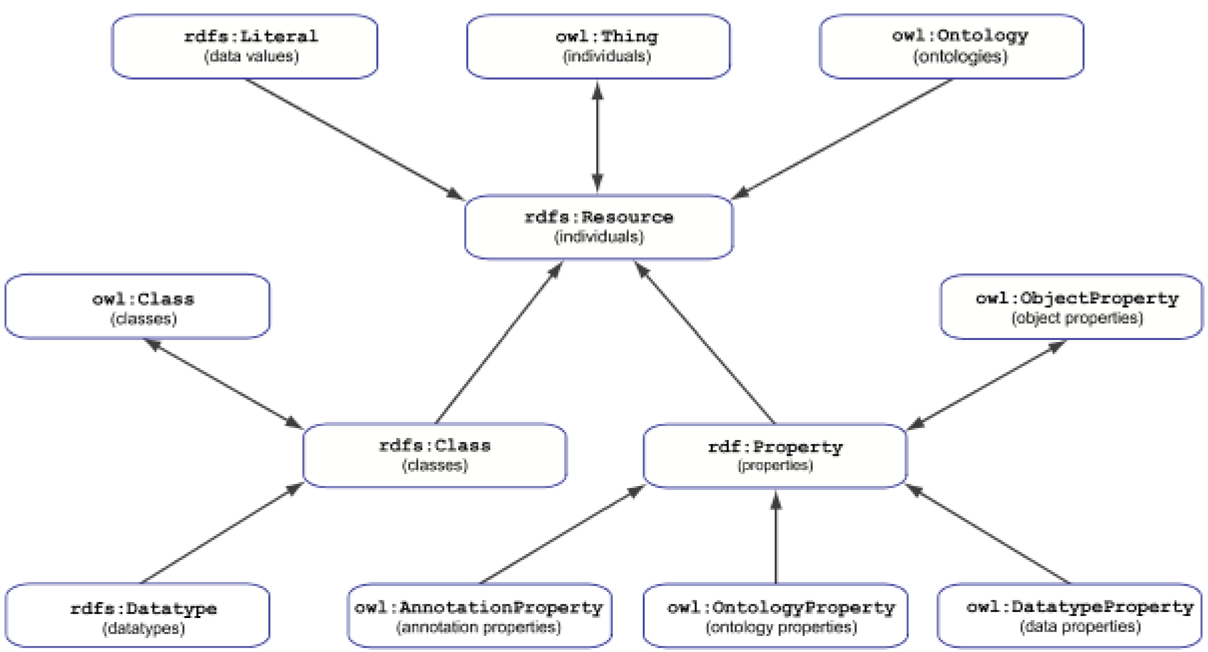

Figure 1: Parts Hierarchy of the OWL 2 RDF-Based Semantics
Each *node* is labeled with a class IRI that represents a part of the universe of an OWL 2 RDF-based interpretation. *Arrows* point from parts to their super parts.

W3C Recommendation

2 Ontologies

This section determines the *syntax* for the OWL 2 RDF-Based Semantics, and gives an overview on typical *content of ontologies* for ontology management tasks.

2.1 Syntax

Following Sections 0.2 and 0.3 of the RDF Semantics specification [*RDF Semantics*], the OWL 2 RDF-Based Semantics is defined on every ***RDF graph*** (Section 6.2 of RDF Concepts [*RDF Concepts*]), i.e. on every set of ***RDF triples*** (Section 6.1 of RDF Concepts [*RDF Concepts*]).

In accordance with the rest of the OWL 2 specification (see Section 2.4 of the OWL 2 Structural Specification [*OWL 2 Specification*]), this document uses an extended notion of an RDF graph by allowing the RDF triples in an RDF graph to contain arbitrary ***IRIs*** ("Internationalized Resource Identifiers") according to RFC 3987 [*RFC 3987*]. In contrast, the RDF Semantics specification [*RDF Semantics*] is defined on RDF graphs containing *URIs* [*RFC 2396*]. This change is backward compatible with the RDF specification, since URIs are also IRIs.

Terminological note: The document at hand uses the term "IRI" in accordance with the rest of the OWL 2 specification (see Section 2.4 of the OWL 2 Structural Specification [*OWL 2 Specification*]), whereas the RDF Semantics specification [*RDF Semantics*] uses the term "URI reference". According to RFC 3987 [*RFC 3987*], the term "IRI" stands for an absolute resource identifier with optional fragment, which is what is being used throughout this document. In contrast, the term "IRI reference" additionally covers *relative* references, which are never used in this document.

Convention: In this document, IRIs are abbreviated in the way defined by Section 2.4 of the OWL 2 Structural Specification [*OWL 2 Specification*], i.e., the abbreviations consist of a *prefix name* and a *local part*, such as "`prefix:localpart`".

The definition of an RDF triple according to Section 6.1 of RDF Concepts [*RDF Concepts*] is restricted to cases where the *subject* of an RDF triple is an IRI or a *blank node* (Section 6.6 of RDF Concepts [*RDF Concepts*]), and where the *predicate* of an RDF triple is an IRI. As a consequence, the definition does not treat cases, where, for example, the subject of a triple is a *literal* (Section 6.5 of RDF Concepts [*RDF Concepts*]), as in `"s" ex:p ex:o`, or where the predicate of a triple is a blank node, as in `ex:s _:p ex:o`. In order to allow for interoperability with other existing and future technologies and tools, the document at hand does not explicitly forbid the use of ***generalized RDF graphs*** consisting of ***generalized RDF triples***, which are defined to allow for IRIs, literals and blank nodes to occur in the subject, predicate and object position. Thus, an RDF graph *may* contain generalized RDF triples, but an implementation is not required to support generalized RDF graphs. Note that every RDF graph consisting entirely of RDF triples according to Section 6.1 of RDF Concepts [*RDF Concepts*] is also a generalized RDF graph.

Terminological notes: The term **"*OWL 2 Full*"** refers to the language that is determined by the set of all RDF graphs being interpreted using the OWL 2 RDF-Based Semantics. Further, in this document the term **"*OWL 2 Full ontology*"** (or simply **"*ontology*"**, unless there is any risk of confusion) will be used interchangeably with the term "RDF graph".

W3C Recommendation

2.2 Content of Ontologies (Informative)

While there do not exist any syntactic restrictions on the set of RDF graphs that can be interpreted by the OWL 2 RDF-Based Semantics, in practice an ontology will often contain certain kinds of constructs that are aimed to support ontology management tasks. Examples are ***ontology headers*** and ***ontology IRIs***, as well as constructs that are about ***versioning***, ***importing*** and ***annotating*** of ontologies, including the concept of ***incompatibility*** between ontologies.

These topics are outside the scope of this semantics specification. Section 3 of the OWL 2 Structural Specification [*OWL 2 Specification*] deals with these topics in detail, and can therefore be used as a guide on how to apply these constructs in OWL 2 Full ontologies accordingly. The mappings of all these constructs to their respective RDF encoding are defined in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*].

3 Vocabulary

This section specifies the *OWL 2 RDF-Based vocabulary*, and lists the names of the *datatypes* and *facets* used under the OWL 2 RDF-Based Semantics.

3.1 Standard Prefixes

Table 3.1 lists the standard prefix names and their prefix IRIs used in this document.

Table 3.1: Standard Prefixes

	Prefix Name	**Prefix IRI**
OWL	`owl`	`http://www.w3.org/2002/07/owl#`
RDF	`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
RDFS	`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`
XML Schema	`xsd`	`http://www.w3.org/2001/XMLSchema#`

3.2 Vocabulary Terms

Table 3.2 lists the IRIs of the *OWL 2 RDF-Based vocabulary*, which is the set of vocabulary terms that are specific for the OWL 2 RDF-Based Semantics. This vocabulary extends the RDF and RDFS vocabularies as specified in Sections 3.1 and 4.1 of the RDF Semantics [*RDF Semantics*], respectively. Table 3.2 does not mention those IRIs that will be listed in Section 3.3 on datatype names or Section 3.4 on facet names.

Implementations are *not* required to support the IRI `owl:onProperties`, but *may* support it in order to realize *n-ary dataranges* with arity ≥ 2 (see Sections 7 and 8.4 of the OWL 2 Structural Specification [*OWL 2 Specification*] for further information).

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

Table 3.2: OWL 2 RDF-Based Vocabulary

```
owl:AllDifferent owl:AllDisjointClasses owl:AllDisjointProperties
owl:allValuesFrom owl:annotatedProperty owl:annotatedSource
owl:annotatedTarget 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:ObjectProperty owl:onClass owl:onDataRange owl:onDatatype owl:oneOf
owl:onProperty owl:onProperties owl:Ontology owl:OntologyProperty
owl:priorVersion owl:propertyChainAxiom owl:propertyDisjointWith
owl:qualifiedCardinality owl:ReflexiveProperty owl:Restriction owl:sameAs
owl:someValuesFrom owl:sourceIndividual owl:SymmetricProperty
owl:targetIndividual owl:targetValue owl:Thing owl:topDataProperty
owl:topObjectProperty owl:TransitiveProperty owl:unionOf owl:versionInfo
owl:versionIRI owl:withRestrictions
```

3.3 Datatype Names

Table 3.3 lists the IRIs of the *datatypes* used in the OWL 2 RDF-Based Semantics. The datatype `rdf:XMLLiteral` is described in Section 3.1 of the RDF Semantics [*RDF Semantics*]. All other datatypes are described in Section 4 of the OWL 2 Structural Specification [*OWL 2 Specification*]. The normative set of datatypes of the OWL 2 RDF-Based Semantics equals the set of datatypes described in Section 4 of the OWL 2 Structural Specification [*OWL 2 Specification*].

Table 3.3: Datatypes of the OWL 2 RDF-Based Semantics

```
xsd:anyURI xsd:base64Binary xsd:boolean xsd:byte xsd:dateTime
xsd:dateTimeStamp 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 rdf:PlainLiteral xsd:positiveInteger owl:rational
owl:real xsd:short xsd:string xsd:token xsd:unsignedByte xsd:unsignedInt
xsd:unsignedLong xsd:unsignedShort rdf:XMLLiteral
```

3.4 Facet Names

Table 3.4 lists the IRIs of the *facets* used in the OWL 2 RDF-Based Semantics. Each datatype listed in Section 3.3 has a (possibly empty) set of constraining facets. All facets are described in Section 4 of the OWL 2 Structural Specification [*OWL 2 Specification*] in the context of their respective datatypes. The normative set of facets of the OWL 2 RDF-Based Semantics equals the set of facets described in Section 4 of the OWL 2 Structural Specification [*OWL 2 Specification*].

In this specification, facets are used for defining *datatype restrictions* (see Section 5.7). For example, to refer to the set of all strings of length 5 one can restrict the datatype `xsd:string` (Section 3.3) by the facet `xsd:length` and the value 5.

Table 3.4: Facets of the OWL 2 RDF-Based Semantics

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

4 Interpretations

The OWL 2 RDF-Based Semantics provides *vocabulary interpretations* and *vocabulary entailment* (see Section 2.1 of the RDF Semantics [*RDF Semantics*]) for the RDF and RDFS vocabularies and for the OWL 2 RDF-Based vocabulary. This section defines *OWL 2 RDF-Based datatype maps* and *OWL 2 RDF-Based interpretations*, and specifies what *satisfaction* of ontologies, *consistency* and *entailment* means under the OWL 2 RDF-Based Semantics. In addition, the so called "*parts*" *of the universe* of an OWL 2 RDF-Based interpretation are defined.

4.1 Datatype Maps

According to Section 5.1 of the RDF Semantics specification [*RDF Semantics*], a ***datatype*** *d* has the following components:

- LS(*d*), the *lexical space* of *d*, which is a set of *lexical forms*;
- VS(*d*), the *value space* of *d*, which is a set of *data values*;
- L2V(*d*), the *lexical-to-value mapping* of *d*, which maps lexical forms in LS(*d*) to data values in VS(*d*).

Terminological notes: The document at hand uses the term "*data value*" in accordance with the rest of the OWL 2 specification (see Section 4 of the OWL 2 Structural Specification [*OWL 2 Specification*]), whereas the RDF Semantics specification [*RDF Semantics*] uses the term "*datatype value*" instead. Further, the names "LS" and "VS", which stand for the lexical space and the value space of a datatype, respectively, are *not* used in the RDF Semantics specification, but have been introduced here for easier reference.

In this document, the basic definition of a datatype is extended to take *facets* into account. See Section 3.4 for information and an example on facets. Note that Section 5.1 of the RDF Semantics specification [*RDF Semantics*] explicitly permits that semantic extensions may impose more elaborate datatyping conditions than those listed above.

A ***datatype with facets*** *d* is a datatype that has the following additional components:

- FS(*d*), the *facet space* of *d*, which is a set of pairs of the form (*F* , *v*), where *F* is an IRI called the *constraining facet* and *v* is an arbitrary data value called the *constraining value*;
- F2V(*d*), the *facet-to-value mapping* of *d*, which maps each facet-value pair (*F* , *v*) in FS(*d*) to a subset of VS(*d*).

Note that it is not further specified what the nature of the denotation of a facet IRI is, i.e. it is only known that a facet IRI denotes some individual. Semantic extensions *may* impose

W3C Recommendation

further restrictions on the denotations of facets. In fact, Section 5.3 will define additional restrictions on facets.

Also note that for a datatype *d* and a facet-value pair (*F* , *v*) in FS(*d*) the value *v* is not required to be included in the value space VS(*d*) of *d* itself. For example, the datatype `xsd:string` (Section 3.3) has the facet `xsd:length` (Section 3.4), which takes nonnegative integers as its constraining values rather than strings.

In this document, it will always be assumed from now on that any datatype *d* is a datatype with facets. If the facet space FS(*d*) of a datatype *d* has not been explicitly defined, or if it is not derived from another datatype's facet space according to some well defined condition, then FS(*d*) is the empty set. Unless there is any risk of confusion, the term "*datatype*" will always refer to a datatype with facets.

Section 5.1 of the RDF Semantics specification [*RDF Semantics*] further defines a ***datatype map*** *D* to be a set of name-datatype pairs (*u* , *d*) consisting of an IRI *u* and a datatype *d*, such that no IRI appears twice in the set. As a consequence of what has been said before, in this document every datatype map *D* will entirely consist of datatypes with facets.

The following definition specifies what an *OWL 2 RDF-Based datatype map* is.

Definition 4.1 (OWL 2 RDF-Based Datatype Map): A datatype map *D* is an *OWL 2 RDF-Based datatype map*, if and only if for every datatype name *u* listed in Section 3.3 and its respective set of constraining facets (Section 3.4) there is a name-datatype pair (*u*, *d*) in *D* with the specified lexical space LS(*d*), value space VS(*d*), lexical-to-value mapping L2V(*d*), facet space FS(*d*) and facet-to-value mapping F2V(*d*).

Note that Definition 4.1 does not prevent *additional* datatypes to be in an OWL 2 RDF-Based datatype map. For the special case of an OWL 2 RDF-Based datatype map *D* that exclusively contains the datatypes listed in Section 3.3, it is ensured that there are datatypes available for all the facet values, i.e., for every name-datatype pair (*u* , *d*) in *D* and for every facet-value pair (*F* , *v*) in FS(*d*) there exists a name-datatype pair (u^* , d^*) in *D* such that *v* is in VS(d^*).

4.2 Vocabulary Interpretations

From the RDF Semantics specification [*RDF Semantics*], let *V* be a set of literals and IRIs containing the RDF and RDFS vocabularies, and let *D* be a datatype map according to Section 5.1 of the RDF Semantics [*RDF Semantics*] (and accordingly Section 4.1). A ***D-interpretation*** *I* of *V* with respect to *D* is a tuple

I = (IR , IP , IEXT , IS , IL , LV) .

IR is the *universe* of *I*, i.e., a nonempty set that contains at least the denotations of literals and IRIs in *V*. IP is a subset of IR, the *properties* of *I*. LV, the *data values* of *I*, is a subset of IR that contains at least the set of plain literals (see Section 6.5 of RDF Concepts [*RDF Concepts*]) in *V*, and the value spaces of each datatype of *D*. IEXT is used to associate properties with their *property extension*, and is a mapping from IP to the powerset of IR × IR. IS is a mapping from *IRIs* in *V* to their denotations in IR. In particular, IS(*u*) = *d* for any name-datatype pair (*u* , *d*) in *D*. IL is a mapping from *typed literals* "*s*"^^*u* in *V* to their denotations in IR, where IL("*s*"^^*u*) = L2V(*d*)(*s*), provided that *d* is a datatype of *D*, IS(*u*) = *d*, and *s* is in the lexical space LS(*d*); otherwise IL("*s*"^^*u*) is not in LV.

Convention: Following the practice introduced in Section 1.4 of the RDF Semantics [*RDF Semantics*], for a given interpretation *I* of a vocabulary *V* the notation "*I*(*x*)" will be used instead of "IL(*x*)" and "IS(*x*)" for the typed literals and IRIs *x* in *V*, respectively.

As detailed in the RDF Semantics [*RDF Semantics*], a D-interpretation has to meet all the semantic conditions for ground graphs and blank nodes, those for RDF interpretations and RDFS interpretations, and the "general semantic conditions for datatypes".

In this document, the basic definition of a D-interpretation is extended to take *facets* into account.

A ***D-interpretation with facets*** *I* is a D-interpretation for a datatype map *D* consisting entirely of datatypes with facets (Section 4.1), where *I* meets the following additional semantic conditions: for each name-datatype pair (*u* , *d*) in *D* and each facet-value pair (*F* , *v*) in the facet space FS(*d*)

- *F* is in the vocabulary *V* of *I*;
- a name-datatype pair (u^* , d^*) exists in *D*, such that *v* is in the value space VS(d^*).

In this document, it will always be assumed from now on that any D-interpretation *I* is a D-interpretation with facets. Unless there is any risk of confusion, the term "*D-interpretation*" will always refer to a D-interpretation with facets.

The following definition specifies what an *OWL 2 RDF-Based interpretation* is.

Definition 4.2 (OWL 2 RDF-Based Interpretation): Let *D* be an OWL 2 RDF-Based datatype map, and let *V* be a vocabulary that includes the RDF and RDFS vocabularies and the OWL 2 RDF-Based vocabulary together with all the datatype and facet names listed in Section 3. An *OWL 2 RDF-Based interpretation*, *I* = (IR , IP , IEXT , IS , IL , LV), of *V* with respect to *D* is a D-interpretation of *V* with respect to *D* that meets all the extra semantic conditions given in Section 5.

4.3 Satisfaction, Consistency and Entailment

The following definitions specify what it means for an RDF graph to be *satisfied* by a given OWL 2 RDF-Based interpretation, to be *consistent* under the OWL 2 RDF-Based Semantics, and to *entail* another RDF graph.

The notion of *satisfaction* under the OWL 2 RDF-Based Semantics is based on the notion of satisfaction for D-interpretations and Simple interpretations, as defined in the RDF Semantics [*RDF Semantics*]. In essence, in order to satisfy an RDF graph, an interpretation *I* has to satisfy all the triples in the graph, i.e., for a triple "*s p o*" it is necessary that the relationship (*I*(*s*) , *I*(*o*)) ∈ IEXT(*I*(*p*)) holds (special treatment exists for blank nodes, as detailed in Section 1.5 of the RDF Semantics [*RDF Semantics*]). In other words, the given graph has to be compatible with the specific form of the IEXT mapping of *I*. The distinguishing aspect of *OWL 2 RDF-Based satisfaction* is that an interpretation *I* needs to meet all the OWL 2 RDF-Based semantic conditions (see Section 5), which have a constraining effect on the possible forms an IEXT mapping can have.

Definition 4.3 (OWL 2 RDF-Based Satisfaction): Let *G* be an RDF graph, let *D* be an OWL 2 RDF-Based datatype map, let *V* be a vocabulary that includes the RDF and RDFS vocabularies and the OWL 2 RDF-Based vocabulary together with all the datatype and facet names listed in Section 3, and let *I* be a D-interpretation of *V* with respect to *D*. *I OWL 2*

RDF-Based satisfies G with respect to *V* and *D* if and only if *I* is an OWL 2 RDF-Based interpretation of *V* with respect to *D* that satisfies *G* as a D-interpretation of *V* with respect to *D* according to the RDF Semantics [*RDF Semantics*].

Definition 4.4 (OWL 2 RDF-Based Consistency): Let *S* be a collection of RDF graphs, and let *D* be an OWL 2 RDF-Based datatype map. *S* is *OWL 2 RDF-Based consistent* with respect to *D* if and only if there is some OWL 2 RDF-Based interpretation *I* with respect to *D* of some vocabulary *V* that includes the RDF and RDFS vocabularies and the OWL 2 RDF-Based vocabulary together with all the datatype and facet names listed in Section 3, such that *I* OWL 2 RDF-Based satisfies all the RDF graphs in *S* with respect to *V* and *D*.

Definition 4.5 (OWL 2 RDF-Based Entailment): Let S_1 and S_2 be collections of RDF graphs, and let *D* be an OWL 2 RDF-Based datatype map. S_1 *OWL 2 RDF-Based entails* S_2 with respect to *D* if and only if for every OWL 2 RDF-Based interpretation *I* with respect to *D* of any vocabulary *V* that includes the RDF and RDFS vocabularies and the OWL 2 RDF-Based vocabulary together with all the datatype and facet names listed in Section 3 the following holds: If *I* OWL 2 RDF-Based satisfies all the RDF graphs in S_1 with respect to *V* and *D*, then *I* OWL 2 RDF-Based satisfies all the RDF graphs in S_2 with respect to *V* and *D*.

4.4 Parts of the Universe

Table 4.1 defines the "*parts*" of the universe of a given OWL 2 RDF-Based interpretation *I*.

The second column tells the *name* of the part. The third column gives a *definition* of the part in terms of the mapping IEXT of *I*, and by referring to a particular term of the RDF, RDFS or OWL 2 RDF-Based vocabulary.

As an example, the part of all datatypes is named "IDC", and it is defined as the set of all individuals *x* for which the relationship "(*x* , *I*(`rdfs:Datatype`)) ∈ IEXT(*I*(`rdf:type`))" holds. According to the semantics of `rdf:type`, as defined in Section 4.1 of the RDF Semantics [*RDF Semantics*], this means that the name "IDC" denotes the class extension (see Section 4.5) of *I*(`rdfs:Datatype`).

Table 4.1: Parts of the Universe

	Name of Part *S*	**Definition of *S* as { *x* ∈ IR | (*x* , *I*(*E*)) ∈ IEXT(*I*(`rdf:type`)) } where IRI *E* is**
individuals	IR	`rdfs:Resource`
data values	LV	`rdfs:Literal`
ontologies	IX	`owl:Ontology`
classes	IC	`rdfs:Class`
datatypes	IDC	`rdfs:Datatype`
properties	IP	`rdf:Property`
data properties	IODP	`owl:DatatypeProperty`
ontology properties	IOXP	`owl:OntologyProperty`

W3C Recommendation

annotation properties	IOAP	`owl:AnnotationProperty`

4.5 Class Extensions

The mapping ICEXT from IC to the powerset of IR, which associates classes with their *class extension*, is defined for every $c \in$ IC as

ICEXT(c) = { $x \in$ IR | (x , c) $\in$ IEXT(I(`rdf:type`)) } .

5 Semantic Conditions

This section defines the semantic conditions of the OWL 2 RDF-Based Semantics. The semantic conditions presented here are basically only those for the specific constructs of OWL 2. The complete set of semantic conditions for the OWL 2 RDF-Based Semantics is the combination of the semantic conditions presented here and the semantic conditions for Simple Entailment, RDF, RDFS and D-Entailment, as specified in the RDF Semantics specification [*RDF Semantics*].

All semantic conditions in this section are defined with respect to an interpretation I. Section 5.1 specifies semantic conditions for the different parts of the universe of the interpretation being considered (compare Section 4.4). Section 5.2 and Section 5.3 list semantic conditions for the classes and the properties of the OWL 2 RDF-Based vocabulary. In the rest of this section, the OWL 2 RDF-Based semantic conditions for the different language constructs of OWL 2 are specified.

Conventions used in this Section

iff: Throughout this section the term "iff" is used as a shortform for "if and only if".

Conjunctive commas: A comma ("`,`") separating two assertions in a semantic condition, as in "$c \in$ IC , $p \in$ IP", is read as a logical "*and*". Further, a comma separating two variables, as in "c, $d \in$ IC", is used for abbreviating two comma separated assertions, "$c \in$ IC , $d \in$ IC" in this example.

Unscoped variables: If no explicit scope is given for a variable "x", as in "∀ x : …" or "{ x | … }", then "x" is unconstrained, which means $x \in$ IR, i.e. "x" denotes an arbitrary individual in the universe.

Set cardinality: For a set S, an expression of the form "#S" means the number of elements in S.

Sequence expressions: An expression of the form "s sequence of a_1 , … , $a_n \in S$" means that "s" represents an RDF list of $n \geq 0$ individuals a_1 , … , a_n, all of them being members of the set S. Precisely, $s = I$(`rdf:nil`) for $n = 0$; and for $n > 0$ there exist $z_1 \in$ IR , … , $z_n \in$ IR, such that

$s = z_1$,
$a_1 \in S$, (z_1 , a_1) $\in$ IEXT(I(`rdf:first`)) , (z_1 , z_2) $\in$ IEXT(I(`rdf:rest`)) ,
… ,
$a_n \in S$, (z_n , a_n) $\in$ IEXT(I(`rdf:first`)) , (z_n , I(`rdf:nil`)) $\in$ IEXT(I(`rdf:rest`)) .

Note, as mentioned in Section 3.3.3 of the RDF Semantics [*RDF Semantics*], there are no semantic constraints that enforce "well-formed" sequence structures. So, for example, it is possible for a sequence head *s* to refer to more than one sequence.

Set names: The following names are used as convenient abbreviations for certain sets:

- ISEQ: The set of all sequences. This set equals the class extension of `rdf:List`, i.e., ISEQ := ICEXT(*I*(`rdf:List`)).
- INNI: The set of all nonnegative integers. This set equals the value space of the datatype `xsd:nonNegativeInteger`, i.e., INNI := ICEXT(*I*(`xsd:nonNegativeInteger`)), but is also subsumed by the value spaces of other numerical datatypes, such as `xsd:integer`.

Notes on the Form of Semantic Conditions (Informative)

One design goal of OWL 2 was to ensure an appropriate degree of alignment between the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] under the different constraints the two semantics have to meet. The way this semantic alignment is described is via the *OWL 2 correspondence theorem* in Section 7.2. For this theorem to hold, the semantic conditions that treat the RDF encoding of OWL 2 *axioms* (compare Section 3.2.5 of the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*] and Section 9 of the OWL 2 Structural Specification [*OWL 2 Specification*]), such as inverse property axioms, must have the form of "iff" ("if-and-only-if") conditions. This means that these semantic conditions completely determine the semantics of the encoding of these constructs. On the other hand, the RDF encoding of OWL 2 *expressions* (compare Section 3.2.4 of the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*] and Sections 6 – 8 of the OWL 2 Structural Specification [*OWL 2 Specification*]), such as property restrictions, are treated by "if-then" conditions. These weaker semantic conditions for expressions are sufficient for the correspondence theorem to hold, so there is no necessity to define stronger "iff" conditions under the OWL 2 RDF-Based Semantics for these language constructs.

Special cases are the semantic conditions for Boolean connectives of classes and for enumerations. These language constructs build OWL 2 expressions. But for backward compatibility reasons there is also RDF encoding of *axioms* based on the vocabulary for these language constructs (see Table 18 in Section 3.2.5 of the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]). For example, an RDF expression of the form

```
ex:c1 owl:unionOf ( ex:c2 ex:c3 ) .
```

is mapped by the reverse RDF mapping to an OWL 2 axiom that states the equivalence of the class denoted by `ex:c1` with the union of the classes denoted by `ex:c2` and `ex:c3`. In order to ensure that the correspondence theorem holds, and in accordance with the original OWL 1 RDF-Compatible Semantics specification [*OWL 1 RDF-Compatible Semantics*], the semantic conditions for the mentioned language constructs are therefore "iff" conditions.

Further, special treatment exists for OWL 2 axioms that have a *multi-triple representation* in RDF, where the different triples share a common "*root node*", such as the blank node "`_:x`" in the following example:

```
_:x rdf:type owl:AllDisjointClasses .
_:x owl:members ( ex:c1 ex:c2 ) .
```

In essence, the semantic conditions for the encoding of these language constructs are "iff" conditions, as usual for axioms. However, in order to cope with the specific syntactic aspect

W3C Recommendation

of a "root node", the "iff" conditions of these language constructs have been split into two "if-then" conditions, where the "if-then" condition representing the right-to-left direction contains an additional premise having the form "∃ *z* ∈ IR". The purpose of this premise is to ensure the existence of an individual that is needed to satisfy the root node under the OWL 2 RDF-Based semantics. The language constructs in question are *n-ary disjointness axioms* in Section 5.10, and *negative property assertions* in Section 5.15.

The "if-then" semantic conditions in this section sometimes do not explicitly list all typing statements in their consequent that one might expect. For example, the semantic condition for `owl:someValuesFrom` restrictions in Section 5.6 does not list the statement "*x* ∈ ICEXT(*I*(`owl:Restriction`))" on its right hand side. Consequences are generally not mentioned, if they can already be deduced by other means. Often, these redundant consequences follow from the semantic conditions for *vocabulary classes* and *vocabulary properties* in Section 5.2 and Section 5.3, respectively, occasionally in connection with the semantic conditions for the *parts of the universe* in Section 5.1. In the example above, the omitted consequence can be obtained from the third column of the entry for `owl:someValuesFrom` in the table in Section 5.3, which determines that IEXT(*I*(`owl:someValuesFrom`)) ⊆ ICEXT(*I*(`owl:Restriction`)) × IC.

5.1 Semantic Conditions for the Parts of the Universe

Table 5.1 lists the semantic conditions for the parts of the universe of the OWL 2 RDF-Based interpretation being considered. Additional semantic conditions affecting these parts are given in Section 5.2.

The first column tells the *name* of the part, as defined in Section 4.4. The second column defines certain *conditions* on the part. In most cases, the column specifies for the part by which other part it is subsumed, and thus the position of the part in the "parts hierarchy" of the universe is narrowed down. The third column provides further *information about the instances* of those parts that consist of classes or properties. In general, if the part consists of classes, then for the class extensions of the member classes is specified by which part of the universe they are subsumed. If the part consists of properties, then the domains and ranges of the member properties are determined.

Table 5.1: Semantic Conditions for the Parts of the Universe

Name of Part *S*	Conditions on *S*	Conditions on Instances *x* of *S*
IR	*S* ≠ ∅	
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

W3C Recommendation

IOXP	$S \subseteq$ IP	IEXT(x) $\subseteq$ IX × IX
IOAP	$S \subseteq$ IP	IEXT(x) $\subseteq$ IR × IR

5.2 Semantic Conditions for the Vocabulary Classes

Table 5.2 lists the semantic conditions for the classes that have IRIs in the OWL 2 RDF-Based vocabulary. In addition, the table contains all those classes with IRIs in the RDF and RDFS vocabularies that represent parts of the universe of the OWL 2 RDF-Based interpretation being considered (Section 4.4). The semantic conditions for the remaining classes with names in the RDF and RDFS vocabularies can be found in the RDF Semantics specification [*RDF Semantics*].

The first column tells the *IRI* of the class. The second column defines of what particular *kind* a class is, i.e. whether it is a general class (a member of the part IC) or a datatype (a member of IDC). The third column specifies for the class extension of the class by which part of the universe (Section 4.4) it is *subsumed*: from an entry of the form "ICEXT(I(C)) $\subseteq$ S", for a class IRI C and a set S, and given an RDF triple of the form "*u* `rdf:type` *C*", one can deduce that the relationship "I(u) $\in$ S" holds. Note that some entries are of the form "ICEXT(I(C)) = S", which means that the class extension is exactly specified to be that set. See Section 5.1 for further semantic conditions on those classes that represent *parts*.

Not included in this table are the *datatypes* of the OWL 2 RDF-Based Semantics with IRIs listed in Section 3.3. For each such datatype IRI E, the following semantic conditions hold (as a consequence of the fact that E is a member of the datatype map of every OWL 2 RDF-Based interpretation according to Definition 4.2, and by the "general semantic conditions for datatypes" listed in Section 5.1 of the RDF Semantics [*RDF Semantics*]):

- I(E) $\in$ IDC
- ICEXT(I(E)) $\subseteq$ LV

Table 5.2: Semantic Conditions for the Vocabulary Classes

IRI *E*	*I*(*E*)	ICEXT(*I*(*E*))
`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

W3C Recommendation

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

5.3 Semantic Conditions for the Vocabulary Properties

Table 5.3 lists the semantic conditions for the properties that have IRIs in the OWL 2 RDF-Based vocabulary. In addition, the table contains all those properties with IRIs in the RDFS vocabulary that are specified to be annotation properties under the OWL 2 RDF-Based Semantics. The semantic conditions for the remaining properties with names in the RDF and RDFS vocabularies can be found in the RDF Semantics specification [*RDF Semantics*].

The first column tells the *IRI* of the property. The second column defines of what particular *kind* a property is, i.e. whether it is a general property (a member of the part IP), a datatype property (a member of IODP), an ontology property (a member of IOXP) or an annotation property (a member of IOAP). The third column specifies the *domain and range* of the property: from an entry of the form "IEXT($I(p)$) ⊆ $S_1 \times S_2$", for a property IRI p and sets S_1

W3C Recommendation

and S_2, and given an RDF triple "*s p o*", one can deduce the relationships "$I(s) \in S_1$" and "$I(o) \in S_2$". Note that some entries are of the form "$\text{IEXT}(I(p)) = S_1 \times S_2$", which means that the property extension is exactly specified to be the Cartesian product of the two sets.

Not included in this table are the *facets* of the OWL 2 RDF-Based Semantics with IRIs listed in Section 3.4, which are used to specify datatype restrictions (see Section 5.7). For each such facet IRI *E*, the following semantic conditions *extend* the basic semantics specification that has been given for *datatypes with facets* in Section 4.1:

- $I(E) \in$ IODP
- $\text{IEXT}(I(E)) \subseteq \text{IR} \times \text{LV}$

Implementations are *not* required to support the semantic condition for `owl:onProperties`, but *may* support it in order to realize *n-ary dataranges* with arity ≥ 2 (see Sections 7 and 8.4 of the OWL 2 Structural Specification [*OWL 2 Specification*] for further information).

Informative notes:

`owl:topObjectProperty` relates every two individuals in the universe with each other. Likewise, `owl:topDataProperty` relates every individual with every data value. Further, `owl:bottomObjectProperty` and `owl:bottomDataProperty` stand both for the *empty* relationship.

The ranges of the properties `owl:deprecated` and `owl:hasSelf` are not restricted in any form, and, in particular, they are not restricted to Boolean values. The actual object values of these properties do not have any intended meaning, but could as well have been defined to be of any other value. Therefore, the semantics given here are of a form that the values can be arbitrarily chosen without leading to any nontrivial semantic conclusions. It is, however, recommended to still use an object literal of the form `"true"^^xsd:boolean` in ontologies, in order to not get in conflict with the required usage of these properties in scenarios that ask for applying the reverse RDF mapping (compare Table 13 in Section 3.2.4 of the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*] for `owl:hasSelf`, and Section 5.5 of the OWL 2 Structural Specification [*OWL 2 Specification*] for `owl:deprecated`).

The range of the property `owl:annotatedProperty` is unrestricted, i.e. it is *not* specified as the set of properties. Annotations are meant to be "semantically weak", i.e. their formal meaning should not significantly exceed that originating from the RDF Semantics specification.

Several properties, such as `owl:priorVersion`, have been specified as both ontology properties and annotation properties, in order to be in line with both the original OWL 1 RDF-Compatible Semantics specification [*OWL 1 RDF-Compatible Semantics*] and the rest of the OWL 2 specification (see Section 5.5 of the OWL 2 Structural Specification [*OWL 2 Specification*]).

Table 5.3: Semantic Conditions for the Vocabulary Properties

IRI *E*	*I*(*E*)	IEXT(*I*(*E*))
`owl:allValuesFrom`	∈ IP	⊆ ICEXT(*I*(`owl:Restriction`)) × IC
`owl:annotatedProperty`	∈ IP	⊆ IR × IR
`owl:annotatedSource`	∈ IP	⊆ IR × IR

W3C Recommendation

owl:annotatedTarget	∈ IP	⊆ IR × IR
owl:assertionProperty	∈ IP	⊆ ICEXT(*I*(owl:NegativePropertyAssertion)) × IP
owl:backwardCompatibleWith	∈ IOXP , ∈ IOAP	⊆ IX × IX
owl:bottomDataProperty	∈ IODP	= ∅
owl:bottomObjectProperty	∈ IP	= ∅
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 , ∈ IOAP	⊆ IX × IX
owl:intersectionOf	∈ IP	⊆ IC × ISEQ
owl:inverseOf	∈ IP	⊆ IP × IP
rdfs:isDefinedBy	∈ IOAP	⊆ IR × IR
rdfs:label	∈ IOAP	⊆ IR × LV

W3C Recommendation

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: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:priorVersion	∈ IOXP , ∈ IOAP	⊆ IX × IX
owl:propertyChainAxiom	∈ 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: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

W3C Recommendation

owl:versionInfo	∈ IOAP	⊆ IR × IR
owl:versionIRI	∈ IOXP	⊆ IX × IX
owl:withRestrictions	∈ IP	⊆ IDC × ISEQ

5.4 Semantic Conditions for Boolean Connectives

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

Informative notes: Of the three pairs of semantic conditions in the table every first is an "iff" condition, since the corresponding OWL 2 language constructs are both class expressions and axioms. In contrast, the semantic condition on datatype complements is an "if-then" condition, since it only corresponds to a datarange expression. See the *notes on the form of semantic conditions* for further information. For the remaining semantic conditions that treat the cases of intersections and unions of datatypes it is sufficient to have "if-then" conditions, since stronger "iff" conditions would be redundant due to the more general "iff" conditions that already exist for classes. Note that the datatype related semantic conditions do not apply to empty sets, but one can still receive a datatype from an empty set by explicitly asserting the resulting class to be an instance of class `rdfs:Datatype`.

Table 5.4: Semantic Conditions for Boolean Connectives

if s sequence of c_1 , … , c_n ∈ IR then		
(z , s) ∈ IEXT(I(owl:intersectionOf))	**iff**	z , c_1 , … , c_n ∈ IC , ICEXT(z) = ICEXT(c_1) ∩ … ∩ ICEXT(c_n)
if	**then**	
s sequence of d_1 , … , d_n ∈ IDC , n ≥ 1 , (z , s) ∈ IEXT(I(owl:intersectionOf))	z ∈ IDC	
if s sequence of c_1 , … , c_n ∈ IR then		
(z , s) ∈ IEXT(I(owl:unionOf))	**iff**	z , c_1 , … , c_n ∈ IC , ICEXT(z) = ICEXT(c_1) ∪ … ∪ ICEXT(c_n)
if	**then**	
s sequence of d_1 , … , d_n ∈ IDC , n ≥ 1 , (z , s) ∈ IEXT(I(owl:unionOf))	z ∈ IDC	
(z , c) ∈ IEXT(I(owl:complementOf))	**iff**	z , c ∈ IC , ICEXT(z) = IR \ ICEXT(c)

if	then
(z , d) ∈ IEXT(I(owl:datatypeComplementOf))	ICEXT(z) = LV \ ICEXT(d)

5.5 Semantic Conditions for Enumerations

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 data values is a datatype.

Informative notes: The first semantic condition is an "iff" condition, since the corresponding OWL 2 language construct is both a class expression and an axiom. See the *notes on the form of semantic conditions* for further information. For the remaining semantic condition that treats the case of enumerations of data values it is sufficient to have an "if-then" condition, since a stronger "iff" condition would be redundant due to the more general "iff" condition that already exists for individuals. Note that the data value related semantic condition does not apply to empty sets, but one can still receive a datatype from an empty set by explicitly asserting the resulting class to be an instance of class rdfs:Datatype.

Table 5.5: Semantic Conditions for Enumerations

if s sequence of a_1 , … , a_n ∈ IR then		
(z , s) ∈ IEXT(I(owl:oneOf))	**iff**	z ∈ IC , ICEXT(z) = { a_1 , … , a_n }

if	then
s sequence of v_1 , … , v_n ∈ LV , $n \geq 1$, (z , s) ∈ IEXT(I(owl:oneOf))	z ∈ IDC

5.6 Semantic Conditions for Property Restrictions

Table 5.6 lists the semantic conditions for property restrictions.

Value restrictions require that some or all of the values of a certain property must be instances of a given class or data range, or that the property has a specifically defined value. By placing a *self restriction* on some given property one only considers those individuals that are reflexively related to themselves via this property. *Cardinality restrictions* determine how often a certain property is allowed to be applied to a given individual. *Qualified cardinality restrictions* are more specific than cardinality restrictions in that they determine the quantity of a property application with respect to a particular class or data range from which the property values are taken.

Implementations are *not* required to support the semantic conditions for owl:onProperties, but *may* support them in order to realize *n-ary dataranges* with arity ≥ 2 (see Sections 7 and 8.4 of the OWL 2 Structural Specification [*OWL 2 Specification*] for further information).

W3C Recommendation

Informative notes: All the semantic conditions are "if-then" conditions, since the corresponding OWL 2 language constructs are class expressions. The "if-then" conditions generally only list those consequences on their right hand side that are specific for the respective condition, i.e. consequences that do not already follow by other means. See the *notes on the form of semantic conditions* for further information. Note that the semantic condition for *self restrictions* does not constrain the right hand side of a `owl:hasSelf` assertion to be the Boolean value `"true"^^xsd:boolean`. See Section 5.3 for an explanation.

Table 5.6: Semantic Conditions for Property Restrictions

if	**then**
$(z , c) \in$ IEXT(I(`owl:someValuesFrom`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid \exists\, y : (x , y) \in$ IEXT(p) and $y \in$ ICEXT(c) $\}$
s sequence of $p_1 , \ldots , p_n \in$ IR , $n \geq 1$, $(z , c) \in$ IEXT(I(`owl:someValuesFrom`)) , $(z , s) \in$ IEXT(I(`owl:onProperties`))	$p_1 , \ldots , p_n \in$ IP , ICEXT(z) = $\{ x \mid \exists\, y_1 , \ldots , y_n : (x , y_k) \in$ IEXT(p_k) for each $1 \leq k \leq n$ and $(y_1 , \ldots , y_n) \in$ ICEXT(c) $\}$
$(z , c) \in$ IEXT(I(`owl:allValuesFrom`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid \forall\, y : (x , y) \in$ IEXT(p) implies $y \in$ ICEXT(c) $\}$
s sequence of $p_1 , \ldots , p_n \in$ IR , $n \geq 1$, $(z , c) \in$ IEXT(I(`owl:allValuesFrom`)) , $(z , s) \in$ IEXT(I(`owl:onProperties`))	$p_1 , \ldots , p_n \in$ IP , ICEXT(z) = $\{ x \mid \forall\, y_1 , \ldots , y_n : (x , y_k) \in$ IEXT(p_k) for each $1 \leq k \leq n$ implies $(y_1 , \ldots , y_n) \in$ ICEXT(c) $\}$
$(z , a) \in$ IEXT(I(`owl:hasValue`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid (x , a) \in$ IEXT(p) $\}$
$(z , v) \in$ IEXT(I(`owl:hasSelf`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid (x , x) \in$ IEXT(p) $\}$
$(z , n) \in$ IEXT(I(`owl:minCardinality`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid \#\{ y \mid (x , y) \in$ IEXT(p) $\} \geq n \}$
$(z , n) \in$ IEXT(I(`owl:maxCardinality`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid \#\{ y \mid (x , y) \in$ IEXT(p) $\} \leq n \}$
$(z , n) \in$ IEXT(I(`owl:cardinality`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`))	ICEXT(z) = $\{ x \mid \#\{ y \mid (x , y) \in$ IEXT(p) $\} = n \}$
$(z , n) \in$ IEXT(I(`owl:minQualifiedCardinality`)) , $(z , p) \in$ IEXT(I(`owl:onProperty`)) , $(z , c) \in$ IEXT(I(`owl:onClass`))	ICEXT(z) = $\{ x \mid \#\{ y \mid (x , y) \in$ IEXT(p) and $y \in$ ICEXT(c) $\} \geq n \}$

W3C Recommendation

(*z* , *n*) ∈ IEXT(*I*(`owl:minQualifiedCardinality`)) , (*z* , *p*) ∈ IEXT(*I*(`owl:onProperty`)) , (*z* , *d*) ∈ IEXT(*I*(`owl:onDataRange`))	*p* ∈ IODP , ICEXT(*z*) = { *x* | #{ *y* ∈ LV | (*x* , *y*) ∈ IEXT(*p*) and *y* ∈ ICEXT(*d*) } ≥ *n* }
(*z* , *n*) ∈ IEXT(*I*(`owl:maxQualifiedCardinality`)) , (*z* , *p*) ∈ IEXT(*I*(`owl:onProperty`)) , (*z* , *c*) ∈ IEXT(*I*(`owl:onClass`))	ICEXT(*z*) = { *x* | #{ *y* | (*x* , *y*) ∈ IEXT(*p*) and *y* ∈ ICEXT(*c*) } ≤ *n* }
(*z* , *n*) ∈ IEXT(*I*(`owl:maxQualifiedCardinality`)) , (*z* , *p*) ∈ IEXT(*I*(`owl:onProperty`)) , (*z* , *d*) ∈ IEXT(*I*(`owl:onDataRange`))	*p* ∈ IODP , ICEXT(*z*) = { *x* | #{ *y* ∈ LV | (*x* , *y*) ∈ IEXT(*p*) and *y* ∈ ICEXT(*d*) } ≤ *n* }
(*z* , *n*) ∈ IEXT(*I*(`owl:qualifiedCardinality`)) , (*z* , *p*) ∈ IEXT(*I*(`owl:onProperty`)) , (*z* , *c*) ∈ IEXT(*I*(`owl:onClass`))	ICEXT(*z*) = { *x* | #{ *y* | (*x* , *y*) ∈ IEXT(*p*) and *y* ∈ ICEXT(*c*) } = *n* }
(*z* , *n*) ∈ IEXT(*I*(`owl:qualifiedCardinality`)) , (*z* , *p*) ∈ IEXT(*I*(`owl:onProperty`)) , (*z* , *d*) ∈ IEXT(*I*(`owl:onDataRange`))	*p* ∈ IODP , ICEXT(*z*) = { *x* | #{ *y* ∈ LV | (*x* , *y*) ∈ IEXT(*p*) and *y* ∈ ICEXT(*d*) } = *n* }

5.7 Semantic Conditions for Datatype Restrictions

Table 5.7 lists the semantic conditions for datatype restrictions, which are used to define sub datatypes of existing datatypes by restricting the original datatype by means of a set of facet-value pairs. See Section 3.4 for information and an example on constraining facets.

Certain special cases exist: If no facet-value pair is applied to a given datatype, then the resulting datatype will be equivalent to the original datatype. Further, if a facet-value pair is applied to a datatype without being a member of the datatype's facet space, then the ontology cannot be satisfied and will therefore be inconsistent. In particular, a datatype restriction with one or more specified facet-value pairs will result in an inconsistent ontology, if applied to a datatype with an empty facet space.

The set **IFS** is defined by IFS(*d*) := { (*I*(*F*) , *v*) | (*F* , *v*) ∈ FS(*d*) } , where *d* is a datatype, *F* is the IRI of a constraining facet, and *v* is a constraining value of the facet. This set corresponds to the facet space FS(*d*), as defined in Section 4.1, but rather consists of pairs of the *denotation* of a facet and a value.

The mapping **IF2V** is defined by IF2V(*d*)((*I*(*F*) , *v*)) := F2V(*d*)((*F* , *v*)) , where *d* is a datatype, *F* is the IRI of a constraining facet, and *v* is a constraining value of the facet. This mapping corresponds to the facet-to-value mapping F2V(*d*), as defined in Section 4.1, resulting in the same subsets of the value space VS(*d*), but rather applies to pairs of the *denotation* of a facet and a value.

W3C Recommendation

Informative notes: The semantic condition is an "if-then" condition, since the corresponding OWL 2 language construct is a datarange expression. The "if-then" condition only lists those consequences on its right hand side that are specific for the condition, i.e. consequences that do not already follow by other means. See the *notes on the form of semantic conditions* for further information.

Table 5.7: Semantic Conditions for Datatype Restrictions

if	then
s sequence of z_1 , … , z_n ∈ IR , f_1 , … , f_n ∈ IP , (z , d) ∈ IEXT(I(`owl:onDatatype`)) , (z , s) ∈ IEXT(I(`owl:withRestrictions`)) , (z_1 , v_1) ∈ IEXT(f_1) , … , (z_n , v_n) ∈ IEXT(f_n)	f_1 , … , f_n ∈ IODP , v_1 , … , v_n ∈ LV , (f_1 , v_1) , … , (f_n , v_n) ∈ IFS(d) , ICEXT(z) = ICEXT(d) ∩ IF2V(d)((f_1 , v_1)) ∩ … ∩ IF2V(d)((f_n , v_n))

5.8 Semantic Conditions for the RDFS Vocabulary

Table 5.8 *extends* the RDFS semantic conditions for subclass axioms, subproperty axioms, domain axioms and range axioms. The semantic conditions provided here are "iff" conditions, while the original semantic conditions, as specified in Section 4.1 of the RDF Semantics [*RDF Semantics*], are weaker "if-then" conditions. Only the additional semantic conditions are given here and the other conditions of RDF and RDFS are retained.

Informative notes: All the semantic conditions are "iff" conditions, since the corresponding OWL 2 language constructs are axioms. See the *notes on the form of semantic conditions* for further information.

Table 5.8: Semantic Conditions for the RDFS Vocabulary

(c_1 , c_2) ∈ IEXT(I(`rdfs:subClassOf`))	**iff**	c_1 , c_2 ∈ IC , ICEXT(c_1) ⊆ ICEXT(c_2)
(p_1 , p_2) ∈ IEXT(I(`rdfs:subPropertyOf`))		p_1 , p_2 ∈ IP , IEXT(p_1) ⊆ IEXT(p_2)
(p , c) ∈ IEXT(I(`rdfs:domain`))		p ∈ IP , c ∈ IC , ∀ x , y : (x , y) ∈ IEXT(p) implies x ∈ ICEXT(c)
(p , c) ∈ IEXT(I(`rdfs:range`))		p ∈ IP , c ∈ IC , ∀ x , y : (x , y) ∈ IEXT(p) implies y ∈ ICEXT(c)

5.9 Semantic Conditions for Equivalence and Disjointness

Table 5.9 lists the semantic conditions for specifying that two individuals are equal or different from each other, and that either two classes or two properties are equivalent or disjoint with each other, respectively. The property `owl:equivalentClass` is also used to

W3C Recommendation

formulate *datatype definitions* (see Section 9.4 of the OWL 2 Structural Specification [*OWL 2 Specification*] for information about datatype definitions). In addition, the table treats disjoint union axioms.

Informative notes: All the semantic conditions are "iff" conditions, since the corresponding OWL 2 language constructs are axioms. See the *notes on the form of semantic conditions* for further information.

Table 5.9: Semantic Conditions for Equivalence and Disjointness

$(a_1 , a_2) \in$ IEXT(I(`owl:sameAs`))	iff	$a_1 = a_2$
$(a_1 , a_2) \in$ IEXT(I(`owl:differentFrom`))		$a_1 \neq a_2$
$(c_1 , c_2) \in$ IEXT(I(`owl:equivalentClass`))		$c_1 , c_2 \in$ IC , ICEXT(c_1) = ICEXT(c_2)
$(c_1 , c_2) \in$ IEXT(I(`owl:disjointWith`))		$c_1 , c_2 \in$ IC , ICEXT(c_1) $\cap$ ICEXT(c_2) = $\emptyset$
$(p_1 , p_2) \in$ IEXT(I(`owl:equivalentProperty`))		$p_1 , p_2 \in$ IP , IEXT(p_1) = IEXT(p_2)
$(p_1 , p_2) \in$ IEXT(I(`owl:propertyDisjointWith`))		$p_1 , p_2 \in$ IP , IEXT(p_1) $\cap$ IEXT(p_2) = $\emptyset$
if s sequence of $c_1 , \ldots , c_n \in$ IR then		
$(c , s) \in$ IEXT(I(`owl:disjointUnionOf`))	**iff**	$c , c_1 , \ldots , c_n \in$ IC , ICEXT(c) = ICEXT(c_1) $\cup \ldots \cup$ ICEXT(c_n) , ICEXT(c_j) $\cap$ ICEXT(c_k) = $\emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$

5.10 Semantic Conditions for N-ary Disjointness

Table 5.10 lists the semantic conditions for specifying n-ary diversity and disjointness axioms, i.e. that several given individuals are mutually different from each other, and that several given classes or properties are mutually disjoint with each other, respectively.

Note that there are two alternative ways to specify `owl:AllDifferent` axioms, by using either the property `owl:members` that is used for all other constructs, too, or by applying the legacy property `owl:distinctMembers`. Both variants have an equivalent formal meaning.

Informative notes: The semantic conditions essentially represent "iff" conditions, since the corresponding OWL 2 language constructs are axioms. However, there are actually *two* semantic conditions for each language construct due to the multi-triple RDF encoding of these language constructs. The "if-then" conditions only list those consequences on their right hand side that are specific for the respective condition, i.e. consequences that do not already follow by other means. See the *notes on the form of semantic conditions* for further information.

W3C Recommendation

Table 5.10: Semantic Conditions for N-ary Disjointness

if	then
s sequence of $a_1 , \ldots , a_n \in$ IR , $z \in$ ICEXT(I(`owl:AllDifferent`)) , (z , s) $\in$ IEXT(I(`owl:members`))	$a_j \neq a_k$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
if	**then exists $z \in$ IR**
s sequence of $a_1 , \ldots , a_n \in$ IR , $a_j \neq a_k$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$	$z \in$ ICEXT(I(`owl:AllDifferent`)) , (z , s) $\in$ IEXT(I(`owl:members`))
if	**then**
s sequence of $a_1 , \ldots , a_n \in$ IR , $z \in$ ICEXT(I(`owl:AllDifferent`)) , (z , s) $\in$ IEXT(I(`owl:distinctMembers`))	$a_j \neq a_k$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
if	**then exists $z \in$ IR**
s sequence of $a_1 , \ldots , a_n \in$ IR , $a_j \neq a_k$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$	$z \in$ ICEXT(I(`owl:AllDifferent`)) , (z , s) $\in$ IEXT(I(`owl:distinctMembers`))
if	**then**
s sequence of $c_1 , \ldots , c_n \in$ IR , $z \in$ ICEXT(I(`owl:AllDisjointClasses`)) , (z , s) $\in$ IEXT(I(`owl:members`))	$c_1 , \ldots , c_n \in$ IC , ICEXT(c_j) $\cap$ ICEXT(c_k) $= \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
if	**then exists $z \in$ IR**
s sequence of $c_1 , \ldots , c_n \in$ IC , ICEXT(c_j) $\cap$ ICEXT(c_k) $= \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$	$z \in$ ICEXT(I(`owl:AllDisjointClasses`)) , (z , s) $\in$ IEXT(I(`owl:members`))
if	**then**
s sequence of $p_1 , \ldots , p_n \in$ IR , $z \in$ ICEXT(I(`owl:AllDisjointProperties`)) , (z , s) $\in$ IEXT(I(`owl:members`))	$p_1 , \ldots , p_n \in$ IP , IEXT(p_j) $\cap$ IEXT(p_k) $= \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
if	**then exists $z \in$ IR**
s sequence of $p_1 , \ldots , p_n \in$ IP , IEXT(p_j) $\cap$ IEXT(p_k) $= \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$	$z \in$ ICEXT(I(`owl:AllDisjointProperties`)) , (z , s) $\in$ IEXT(I(`owl:members`))

W3C Recommendation

5.11 Semantic Conditions for Sub Property Chains

Table 5.11 lists the semantic conditions for sub property chains, which allow for specifying complex property subsumption axioms.

As an example, one can define a sub property chain axiom that specifies the chain consisting of the property extensions of properties `ex:hasFather` and `ex:hasBrother` to be a sub relation of the extension of the property `ex:hasUncle`.

Informative notes: The semantic condition is an "iff" condition, since the corresponding OWL 2 language construct is an axiom. See the *notes on the form of semantic conditions* for further information. The semantics has been specified in a way such that a sub property chain axiom can be satisfied without requiring the existence of a property that has the property chain as its property extension.

Table 5.11: Semantic Conditions for Sub Property Chains

if s sequence of p_1 , … , p_n ∈ IR then		
(p , s) ∈ IEXT(I(`owl:propertyChainAxiom`))	**iff**	p ∈ IP , p_1 , … , p_n ∈ IP , ∀ y_0 , … , y_n : (y_0 , y_1) ∈ IEXT(p_1) and … and (y_{n-1} , y_n) ∈ IEXT(p_n) implies (y_0 , y_n) ∈ IEXT(p)

5.12 Semantic Conditions for Inverse Properties

Table 5.12 lists the semantic conditions for inverse property axioms. The inverse of a given property is the corresponding property with subject and object swapped for each property assertion built from it.

Informative notes: The semantic condition is an "iff" condition, since the corresponding OWL 2 language construct is an axiom. See the *notes on the form of semantic conditions* for further information.

Table 5.12: Semantic Conditions for Inverse Properties

(p_1 , p_2) ∈ IEXT(I(`owl:inverseOf`))	**iff**	p_1 , p_2 ∈ IP , IEXT(p_1) = { (x , y) | (y , x) ∈ IEXT(p_2) }

5.13 Semantic Conditions for Property Characteristics

Table 5.13 lists the semantic conditions for property characteristics.

If a property is *functional*, then at most one distinct value can be assigned to any given individual via this property. An *inverse functional* property can be regarded as a "key" property, i.e. no two different individuals can be assigned the same value via this property. A *reflexive* property relates every individual in the universe to itself, whereas an *irreflexive* property does not relate any individual with itself. If two individuals are related by a *symmetric* property, then this property also relates them reversely, while this is never the case for an *asymmetric* property. A *transitive* property that relates an individual *a* with an individual *b*, and the latter with an individual *c*, also relates *a* with *c*.

Informative notes: All the semantic conditions are "iff" conditions, since the corresponding OWL 2 language constructs are axioms. See the *notes on the form of semantic conditions* for further information.

Table 5.13: Semantic Conditions for Property Characteristics

$p \in$ ICEXT(I(`owl:FunctionalProperty`))	iff	$p \in$ IP , $\forall\, x , y_1 , y_2 : (x , y_1) \in$ IEXT(p) and $(x , y_2) \in$ IEXT(p) implies $y_1 = y_2$
$p \in$ ICEXT(I(`owl:InverseFunctionalProperty`))		$p \in$ IP , $\forall\, x_1 , x_2 , y : (x_1 , y) \in$ IEXT(p) and $(x_2 , y) \in$ IEXT(p) implies $x_1 = x_2$
$p \in$ ICEXT(I(`owl:ReflexiveProperty`))		$p \in$ IP , $\forall\, x : (x , x) \in$ IEXT(p)
$p \in$ ICEXT(I(`owl:IrreflexiveProperty`))		$p \in$ IP , $\forall\, x : (x , x) \notin$ IEXT(p)
$p \in$ ICEXT(I(`owl:SymmetricProperty`))		$p \in$ IP , $\forall\, x , y : (x , y) \in$ IEXT(p) implies $(y , x) \in$ IEXT(p)
$p \in$ ICEXT(I(`owl:AsymmetricProperty`))		$p \in$ IP , $\forall\, x , y : (x , y) \in$ IEXT(p) implies $(y , x) \notin$ IEXT(p)
$p \in$ ICEXT(I(`owl:TransitiveProperty`))		$p \in$ IP , $\forall\, x , y , z : (x , y) \in$ IEXT(p) and $(y , z) \in$ IEXT(p) implies $(x , z) \in$ IEXT(p)

5.14 Semantic Conditions for Keys

Table 5.14 lists the semantic conditions for Keys.

Keys provide an alternative to inverse functional properties (see Section 5.13). They allow for defining a property as a key local to a given class: the specified property will have the features of a key only for individuals being instances of the class, and no assumption is made about individuals for which membership of the class cannot be entailed. Further, it is possible to define "compound keys", i.e. several properties can be combined into a single key applicable to composite values. Note that keys are not functional by default under the OWL 2 RDF-Based Semantics.

Informative notes: The semantic condition is an "iff" condition, since the corresponding OWL 2 language construct is an axiom. See the *notes on the form of semantic conditions* for further information.

Table 5.14: Semantic Conditions for Keys

if s sequence of $p_1 , \ldots , p_n \in$ IR then

(c , s) ∈ IEXT(I(owl:hasKey))	**iff**	c ∈ IC , p_1 , … , p_n ∈ IP , ∀ x , y , z_1 , … , z_n : if x ∈ ICEXT(c) and y ∈ ICEXT(c) and (x , z_k) ∈ IEXT(p_k) and (y , z_k) ∈ IEXT(p_k) for each $1 \le k \le n$ then $x = y$

5.15 Semantic Conditions for Negative Property Assertions

Table 5.15 lists the semantic conditions for negative property assertions. They allow to state that two given individuals are *not* related by a given property.

The second form based on `owl:targetValue` is more specific than the first form based on `owl:targetIndividual` in that the second form is restricted to the case of negative *data* property assertions. Note that the second form will coerce the target value of a negative property assertion into a data value, due to the range defined for the property `owl:targetValue` in Section 5.3.

Informative notes: The semantic conditions essentially represent "iff" conditions, since the corresponding OWL 2 language constructs are axioms. However, there are actually *two* semantic conditions for each language construct, due to the multi-triple RDF encoding of these language constructs. The "if-then" conditions only list those consequences on their right hand side that are specific for the respective condition, i.e. consequences that do not already follow by other means. See the *notes on the form of semantic conditions* for further information.

Table 5.15: Semantic Conditions for Negative Property Assertions

if	then
(z , a_1) ∈ IEXT(I(owl:sourceIndividual)) , (z , p) ∈ IEXT(I(owl:assertionProperty)) , (z , a_2) ∈ IEXT(I(owl:targetIndividual))	(a_1 , a_2) ∉ IEXT(p)
if	**then exists z ∈ IR**
a_1 ∈ IR , p ∈ IP , a_2 ∈ IR , (a_1 , a_2) ∉ IEXT(p)	(z , a_1) ∈ IEXT(I(owl:sourceIndividual)) , (z , p) ∈ IEXT(I(owl:assertionProperty)) , (z , a_2) ∈ IEXT(I(owl:targetIndividual))
if	**then**
(z , a) ∈ IEXT(I(owl:sourceIndividual)) , (z , p) ∈ IEXT(I(owl:assertionProperty)) , (z , v) ∈ IEXT(I(owl:targetValue))	p ∈ IODP , (a , v) ∉ IEXT(p)

if	then exists $z \in$ IR
$a \in$ IR , $p \in$ IODP , $v \in$ LV , (a , v) $\notin$ IEXT(p)	(z , a) $\in$ IEXT(I(owl:sourceIndividual)) , (z , p) $\in$ IEXT(I(owl:assertionProperty)) , (z , v) $\in$ IEXT(I(owl:targetValue))

6 Appendix: Axiomatic Triples (Informative)

The RDF Semantics specification [*RDF Semantics*] defines so called "*axiomatic triples*" as part of the semantics of RDF and RDFS. Unlike the RDF Semantics, the OWL 2 RDF-Based Semantics does not normatively specify any axiomatic triples, since one cannot expect to find a set of RDF triples that fully captures all "axiomatic aspects" of the OWL 2 RDF-Based Semantics. Furthermore, axiomatic triples for the OWL 2 RDF-Based Semantics could, in principle, contain arbitrarily complex class expressions, e.g. the union of several classes, and by this it becomes nonobvious which of several possible nonequivalent sets of axiomatic triples should be selected. However, the OWL 2 RDF-Based Semantics includes many semantic conditions that can in a sense be regarded as being "axiomatic", and thus can be considered a replacement for the missing axiomatic triples. After an overview on axiomatic triples for RDF and RDFS in Section 6.1, Sections 6.2 and 6.3 will discuss how the "axiomatic" semantic conditions of the OWL 2 RDF-Based Semantics relate to axiomatic triples. Based on this discussion, an explicit example set of axiomatic triples that is compatible with the OWL 2 RDF-Based Semantics will be provided in Section 6.4.

6.1 Axiomatic Triples in RDF

In RDF and RDFS [*RDF Semantics*], axiomatic triples are used to provide basic meaning for all the vocabulary terms of the two languages. This formal meaning is independent of any given RDF graph, and it even holds for vocabulary terms, which do not occur in a graph that is interpreted by an RDF or RDFS interpretation. As a consequence, all the axiomatic triples of RDF and RDFS are entailed by the *empty* graph, when being interpreted under the semantics of RDF or RDFS, respectively.

Examples of RDF and RDFS axiomatic triples are:

(1) `rdf:type rdf:type rdf:Property .`
(2) `rdf:type rdfs:domain rdfs:Resource .`
(3) `rdf:type rdfs:range rdfs:Class .`
(4) `rdfs:Datatype rdfs:subClassOf rdfs:Class .`
(5) `rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .`

As shown by these examples, axiomatic triples are typically used by the RDF Semantics specification to determine the part of the universe to which the denotation of a vocabulary term belongs (1). In the case of a property, the domain (2) and range (3) is specified as well. Also, in some cases, hierarchical relationships between classes (4) or properties (5) of the vocabulary are determined.

Under the OWL 2 RDF-Based Semantics, all the axiomatic triples of RDF and RDFS could, in principle, be replaced by "axiomatic" semantic conditions that have neither premises nor bound variables. By applying the *RDFS semantic conditions* given in Section 5.8, the example axiomatic triples (1) – (5) can be equivalently restated as:

W3C Recommendation

I(`rdf:type`) ∈ ICEXT(*I*(`rdf:Property`)) ,
IEXT(*I*(`rdf:type`)) ⊆ ICEXT(*I*(`rdfs:Resource`)) × ICEXT(*I*(`rdfs:Class`)) ,
ICEXT(*I*(`rdfs:Datatype`)) ⊆ ICEXT(*I*(`rdfs:Class`)) ,
IEXT(*I*(`rdfs:isDefinedBy`)) ⊆ IEXT(*I*(`rdfs:seeAlso`)) .

All the axiomatic triples of RDF and RDFS can be considered "*simple*" in the sense that they have in their object position only single terms from the RDF and RDFS vocabularies, and no complex class or property expressions appear there.

6.2 Axiomatic Triples for the Vocabulary Classes

The semantic conditions for *vocabulary classes* in Section 5.2 can be considered as corresponding to a set of axiomatic triples for the classes in the vocabulary of the OWL 2 RDF-Based Semantics.

First, for each IRI *E* occurring in the first column of Table 5.2, if the *second* column contains an entry of the form "*I*(*E*) ∈ *S*" for some set *S*, then this entry corresponds to an RDF triple of the form "*E* `rdf:type` *C*", where *C* is the IRI of a vocabulary class with ICEXT(*I*(*C*)) = *S*. In the table, *S* will always be either the part IC of all classes, or some sub part of IC. Hence, in a corresponding RDF triple the IRI *C* will be one of "`rdfs:Class`", "`owl:Class`" (*S*=IC in both cases) or "`rdfs:Datatype`" (*S*=IDC).

For example, for the IRI "`owl:FunctionalProperty`", the semantic condition

I(`owl:FunctionalProperty`) ∈ IC

has the corresponding axiomatic triple

```
owl:FunctionalProperty rdf:type rdfs:Class .
```

Further, for each IRI *E* in the first column of the table, if the *third* column contains an entry of the form "ICEXT(*I*(*E*)) ⊆ *S*" (or "ICEXT(*I*(*E*)) = *S*") for some set *S*, then this entry corresponds to an RDF triple of the form "*E* `rdfs:subClassOf` *C*" (or additionally "*C* `rdfs:subClassOf` *E*"), where *C* is the IRI of a vocabulary class with ICEXT(*I*(*C*)) = *S*. In each case, *S* will be one of the parts of the universe of *I*.

For example, the semantic condition

ICEXT(*I*(`owl:FunctionalProperty`)) ⊆ IP

has the corresponding axiomatic triple

```
owl:FunctionalProperty rdfs:subClassOf rdf:Property .
```

In addition, the semantic conditions for the *parts of the universe* in Table 5.1 of Section 5.1 have to be taken into account. In particular, if an entry in the *second* column of Table 5.1 is of the form "$S_1 \subseteq S_2$" for some sets S_1 and S_2, then this corresponds to an RDF triple of the form "C_1 `owl:subClassOf` C_2", where C_1 and C_2 are the IRIs of vocabulary classes with ICEXT(*I*(C_1)) = S_1 and ICEXT(*I*(C_2)) = S_2, respectively, according to Section 5.2.

Section 5.2 also specifies semantic conditions for all the *datatypes* of the OWL 2 RDF-Based Semantics, as listed in Section 3.3. For each datatype IRI *E*, such as *E* := "`xsd:string`", for the semantic conditions "$I(E) \in$ IDC" and "ICEXT($I(E)$) ⊆ LV" the corresponding axiomatic triples are of the form

```
E rdf:type rdfs:Datatype .
E rdfs:subClassOf rdfs:Literal .
```

In analogy to Section 6.1 for the RDF axiomatic triples, all the axiomatic triples for the vocabulary classes (including datatypes) can be considered "*simple*" in the sense that they will have in their object position only single terms from the RDF, RDFS and OWL 2 RDF-Based vocabularies (Section 3.2).

Note that some of the axiomatic triples obtained in this way already follow from the semantics of RDF and RDFS, as defined in the RDF Semantics [*RDF Semantics*].

6.3 Axiomatic Triples for the Vocabulary Properties

The semantic conditions for *vocabulary properties* in Section 5.3 can be considered as corresponding to a set of axiomatic triples for the properties in the vocabulary of the OWL 2 RDF-Based Semantics.

First, for each IRI *E* occurring in the first column of Table 5.3, if the *second* column contains an entry of the form "$I(E) \in S$" for some set *S*, then this entry corresponds to an RDF triple of the form "*E* `rdf:type` *C*", where *C* is the IRI of a vocabulary class with ICEXT($I(C)$) = *S*. In the table, *S* will always be either the part IP of all properties, or some sub part of IP. Hence, in a corresponding RDF triple the IRI *C* will be one of "`rdf:Property`", "`owl:ObjectProperty`", (*S*=IP in both cases), "`owl:DatatypeProperty`" (*S*=IODP), "`owl:OntologyProperty`" (*S*=IOXP) or "`owl:AnnotationProperty`" (*S*=IOAP).

For example, for the IRI "`owl:disjointWith`", the semantic condition

I(`owl:disjointWith`) ∈ IP

has the corresponding axiomatic triple

```
owl:disjointWith rdf:type rdf:Property .
```

Further, for each IRI *E* in the first column of the table, if the *third* column contains an entry of the form "IEXT($I(E)$) ⊆ $S_1 \times S_2$" for some sets S_1 and S_2, then this entry corresponds to RDF triples of the form "*E* `rdfs:domain` C_1" and "*E* `rdfs:range` C_2", where C_1 and C_2 are the IRIs of vocabulary classes with ICEXT($I(C_1)$) = S_1 and ICEXT($I(C_2)$) = S_2, respectively. Note that the sets S_1 and S_2 do *not always* correspond to any of the parts of the universe of *I*.

For example, the semantic condition

IEXT(*I*(`owl:disjointWith`)) ⊆ IC × IC

has the corresponding axiomatic triples

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

W3C Recommendation

Exceptions are the semantic conditions "IEXT(*I*(`owl:topObjectProperty`)) = IR × IR" and "IEXT(*I*(`owl:topDataProperty`)) = IR × LV", since the *exactly* specified property extensions of these properties cannot be expressed solely by domain and range axiomatic triples. For example, the domain and range axiomatic triples for `owl:sameAs` are equal to those for `owl:topObjectProperty`, but the property extension of `owl:sameAs` is different from the property extension of `owl:topObjectProperty`.

Section 5.3 also specifies semantic conditions for all the *facets* of the OWL 2 RDF-Based Semantics, as listed in Section 3.4. For each facet IRI *E*, such as *E* := "`xsd:length`", for the semantic conditions "*I*(*E*) ∈ IODP" and "IEXT(*I*(*E*)) ⊆ IR × LV" the corresponding axiomatic triples are of the form

```
E rdf:type owl:DatatypeProperty .
E rdfs:domain rdfs:Resource .
E rdfs:range rdfs:Literal .
```

In analogy to Section 6.1 for the RDF axiomatic triples, all the axiomatic triples for the vocabulary properties (including facets) can be considered "*simple*" in the sense that they will have in their object position only single terms from the RDF, RDFS and OWL 2 RDF-Based vocabularies (Section 3.2).

6.4 A Set of Axiomatic Triples

This section provides a concrete example set of axiomatic triples based on the discussion in the Sections 6.2 and 6.3. The axiomatic triples are grouped by different tables for the classes and the properties of the OWL 2 RDF-Based vocabulary, for the datatypes and the facets of the OWL 2 RDF-Based Semantics, and for some of the classes and properties of the RDFS vocabulary. Note that this set of axiomatic triples is not meant to be free of redundancy.

Table 6.1: Axiomatic Triples for the Classes of the OWL 2 RDF-Based Vocabulary

`owl:AllDifferent rdf:type rdfs:Class .` `owl:AllDifferent rdfs:subClassOf rdfs:Resource .`	`owl:AllDisjointClasses rdf:type rdfs:Class .` `owl:AllDisjointClasses rdfs:subClassOf rdfs:Resource .`
`owl:AllDisjointProperties rdf:type rdfs:Class .` `owl:AllDisjointProperties rdfs:subClassOf rdfs:Resource .`	`owl:Annotation rdf:type rdfs:Class .` `owl:Annotation rdfs:subClassOf rdfs:Resource .`
`owl:AnnotationProperty rdf:type rdfs:Class .` `owl:AnnotationProperty rdfs:subClassOf rdf:Property .`	`owl:AsymmetricProperty rdf:type rdfs:Class .` `owl:AsymmetricProperty rdfs:subClassOf owl:ObjectProperty .`
`owl:Axiom rdf:type rdfs:Class .` `owl:Axiom rdfs:subClassOf rdfs:Resource .`	`owl:Class rdf:type rdfs:Class .` `owl:Class rdfs:subClassOf rdfs:Class .`
`owl:DataRange rdf:type rdfs:Class .`	`owl:DatatypeProperty rdf:type rdfs:Class .`

owl:DataRange rdfs:subClassOf rdfs:Datatype .	owl:DatatypeProperty rdfs:subClassOf rdf:Property .
owl:DeprecatedClass rdf:type rdfs:Class . owl:DeprecatedClass rdfs:subClassOf rdfs:Class .	owl:DeprecatedProperty rdf:type rdfs:Class . owl:DeprecatedProperty rdfs:subClassOf rdf:Property .
owl:FunctionalProperty rdf:type rdfs:Class . owl:FunctionalProperty rdfs:subClassOf rdf:Property .	owl:InverseFunctionalProperty rdf:type rdfs:Class . owl:InverseFunctionalProperty rdfs:subClassOf owl:ObjectProperty .
owl:IrreflexiveProperty rdf:type rdfs:Class . owl:IrreflexiveProperty rdfs:subClassOf owl:ObjectProperty .	owl:NamedIndividual rdf:type rdfs:Class . owl:NamedIndividual rdfs:subClassOf owl:Thing .
owl:NegativePropertyAssertion rdf:type rdfs:Class . owl:NegativePropertyAssertion rdfs:subClassOf rdfs:Resource .	owl:Nothing rdf:type owl:Class . owl:Nothing rdfs:subClassOf owl:Thing .
owl:ObjectProperty rdf:type rdfs:Class . owl:ObjectProperty rdfs:subClassOf rdf:Property .	owl:Ontology rdf:type rdfs:Class . owl:Ontology rdfs:subClassOf rdfs:Resource .
owl:OntologyProperty rdf:type rdfs:Class . owl:OntologyProperty rdfs:subClassOf rdf:Property .	owl:ReflexiveProperty rdf:type rdfs:Class . owl:ReflexiveProperty rdfs:subClassOf owl:ObjectProperty .
owl:Restriction rdf:type rdfs:Class . owl:Restriction rdfs:subClassOf owl:Class .	owl:SymmetricProperty rdf:type rdfs:Class . owl:SymmetricProperty rdfs:subClassOf owl:ObjectProperty .
owl:Thing rdf:type owl:Class .	owl:TransitiveProperty rdf:type rdfs:Class . owl:TransitiveProperty rdfs:subClassOf owl:ObjectProperty .

Table 6.2: Axiomatic Triples for the Properties of the OWL 2 RDF-Based Vocabulary

owl:allValuesFrom rdf:type rdf:Property . owl:allValuesFrom rdfs:domain owl:Restriction . owl:allValuesFrom rdfs:range rdfs:Class .	owl:annotatedProperty rdf:type rdf:Property . owl:annotatedProperty rdfs:domain rdfs:Resource . owl:annotatedProperty rdfs:range rdfs:Resource .
owl:annotatedSource rdf:type rdf:Property .	owl:annotatedTarget rdf:type rdf:Property .

W3C Recommendation

owl:annotatedSource rdfs:domain rdfs:Resource . owl:annotatedSource rdfs:range rdfs:Resource .	owl:annotatedTarget rdfs:domain rdfs:Resource . owl:annotatedTarget rdfs:range rdfs:Resource .
owl:assertionProperty rdf:type rdf:Property . owl:assertionProperty rdfs:domain owl:NegativePropertyAssertion . owl:assertionProperty rdfs:range rdf:Property .	owl:backwardCompatibleWith rdf:type owl:AnnotationProperty . owl:backwardCompatibleWith rdf:type owl:OntologyProperty . owl:backwardCompatibleWith rdfs:domain owl:Ontology . owl:backwardCompatibleWith rdfs:range owl:Ontology .
owl:bottomDataProperty rdf:type owl:DatatypeProperty . owl:bottomDataProperty rdfs:domain owl:Thing . owl:bottomDataProperty rdfs:range rdfs:Literal .	owl:bottomObjectProperty rdf:type owl:ObjectProperty . owl:bottomObjectProperty rdfs:domain owl:Thing . owl:bottomObjectProperty rdfs:range owl:Thing .
owl:cardinality rdf:type rdf:Property . owl:cardinality rdfs:domain owl:Restriction . owl:cardinality rdfs:range xsd:nonNegativeInteger .	owl:complementOf rdf:type rdf:Property . owl:complementOf rdfs:domain owl:Class . owl:complementOf rdfs:range owl:Class .
owl:datatypeComplementOf rdf:type rdf:Property . owl:datatypeComplementOf rdfs:domain rdfs:Datatype . owl:datatypeComplementOf rdfs:range rdfs:Datatype .	owl:deprecated rdf:type owl:AnnotationProperty . owl:deprecated rdfs:domain rdfs:Resource . owl:deprecated rdfs:range rdfs:Resource .
owl:differentFrom rdf:type rdf:Property . owl:differentFrom rdfs:domain owl:Thing . owl:differentFrom rdfs:range owl:Thing .	owl:disjointUnionOf rdf:type rdf:Property . owl:disjointUnionOf rdfs:domain owl:Class . owl:disjointUnionOf rdfs:range rdf:List .
owl:disjointWith rdf:type rdf:Property . owl:disjointWith rdfs:domain owl:Class . owl:disjointWith rdfs:range owl:Class .	owl:distinctMembers rdf:type rdf:Property . owl:distinctMembers rdfs:domain owl:AllDifferent . owl:distinctMembers rdfs:range rdf:List .
owl:equivalentClass rdf:type rdf:Property . owl:equivalentClass rdfs:domain rdfs:Class . owl:equivalentClass rdfs:range rdfs:Class .	owl:equivalentProperty rdf:type rdf:Property . owl:equivalentProperty rdfs:domain rdf:Property . owl:equivalentProperty rdfs:range rdf:Property .

owl:hasKey rdf:type rdf:Property . owl:hasKey rdfs:domain owl:Class . owl:hasKey rdfs:range rdf:List .	owl:hasSelf rdf:type rdf:Property . owl:hasSelf rdfs:domain owl:Restriction . owl:hasSelf rdfs:range rdfs:Resource .
owl:hasValue rdf:type rdf:Property . owl:hasValue rdfs:domain owl:Restriction . owl:hasValue rdfs:range rdfs:Resource .	owl:imports rdf:type owl:OntologyProperty . owl:imports rdfs:domain owl:Ontology . owl:imports rdfs:range owl:Ontology .
owl:incompatibleWith rdf:type owl:AnnotationProperty . owl:incompatibleWith rdf:type owl:OntologyProperty . owl:incompatibleWith rdfs:domain owl:Ontology . owl:incompatibleWith rdfs:range owl:Ontology .	owl:intersectionOf rdf:type rdf:Property . owl:intersectionOf rdfs:domain rdfs:Class . owl:intersectionOf rdfs:range rdf:List .
owl:inverseOf rdf:type rdf:Property . owl:inverseOf rdfs:domain owl:ObjectProperty . owl:inverseOf rdfs:range owl:ObjectProperty .	owl:maxCardinality rdf:type rdf:Property . owl:maxCardinality rdfs:domain owl:Restriction . owl:maxCardinality rdfs:range xsd:nonNegativeInteger .
owl:maxQualifiedCardinality rdf:type rdf:Property . owl:maxQualifiedCardinality rdfs:domain owl:Restriction . owl:maxQualifiedCardinality rdfs:range xsd:nonNegativeInteger .	owl:members rdf:type rdf:Property . owl:members rdfs:domain rdfs:Resource . owl:members rdfs:range rdf:List .
owl:minCardinality rdf:type rdf:Property . owl:minCardinality rdfs:domain owl:Restriction . owl:minCardinality rdfs:range xsd:nonNegativeInteger .	owl:minQualifiedCardinality rdf:type rdf:Property . owl:minQualifiedCardinality rdfs:domain owl:Restriction . owl:minQualifiedCardinality rdfs:range xsd:nonNegativeInteger .
owl:onClass rdf:type rdf:Property . owl:onClass rdfs:domain owl:Restriction . owl:onClass rdfs:range owl:Class .	owl:onDataRange rdf:type rdf:Property . owl:onDataRange rdfs:domain owl:Restriction . owl:onDataRange rdfs:range rdfs:Datatype .
owl:onDatatype rdf:type rdf:Property . owl:onDatatype rdfs:domain rdfs:Datatype .	owl:oneOf rdf:type rdf:Property . owl:oneOf rdfs:domain rdfs:Class . owl:oneOf rdfs:range rdf:List .

W3C Recommendation

owl:onDatatype rdfs:range rdfs:Datatype .	
owl:onProperty rdf:type rdf:Property . owl:onProperty rdfs:domain owl:Restriction . owl:onProperty rdfs:range rdf:Property .	owl:onProperties rdf:type rdf:Property . owl:onProperties rdfs:domain owl:Restriction . owl:onProperties rdfs:range rdf:List .
owl:priorVersion rdf:type owl:AnnotationProperty . owl:priorVersion rdf:type owl:OntologyProperty . owl:priorVersion rdfs:domain owl:Ontology . owl:priorVersion rdfs:range owl:Ontology .	owl:propertyChainAxiom rdf:type rdf:Property . owl:propertyChainAxiom rdfs:domain owl:ObjectProperty . owl:propertyChainAxiom rdfs:range rdf:List .
owl:propertyDisjointWith rdf:type rdf:Property . owl:propertyDisjointWith rdfs:domain rdf:Property . owl:propertyDisjointWith rdfs:range rdf:Property .	owl:qualifiedCardinality rdf:type rdf:Property . owl:qualifiedCardinality rdfs:domain owl:Restriction . owl:qualifiedCardinality rdfs:range xsd:nonNegativeInteger .
owl:sameAs rdf:type rdf:Property . owl:sameAs rdfs:domain owl:Thing . owl:sameAs rdfs:range owl:Thing .	owl:someValuesFrom rdf:type rdf:Property . owl:someValuesFrom rdfs:domain owl:Restriction . owl:someValuesFrom rdfs:range rdfs:Class .
owl:sourceIndividual rdf:type rdf:Property . owl:sourceIndividual rdfs:domain owl:NegativePropertyAssertion . owl:sourceIndividual rdfs:range owl:Thing .	owl:targetIndividual rdf:type rdf:Property . owl:targetIndividual rdfs:domain owl:NegativePropertyAssertion . owl:targetIndividual rdfs:range owl:Thing .
owl:targetValue rdf:type rdf:Property . owl:targetValue rdfs:domain owl:NegativePropertyAssertion . owl:targetValue rdfs:range rdfs:Literal .	owl:topDataProperty rdf:type owl:DatatypeProperty . owl:topDataProperty rdfs:domain owl:Thing . owl:topDataProperty rdfs:range rdfs:Literal .
owl:topObjectProperty rdf:type rdf:ObjectProperty . owl:topObjectProperty rdfs:domain owl:Thing . owl:topObjectProperty rdfs:range owl:Thing .	owl:unionOf rdf:type rdf:Property . owl:unionOf rdfs:domain rdfs:Class . owl:unionOf rdfs:range rdf:List .

W3C Recommendation

owl:versionInfo rdf:type owl:AnnotationProperty . owl:versionInfo rdfs:domain rdfs:Resource . owl:versionInfo rdfs:range rdfs:Resource .	owl:versionIRI rdf:type owl:OntologyProperty . owl:versionIRI rdfs:domain owl:Ontology . owl:versionIRI rdfs:range owl:Ontology .
owl:withRestrictions rdf:type rdf:Property . owl:withRestrictions rdfs:domain rdfs:Datatype . owl:withRestrictions rdfs:range rdf:List .	

Table 6.3: Axiomatic Triples for the Datatypes of the OWL 2 RDF-Based Semantics

xsd:anyURI rdf:type rdfs:Datatype . xsd:anyURI rdfs:subClassOf rdfs:Literal .	xsd:base64Binary rdf:type rdfs:Datatype . xsd:base64Binary rdfs:subClassOf rdfs:Literal .
xsd:boolean rdf:type rdfs:Datatype . xsd:boolean rdfs:subClassOf rdfs:Literal .	xsd:byte rdf:type rdfs:Datatype . xsd:byte rdfs:subClassOf rdfs:Literal .
xsd:dateTime rdf:type rdfs:Datatype . xsd:dateTime rdfs:subClassOf rdfs:Literal .	xsd:dateTimeStamp rdf:type rdfs:Datatype . xsd:dateTimeStamp rdfs:subClassOf rdfs:Literal .
xsd:decimal rdf:type rdfs:Datatype . xsd:decimal rdfs:subClassOf rdfs:Literal .	xsd:double rdf:type rdfs:Datatype . xsd:double rdfs:subClassOf rdfs:Literal .
xsd:float rdf:type rdfs:Datatype . xsd:float rdfs:subClassOf rdfs:Literal .	xsd:hexBinary rdf:type rdfs:Datatype . xsd:hexBinary rdfs:subClassOf rdfs:Literal .
xsd:int rdf:type rdfs:Datatype . xsd:int rdfs:subClassOf rdfs:Literal .	xsd:integer rdf:type rdfs:Datatype . xsd:integer rdfs:subClassOf rdfs:Literal .
xsd:language rdf:type rdfs:Datatype . xsd:language rdfs:subClassOf rdfs:Literal .	xsd:long rdf:type rdfs:Datatype . xsd:long rdfs:subClassOf rdfs:Literal .
xsd:Name rdf:type rdfs:Datatype . xsd:Name rdfs:subClassOf rdfs:Literal .	xsd:NCName rdf:type rdfs:Datatype . xsd:NCName rdfs:subClassOf rdfs:Literal .
xsd:negativeInteger rdf:type rdfs:Datatype . xsd:negativeInteger rdfs:subClassOf rdfs:Literal .	xsd:NMTOKEN rdf:type rdfs:Datatype . xsd:NMTOKEN rdfs:subClassOf rdfs:Literal .

xsd:nonNegativeInteger rdf:type rdfs:Datatype . xsd:nonNegativeInteger rdfs:subClassOf rdfs:Literal .	xsd:nonPositiveInteger rdf:type rdfs:Datatype . xsd:nonPositiveInteger rdfs:subClassOf rdfs:Literal .
xsd:normalizedString rdf:type rdfs:Datatype . xsd:normalizedString rdfs:subClassOf rdfs:Literal .	rdf:PlainLiteral rdf:type rdfs:Datatype . rdf:PlainLiteral rdfs:subClassOf rdfs:Literal .
xsd:positiveInteger rdf:type rdfs:Datatype . xsd:positiveInteger rdfs:subClassOf rdfs:Literal .	owl:rational rdf:type rdfs:Datatype . owl:rational rdfs:subClassOf rdfs:Literal .
owl:real rdf:type rdfs:Datatype . owl:real rdfs:subClassOf rdfs:Literal .	xsd:short rdf:type rdfs:Datatype . xsd:short rdfs:subClassOf rdfs:Literal .
xsd:string rdf:type rdfs:Datatype . xsd:string rdfs:subClassOf rdfs:Literal .	xsd:token rdf:type rdfs:Datatype . xsd:token rdfs:subClassOf rdfs:Literal .
xsd:unsignedByte rdf:type rdfs:Datatype . xsd:unsignedByte rdfs:subClassOf rdfs:Literal .	xsd:unsignedInt rdf:type rdfs:Datatype . xsd:unsignedInt rdfs:subClassOf rdfs:Literal .
xsd:unsignedLong rdf:type rdfs:Datatype . xsd:unsignedLong rdfs:subClassOf rdfs:Literal .	xsd:unsignedShort rdf:type rdfs:Datatype . xsd:unsignedShort rdfs:subClassOf rdfs:Literal .
rdf:XMLLiteral rdf:type rdfs:Datatype . rdf:XMLLiteral rdfs:subClassOf rdfs:Literal .	

Table 6.4: Axiomatic Triples for the Facets of the OWL 2 RDF-Based Semantics

rdf:langRange rdf:type owl:DatatypeProperty . rdf:langRange rdfs:domain rdfs:Resource . rdf:langRange rdfs:range rdfs:Literal .	xsd:length rdf:type owl:DatatypeProperty . xsd:length rdfs:domain rdfs:Resource . xsd:length rdfs:range rdfs:Literal .
xsd:maxExclusive rdf:type owl:DatatypeProperty . xsd:maxExclusive rdfs:domain rdfs:Resource . xsd:maxExclusive rdfs:range rdfs:Literal .	xsd:maxInclusive rdf:type owl:DatatypeProperty . xsd:maxInclusive rdfs:domain rdfs:Resource . xsd:maxInclusive rdfs:range rdfs:Literal .
xsd:maxLength rdf:type owl:DatatypeProperty .	xsd:minExclusive rdf:type owl:DatatypeProperty .

xsd:maxLength rdfs:domain rdfs:Resource . xsd:maxLength rdfs:range rdfs:Literal .	xsd:minExclusive rdfs:domain rdfs:Resource . xsd:minExclusive rdfs:range rdfs:Literal .
xsd:minInclusive rdf:type owl:DatatypeProperty . xsd:minInclusive rdfs:domain rdfs:Resource . xsd:minInclusive rdfs:range rdfs:Literal .	xsd:minLength rdf:type owl:DatatypeProperty . xsd:minLength rdfs:domain rdfs:Resource . xsd:minLength rdfs:range rdfs:Literal .
xsd:pattern rdf:type owl:DatatypeProperty . xsd:pattern rdfs:domain rdfs:Resource . xsd:pattern rdfs:range rdfs:Literal .	

Table 6.5: Additional Axiomatic Triples for Classes and Properties of the RDFS Vocabulary

rdfs:Class rdfs:subClassOf owl:Class .	rdfs:comment rdf:type owl:AnnotationProperty . rdfs:comment rdfs:domain rdfs:Resource . rdfs:comment rdfs:range rdfs:Literal .
rdfs:Datatype rdfs:subClassOf owl:DataRange .	rdfs:isDefinedBy rdf:type owl:AnnotationProperty . rdfs:isDefinedBy rdfs:domain rdfs:Resource . rdfs:isDefinedBy rdfs:range rdfs:Resource .
rdfs:label rdf:type owl:AnnotationProperty . rdfs:label rdfs:domain rdfs:Resource . rdfs:label rdfs:range rdfs:Literal .	rdfs:Literal rdf:type rdfs:Datatype .
rdf:Property rdfs:subClassOf owl:ObjectProperty .	rdfs:Resource rdfs:subClassOf owl:Thing .
rdfs:seeAlso rdf:type owl:AnnotationProperty . rdfs:seeAlso rdfs:domain rdfs:Resource . rdfs:seeAlso rdfs:range rdfs:Resource .	

W3C Recommendation

7 Appendix: Relationship to the Direct Semantics (Informative)

This section compares the OWL 2 RDF-Based Semantics with the *OWL 2 Direct Semantics* [*OWL 2 Direct Semantics*]. While the OWL 2 RDF-Based Semantics is based on the RDF Semantics specification [*RDF Semantics*], the OWL 2 Direct Semantics is a *description logic* style semantics. Several fundamental differences exist between the two semantics, but there is also a strong relationship basically stating that the OWL 2 RDF-Based Semantics is able to reflect all logical conclusions of the OWL 2 Direct Semantics. This means that the OWL 2 Direct Semantics can in a sense be regarded as a semantics subset of the OWL 2 RDF-Based Semantics.

Technically, the comparison will be performed by comparing the sets of *entailments* that hold for each of the two semantics, respectively. The definition of an ***OWL 2 RDF-Based entailment*** was given in Section 4.3 of this document, while the definition of an ***OWL 2 Direct entailment*** is provided in Section 2.5 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*]. In both cases, entailments are defined for pairs of ontologies, and such an ordered pair of two ontologies will be called an ***entailment query*** in this section.

Comparing the two semantics by means of entailments will only be meaningful if the entailment queries allow for applying both the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics to them. In order to ensure this, the comparison will be restricted to entailment queries, for which the left-hand side and right-hand side ontologies are both ***OWL 2 DL ontologies in RDF graph form***. These are RDF graphs that, by applying the *reverse RDF mapping* [*OWL 2 RDF Mapping*], can be transformed into corresponding ***OWL 2 DL ontologies in Functional Syntax form*** according to the *functional style syntax* defined in the OWL 2 Structural Specification [*OWL 2 Specification*], and which must further meet all the *restrictions on OWL 2 DL ontologies* that are specified in Section 3 of the OWL 2 Structural Specification [*OWL 2 Specification*]. In fact, these restrictions must be *mutually met* by both ontologies that occur in an entailment query, i.e. all these restrictions need to be satisfied as if the two ontologies would be part of a single ontology. Any entailment query that adheres to the conditions defined here will be called an ***OWL 2 DL entailment query***.

Ideally, the relationship between the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics would be of the form that every OWL 2 DL entailment query that is an OWL 2 Direct entailment is also an OWL 2 RDF-Based entailment. However, this desirable relationship cannot hold in general due to a variety of differences that exist between the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics, as demonstrated in Section 7.1.

Fortunately, the problems resulting from these semantic differences can be overcome in a way that for every OWL 2 DL entailment query there is another one for which the desired entailment relationship indeed holds, and the new entailment query is semantically equivalent to the original entailment query under the OWL 2 Direct Semantics. This is the gist of the *OWL 2 correspondence theorem*, which will be presented in Section 7.2. The *proof* of this theorem, given in Section 7.3, will further demonstrate that such a substitute OWL 2 DL entailment query can always be algorithmically constructed by means of simple syntactic transformations.

W3C Recommendation

7.1 Example on Semantic Differences

This section will show that differences exist between the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics, and it will be demonstrated how these semantic differences complicate a comparison of the two semantics in terms of entailments. An example OWL 2 DL entailment query will be given, which will happen to be an OWL 2 Direct entailment without being an OWL 2 RDF-Based entailment. The section will explain the different reasons and will provide a resolution of each of them. It will turn out that the example entailment query can be syntactically transformed into another OWL 2 DL entailment query that is both an OWL 2 Direct entailment and an OWL 2 RDF-Based entailment, while being semantically unchanged compared to the original entailment query under the OWL 2 Direct Semantics. This example will motivate the *OWL 2 correspondence theorem* in Section 7.2 and its proof in Section 7.3.

The example entailment query consists of the following pair (G_1^* , G_2^*) of RDF graphs:

G_1^* :

```
(1) ex:o1 rdf:type owl:Ontology .
(2) ex:c1 rdf:type owl:Class .
(3) ex:c2 rdf:type owl:Class .
(4) ex:c1 rdfs:subClassOf ex:c2 .
```

G_2^* :

```
(1) ex:o2 rdf:type owl:Ontology .
(2) ex:c1 rdf:type owl:Class .
(3) ex:c2 rdf:type owl:Class .
(4) ex:c3 rdf:type owl:Class .
(5) ex:c1 rdfs:subClassOf _:x .
(6) _:x rdf:type owl:Class .
(7) _:x owl:unionOf ( ex:c2 ex:c3 ) .
(8) ex:c3 rdfs:label "c3" .
```

Both G_1^* and G_2^* are OWL 2 DL ontologies in RDF graph form and can therefore be mapped by the reverse RDF mapping [*OWL 2 RDF Mapping*] to the following two OWL 2 DL ontologies in Functional Syntax form F(G_1^*) and F(G_2^*):

F(G_1^*) :

```
(1) Ontology( ex:o1
(2)   Declaration( Class( ex:c1 ) )
(3)   Declaration( Class( ex:c2 ) )
(4)   SubClassOf( ex:c1 ex:c2 )
(5) )
```

F(G_2^*) :

```
(1) Ontology( ex:o2
(2)   Declaration( Class( ex:c1 ) )
(3)   Declaration( Class( ex:c2 ) )
```

```
(4)   Declaration( Class( ex:c3 ) )
(5)   SubClassOf( ex:c1 ObjectUnionOf( ex:c2 ex:c3 ) )
(6)   AnnotationAssertion( rdfs:label ex:c3 "c3" )
(7) )
```

Note that F(G_1^*) and F(G_2^*) mutually meet the restrictions on OWL 2 DL ontologies as specified in Section 3 of the OWL 2 Structural Specification [*OWL 2 Specification*]. For example, none of the IRIs being declared as a class in F(G_1^*) is declared as a datatype in F(G_2^*), since this would not be allowed for an OWL 2 DL entailment query.

It follows that F(G_1^*) OWL 2 Direct entails F(G_2^*). To show this, only the axioms (4) of F(G_1^*) and (5) of F(G_2^*) have to be considered. None of the other statements in the two ontologies are relevant for this OWL 2 Direct entailment to hold, since they do not have a formal meaning under the OWL 2 Direct Semantics. However, it turns out that the RDF graph G_1^* does *not* OWL 2 RDF-Based entail G_2^*, for reasons discussed in detail now.

Reason 1: An Annotation in F(G_2^*). The ontology F(G_2^*) contains an annotation (6). The OWL 2 Direct Semantics does not give a formal meaning to annotations. In contrast, under the OWL 2 RDF-Based Semantics *every* RDF triple occurring in an RDF graph has a formal meaning, including the corresponding annotation triple (8) in G_2^*. Since this annotation triple only occurs in G_2^* but not in G_1^*, there will exist OWL 2 RDF-Based interpretations that satisfy G_1^* without satisfying triple (8) of G_2^*. Hence, G_1^* does *not* OWL 2 RDF-Based entail G_2^*.

Resolution of Reason 1. The annotation triple (8) in G_2^* will be removed, which will avoid requiring OWL 2 RDF-Based interpretations to interpret this triple. The changed RDF graphs will still be OWL 2 DL ontologies in RDF graph form, since annotations are strictly optional in OWL 2 DL ontologies and may therefore be omitted. Also, this operation will not change the formal meaning of the ontologies under the OWL 2 Direct Semantics, since annotations do not have a formal meaning under this semantics.

Reason 2: An Entity Declaration exclusively in F(G_2^*). The ontology F(G_2^*) contains an entity declaration for the class IRI `ex:c3` (4), for which there is no corresponding entity declaration in F(G_1^*). The OWL 2 Direct Semantics does not give a formal meaning to entity declarations, while the OWL 2 RDF-Based Semantics gives a formal meaning to the corresponding declaration statement (4) in G_2^*. The consequences are analog to those described for reason 1.

Resolution of Reason 2. The declaration statement (4) in G_2^* will be copied to G_1^*. An OWL 2 RDF-Based interpretation that satisfies the modified graph G_1^* will then also satisfy the declaration statement. The changed RDF graphs will still be OWL 2 DL ontologies in RDF graph form, since the copied declaration statement is not in conflict with any of the other entity declarations in G_1^*. Also, this operation will not change the formal meaning of the ontologies under the OWL 2 Direct Semantics, since entity declarations do not have a formal meaning under this semantics.

Reason 3: Different Ontology IRIs in F(G_1^*) and F(G_2^*). The ontology IRIs for the two ontologies, given by (1) in F(G_1^*) and by (1) in F(G_2^*), differ from each other. The OWL 2 Direct Semantics does not give a formal meaning to ontology headers, while the OWL 2 RDF-Based Semantics gives a formal meaning to the corresponding header triples (1) in G_1^* and (1) in G_2^*. Since these header triples differ from each other, the consequences are analog to those described for reason 1.

Resolution of Reason 3. The IRI in the subject position of the header triple (1) in G_2^* is changed into a blank node. Due to the existential semantics of blank nodes under the OWL 2 RDF-Based Semantics the resulting triple will then be entailed by triple (1) in G_1^*. The changed RDF graphs will still be OWL 2 DL ontologies in RDF graph form, since an ontology IRI is optional for an OWL 2 DL ontology. (Note, however, that it would have been an error to simply remove triple (1) from G_2^*, since an OWL 2 DL ontology is required to contain an ontology header.) Also, this operation will not change the formal meaning of the ontologies under the OWL 2 Direct Semantics, since ontology headers do not have a formal meaning under this semantics.

Reason 4: A Class Expression in F(G_2^*). Axiom (5) of F(G_2^*) contains a class expression that represents the union of the two classes denoted by `ex:c2` and `ex:c3`. Within G_2^*, this class expression is represented by the triples (6) and (7), both having the blank node "`_:x`" in their respective subject position. The way the OWL 2 RDF-Based Semantics interprets these two triples differs from the way the OWL 2 Direct Semantics treats the class expression in axiom (5) of F(G_2^*).

The OWL 2 Direct Semantics treats classes as *sets*, i.e. subsets of the universe. Thus, the IRIs `ex:c2` and `ex:c3` in F(G_2^*) denote two sets, and the class expression in axiom (5) of F(G_2^*) therefore represents the set that consists of the union of these two sets.

The OWL 2 RDF-Based Semantics, on the other hand, treats classes as *individuals*, i.e. members of the universe. While every class under the OWL 2 RDF-Based Semantics represents a certain subset of the universe, namely its class extension, this set is actually distinguished from the class itself. For two given classes it is ensured under the OWL 2 RDF-Based Semantics, just as for the OWL 2 Direct Semantics, that the union of their class extensions will always exist as a subset of the universe. However, there is no guarantee that there will also exist an individual in the universe that has this set union as its class extension.

Under the OWL 2 RDF-Based Semantics, triple (7) of G_2^* essentially claims that a class exists being the union of two other classes. But since the existence of such a union class is not ensured by G_1^*, there will be OWL 2 RDF-Based interpretations that satisfy G_1^* without satisfying triple (7) of G_2^*. Hence, G_1^* does *not* OWL 2 RDF-Based entail G_2^*.

Resolution of Reason 4. The triples (6) and (7) of G_2^* are copied to G_1^* together with the new triple "`_:x owl:equivalentClass _:x`". In addition, for the IRI `ex:c3`, which only occurs in the union class expression but not in G_1^*, an entity declaration is added to G_1^* by the resolution of reason 2. If an OWL 2 RDF-Based interpretation satisfies the modified graph G_1^*, then the triples (6) and (7) of G_2^* will now be satisfied. The changed RDF graphs will still be OWL 2 DL ontologies in RDF graph form, since the whole set of added triples validly encodes an OWL 2 axiom, and since none of the restrictions on OWL 2 DL ontologies is hurt. Also, this operation will not change the formal meaning of the ontologies under the OWL 2 Direct Semantics, since the added equivalence axiom is a tautology under this semantics.

Note that it would have been an error to simply copy the triples (6) and (7) of G_2^* to G_1^*, without also adding the new triple "`_:x owl:equivalentClass _:x`". This would have produced a class expression that has no connection to any axiom in the ontology. An OWL 2 DL ontology is basically a set of axioms and does not allow for the occurrence of "dangling" class expressions. This is the reason for actually "embedding" the class expression in an axiom. It would have also been wrong to use an *arbitrary* axiom for such an embedding, since it has to be ensured that the formal meaning of the original ontology does not change under the OWL 2 Direct Semantics. However, any *tautological* axiom that contains the original class expression would have been sufficient for this purpose as well.

Complete Resolution: The Transformed Entailment Query.

Combining the resolutions of all the above reasons leads to the following new pair of RDF graphs (G_1 , G_2):

G_1 :

```
(1) ex:o1 rdf:type owl:Ontology .
(2) ex:c1 rdf:type owl:Class .
(3) ex:c2 rdf:type owl:Class .
(4) ex:c3 rdf:type owl:Class .
(5) ex:c1 rdfs:subClassOf ex:c2 .
(6) _:x owl:equivalentClass _:x .
(7) _:x rdf:type owl:Class .
(8) _:x owl:unionOf ( ex:c2 ex:c3 ) .
```

G_2 :

```
(1) _:o rdf:type owl:Ontology .
(2) ex:c1 rdf:type owl:Class .
(3) ex:c2 rdf:type owl:Class .
(4) ex:c3 rdf:type owl:Class .
(5) ex:c1 rdfs:subClassOf _:x .
(6) _:x rdf:type owl:Class .
(7) _:x owl:unionOf ( ex:c2 ex:c3 ) .
```

The following list reiterates the changes compared to the original RDF graphs G_1^* and G_2^*:

- ***Resolution of Reason 1 (Annotation):*** Triple (8) in G_2^* has been removed, i.e. there is no corresponding annotation triple in G_2.
- ***Resolution of Reason 2 (Entity Declaration):*** Triple (4) in G_2^* has been copied to G_1^*, becoming triple (4) in G_1.
- ***Resolution of Reason 3 (Ontology IRIs):*** The IRI in the subject position of triple (1) in G_2^* has been changed into a blank node, becoming triple (1) in G_2.
- ***Resolution of Reason 4 (Class Expression):*** Triples (6) and (7) in G_2^* have been copied to G_1^* together with the new triple "`_:x owl:equivalentClass _:x`", becoming triples (6), (7) and (8) in G_1.

W3C Recommendation

G_1 and G_2 are again OWL 2 DL ontologies in RDF graph form and can be mapped to the following OWL 2 DL ontologies in Functional Syntax form F(G_1) and F(G_2), which again mutually meet the restrictions on OWL 2 DL ontologies:

F(G_1) :

```
(1) Ontology( ex:o1
(2)   Declaration( Class( ex:c1 ) )
(3)   Declaration( Class( ex:c2 ) )
(4)   Declaration( Class( ex:c3 ) )
(5)   SubClassOf( ex:c1 ex:c2 )
(6)   EquivalentClasses( ObjectUnionOf( ex:c2 ex:c3 ) ObjectUnionOf(
ex:c2 ex:c3 ) )
(7) )
```

F(G_2) :

```
(1) Ontology(
(2)   Declaration( Class( ex:c1 ) )
(3)   Declaration( Class( ex:c2 ) )
(4)   Declaration( Class( ex:c3 ) )
(5)   SubClassOf( ex:c1 ObjectUnionOf( ex:c2 ex:c3 ) )
(6) )
```

As said earlier, all the applied changes preserve the formal meaning of the original OWL 2 DL ontologies under the OWL 2 Direct Semantics. Hence, it is still the case that F(G_1) OWL 2 Direct entails F(G_2). However, due to the syntactic transformation the situation has changed for the OWL 2 RDF-Based Semantics: it is now possible to show, by following the lines of argumentation for the resolutions of the different reasons given above, that G_1 OWL 2 RDF-Based entails G_2 as well.

7.2 Correspondence Theorem

This section presents the *OWL 2 correspondence theorem*, which compares the semantic expressivity of the OWL 2 RDF-Based Semantics with that of the OWL 2 Direct Semantics. The theorem basically states that the OWL 2 RDF-Based Semantics is able to reflect all the semantic conclusions of the OWL 2 Direct Semantics, where the notion of a "semantic conclusion" is technically expressed in terms of an *entailment*.

However, as discussed in Section 7.1, there exist semantic differences between the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics, which do not allow for stating that *any* OWL 2 DL entailment query that is an OWL 2 Direct entailment will always also be an OWL 2 RDF-Based entailment. Nevertheless, it can still be ensured that any given OWL 2 DL entailment query can be *substituted* by another OWL 2 DL entailment query in a way that for the substitute entailment query the desired relationship will really hold, while preserving the formal meaning compared to the original entailment query under the OWL 2 Direct Semantics.

In fact, the theorem only makes the seemingly weak assertion that such a substitute entailment query will always *exist*. But the actual *proof for the theorem* in Section 7.3 will be more concrete in that it will substitute each given OWL 2 DL entailment query with a variant that can be algorithmically constructed by applying a set of simple syntactic transformations to the original entailment query. One can get an idea of how this works from Section 7.1.

Technical Note on Corresponding Datatype Maps. A distinction exists between the format of an *OWL 2 RDF-Based datatype map*, as defined by Definition 4.1, and the format of an *OWL 2 Direct datatype map*, as defined in Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*]. It is, however, possible to translate between an OWL 2 RDF-Based datatype map *D* and the corresponding OWL 2 Direct datatype map F(*D*) in the following way:

For an OWL 2 RDF-Based datatype map *D*, the *corresponding OWL 2 Direct datatype map* F(*D*) := (N_{DT} , N_{LS} , N_{FS} , $\cdot^{DT}$, $\cdot^{LS}$, $\cdot^{FS}$) [*OWL 2 Direct Semantics*] is given by

- *Datatype Names:* N_{DT} is defined as the set of all IRIs *u*, for which there is a datatype *d*, such that (*u* , *d*) ∈ *D*.
- *Lexical Space:* For each datatype name *u* ∈ N_{DT}, set $N_{LS}(u)$:= LS(*d*), where (*u* , *d*) ∈ *D*.
- *Facet Space:* For each datatype name *u* ∈ N_{DT}, set $N_{FS}(u)$:= FS(*d*), where (*u* , *d*) ∈ *D*.
- *Value Space:* For each datatype name *u* ∈ N_{DT}, set $(u)^{DT}$:= VS(*d*), where (*u* , *d*) ∈ *D*.
- *Lexical-to-Value Mapping:* For each datatype name *u* ∈ N_{DT} and each lexical form *a* ∈ $N_{LS}(u)$, set $(a , u)^{LS}$:= L2V(*d*)(*a*), where (*u* , *d*) ∈ *D*.
- *Facet-to-Value Mapping:* For each datatype name *u* ∈ N_{DT} and each facet-value pair (*F* , *v*) ∈ $N_{FS}(u)$, set $(F , v)^{FS}$:= F2V(*d*)((*F* , *v*)), where (*u* , *d*) ∈ *D*.

Theorem 7.1 (OWL 2 Correspondence Theorem):

Let *D* be an OWL 2 RDF-Based datatype map according to Definition 4.1, with F(*D*) being the OWL 2 Direct datatype map according to Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] that corresponds to *D* according to the *technical note on corresponding datatype maps*. Let G_1^* and G_2^* be RDF graphs that are OWL 2 DL ontologies in RDF graph form, with F(G_1^*) and F(G_2^*) being the OWL 2 DL ontologies in Functional Syntax form [*OWL 2 Specification*] that result from applying the reverse RDF mapping [*OWL 2 RDF Mapping*] to G_1^* and G_2^*, respectively. Let F(G_1^*) and F(G_2^*) mutually meet the restrictions on OWL 2 DL ontologies as specified in Section 3 of the OWL 2 Structural Specification [*OWL 2 Specification*].

Then, there exist RDF graphs G_1 and G_2 that are OWL 2 DL ontologies in RDF graph form, such that all the following relationships hold, with F(G_1) and F(G_2) being the OWL 2 DL ontologies in Functional Syntax form that result from applying the reverse RDF mapping to G_1 and G_2, respectively:

1. F(G_1) and F(G_2) mutually meet the restrictions on OWL 2 DL ontologies.
2. F(G_1) OWL 2 Direct entails F(G_1^*) with respect to *F(D)*, and F(G_1^*) OWL 2 Direct entails F(G_1) with respect to *F(D)*.
3. F(G_2) OWL 2 Direct entails F(G_2^*) with respect to *F(D)*, and F(G_2^*) OWL 2 Direct entails F(G_2) with respect to *F(D)*.
4. If F(G_1) OWL 2 Direct entails F(G_2) with respect to *F(D)*, then G_1 OWL 2 RDF-Based entails G_2 with respect to *D*.

W3C Recommendation

7.3 Proof for the Correspondence Theorem

This is the sketch of a proof for *Theorem 7.1 (OWL 2 Correspondence Theorem)* in Section 7.2. The proof sketch provides the basic line of argumentation for showing the theorem. However, for complexity reasons, some technical aspects of the theorem are only coarsely treated, and the proof sketch also refrains from considering the full amount of OWL 2 language constructs. For certain steps of the proof there are example calculations that focus only on a small fraction of language constructs, but which can be taken as a hint on how a complete proof taking into account every feature of the OWL 2 RDF-Based Semantics could be constructed in principle. A complete proof could make use of the observation that the definitions of the OWL 2 Direct Semantics and the OWL 2 RDF-Based Semantics, despite their technical differences as outlined in Section 7.1, are closely aligned with respect to the different language constructs of OWL 2.

The proof sketch will make use of an approach that will be called "*balancing*" throughout this section, and which will now be introduced. The basic idea is to substitute the original pair of RDF graphs in an OWL 2 DL entailment query by another entailment query having the same semantic characteristics under the OWL 2 Direct Semantics, but for which the technical differences between the two semantics specifications have no relevant consequences under the OWL 2 RDF-Based Semantics anymore. A concrete example for the application of this approach was given in Section 7.1.

Definition (Balanced): A pair of RDF graphs (G_1 , G_2) is called *balanced*, if and only if G_1 and G_2 are OWL 2 DL ontologies in RDF graph form, such that all the following conditions hold, with F(G_1) and F(G_2) being the OWL 2 DL ontologies in Functional Syntax form [*OWL 2 Specification*] that result from applying the reverse RDF mapping [*OWL 2 RDF Mapping*] to G_1 and G_2, respectively:

1. F(G_1) and F(G_2) mutually meet the restrictions on OWL 2 DL ontologies as specified in Section 3 of the OWL 2 Structural Specification [*OWL 2 Specification*].
2. Nodes in G_1 and G_2:
    1. for every IRI *u* occurring in G_1 or G_2 that corresponds to a non-built-in entity in F(G_1) or F(G_2), respectively, the graph contains, for every entity type *T* of *u*, a declaration statement of the form "*u* `rdf:type` *t*", where *t* is the vocabulary class IRI corresponding to *T* (see Table 7 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*] and Section 5.8 of the OWL 2 Structural Specification [*OWL 2 Specification*]);
    2. every plain or typed literal occurring in G_2 also occurs in G_1 (see Section 4 of the OWL 2 Structural Specification [*OWL 2 Specification*]).
3. G_2 contains exactly one ontology header consisting of a single RDF triple of the form "*x* `rdf:type owl:Ontology`", where *x* is either a blank node or, if an ontology IRI is used in G_1, may alternatively equal that ontology IRI (see Table 4 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]).
4. G_2 does *not* contain:
    1. the RDF encoding of an annotation (see Sections 3.2.2 and 3.2.3, and Table 17 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]);
    2. a statement with an ontology property such as "`owl:imports`";
    3. a deprecation statement based on "`owl:DeprecatedClass`", "`owl:DeprecatedProperty`" and "`owl:deprecated`" (see Table 16 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]);
    4. an annotation property axiom based on "`rdfs:subClassOf`", "`rdfs:domain`" and "`rdfs:range`" (see Table 16 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]).

W3C Recommendation

5. Any of the following sub graphs of G_2 is also a sub graph of G_1:
   1. the RDF encoding of an entity declaration (see Table 7 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]);
   2. the RDF encoding of a property expression (see Table 11 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]);
   3. the RDF encoding of a class expression (see Tables 13 and 15 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]);
   4. the RDF encoding of a data range expression (see Tables 12 and 14 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]);
   5. an RDF sequence (see Table 3 in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*]).

Balancing Lemma: An algorithm exists that terminates on every valid input and that has the following input/output behavior:

The *valid input* of the algorithm is given by all the pairs of RDF graphs (G_1^* , G_2^*), where G_1^* and G_2^* are OWL 2 DL ontologies in RDF graph form, with F(G_1^*) and F(G_2^*) being the OWL 2 DL ontologies in Functional Syntax form [*OWL 2 Specification*] that result from applying the reverse RDF mapping [*OWL 2 RDF Mapping*] to G_1^* and G_2^*, respectively. Further, F(G_1^*) and F(G_2^*) have to mutually meet the restrictions on OWL 2 DL ontologies as specified in Section 3 of the OWL 2 Structural Specification [*OWL 2 Specification*].

For a valid input, the *output* of the algorithm is a pair of RDF graphs (G_1 , G_2), where G_1 and G_2 are OWL 2 DL ontologies in RDF graph form, such that for any OWL 2 RDF-Based datatype map D according to Definition 4.1 all the following relationships hold, with F(G_1) and F(G_2) being the OWL 2 DL ontologies in Functional Syntax form that result from applying the reverse RDF mapping to G_1 and G_2, respectively, and with F(D) being the OWL 2 Direct datatype map according to Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] that corresponds to D according to the *technical note on corresponding datatype maps* in Section 7.2:

1. The pair (G_1 , G_2) is *balanced*.
2. F(G_1) OWL 2 Direct entails F(G_1^*) with respect to *F(D)*, and F(G_1^*) OWL 2 Direct entails F(G_1) with respect to *F(D)*.
3. F(G_2) OWL 2 Direct entails F(G_2^*) with respect to *F(D)*, and F(G_2^*) OWL 2 Direct entails F(G_2) with respect to *F(D)*.

Proof for the Balancing Lemma:

Let the graph pair (G_1^* , G_2^*) be a valid input. The resulting RDF graphs G_1 and G_2 are constructed as follows, starting from copies of G_1^* and G_2^*, respectively.

Since the initial versions of G_1 and G_2 are OWL 2 DL ontologies in RDF graph form, the *canonical parsing process* (CP) for computing the reverse RDF mapping, as described in Section 3 of the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*], can be applied. Based on CP, it is possible to identify within these graphs

- all entity types for every non-built-in IRI,
- all blank nodes that correspond to anonymous individuals, and
- all sub graphs that correspond to OWL 2 language constructs (ontology headers, declarations, expressions, axioms and annotations) as described in the OWL 2 Structural Specification [*OWL 2 Specification*].

Based on this observation, the following steps are performed:

1. Consistently substitute all blank nodes in G_2 such that G_1 and G_2 have no common blank nodes.
2. Apply CP to G_1 and G_2 (without changing these graphs) to identify the entity types of the IRIs, the anonymous individuals, and the sub graphs encoding OWL 2 language constructs.
3. For each sub graph *g* of G_2: remove *g* from G_2, if *g* is the RDF encoding of
   - an *annotation*, or
   - a *deprecation statement*, or
   - an *annotation property axiom*.
4. For the sub graph *g* of G_2 corresponding to the *ontology header* in F(G_2): substitute *g* in G_2 by a triple of the form "*x* `rdf:type owl:Ontology`", where *x* is a new blank node not yet used in G_2.
5. For each non-built-in IRI *u* in G_1 and G_2 and for each entity type *T* of *u* identified by CP: add to G_1 or G_2, respectively, the RDF triple "*u* `rdf:type` *t*", where *t* is the vocabulary class IRI corresponding to *T*.
6. For each plain or typed literal *L* in G_2: add to G_1 the RDF triple "*o* `rdfs:comment` *L*", where *o* is the IRI or blank node of the ontology header triple "*o* `rdf:type owl:Ontology`" in G_1.
7. For each sub graph *g* of G_2 that is the RDF encoding of an *entity declaration*: add *g* to G_1.
8. For each sub graph *g* of G_2 that is the RDF encoding of a *property expression* with root blank node *x*: add *g* to G_1 together with the RDF triple "*x* `owl:equivalentProperty` *x*".
9. For each sub graph *g* of G_2 that is the RDF encoding of a *class expression* with root blank node *x*: add *g* to G_1 together with the RDF triple "*x* `owl:equivalentClass` *x*".
10. For each sub graph *g* of G_2 that is the RDF encoding of a *data range expression* with root blank node *x*:
    - If *g* is part of a *data property restriction expression*, then nothing needs to be done, since the comprising restriction expression is covered by the treatment of class expressions, and therefore *g* occurs in G_1 as well.
    - Otherwise, add a declaration triple to G_1 for a new data property *p* that does not yet occur in G_1 and G_2. Then, the RDF encoding *r* of a *universal data property restriction expression* on property *p* is created for *g*. Let *r* have the new root blank node *y*. Add *r* to G_1 together with the RDF triple "*y* `owl:equivalentClass` *y*".
11. For each sub graph *g* of G_2 that is an RDF sequence with root blank node *x*, which does not occur in the RDF encoding of language constructs already treated by one of the earlier steps, i.e. *g* is part of the encoding of an axiom: create the RDF encoding *r* of an enumeration class expression with a new root blank node *y* having the main RDF triple "*y* `owl:oneOf` *x*". Then, add *r* to G_1 together with the RDF triple "*y* `owl:equivalentClass` *y*". Additionally, for every IRI *u* being a member of the RDF sequence, add to G_1 a typing triple "*u* `rdf:type owl:NamedIndividual`". If one of the sequence members is a blank node *z* that is the root node of some property expression or class expression *e*, then select a new IRI *w* not yet occurring in G_1, consistently replace *z* by *w* everywhere in *r*, add to G_1 the triple "*w* `owl:equivalentProperty` *z*" or "*w* `owl:equivalentClass` *z*", respectively, and add to G_1 the two triples "*w* `rdf:type owl:NamedIndividual`" and "*w* `rdf:type` *t*", where *t* is the vocabulary class IRI that represents the appropriate entity type of the expression *e*. No further treatment of *e* is needed, since *e* is treated by the earlier steps covering expressions.

In the following it is shown that all the claims of the balancing lemma hold.

A: Existence of a Terminating Algorithm. An algorithm *exists* for mapping the input graph pair (G_1^* , G_2^*) to the output graph pair (G_1 , G_2), since CP (applied in step 2) is described in the form of an algorithm in the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*], and since all other steps can obviously be performed algorithmically. The algorithm *terminates*, since CP terminates on arbitrary input graphs, and since all other steps can obviously be executed in finite time.

B: The Resulting RDF Graphs are OWL 2 DL Ontologies. The RDF graphs G_1 and G_2 are OWL 2 DL ontologies in RDF graph form that mutually meet the restrictions on OWL 2 DL ontologies, since the original RDF graphs G_1^* and G_2^* have this feature, and since each of the steps described above transforms a pair of RDF graphs with this feature again into a pair of RDF graphs with this feature, for the following reasons:

- The consistent substitution of blank nodes in step 1 does not change the structure of an OWL 2 DL ontology.
- The application of CP in step 2 does not change the graphs.
- Annotations, deprecation statements and annotation property axioms are optional information in an OWL 2 DL ontology and can therefore be omitted in step 3.
- The ontology header of an OWL 2 DL ontology does neither require the existence of an ontology IRI nor of any ontology properties, and so the substitution of the ontology header in step 4 is a valid operation.
- If an entity has some particular entity type for which there is no explicitly given entity declaration, then the entity declaration may be added, as done in step 5.
- It is allowed to add arbitrary annotations to the ontology header of an OWL 2 DL ontology, as done in step 6.
- Entity declarations may be copied from G_2 to G_1 in step 7 without conflict, since the original ontologies have been assumed to mutually meet the restrictions on OWL 2 DL ontologies regarding different entity declarations for the same IRI (e.g. that one IRI must not be the name of both an object property and a data property).
- Adding to G_1 an axiom that claims equivalence of some property expression (step 8) or class expression (step 9) with itself, where the expression already occurs in G_2, is an allowed operation, since the original ontologies are assumed to mutually meet the restrictions on OWL 2 DL ontologies concerning property and class expressions, and since no syntactic restrictions exist on this specific use of equivalence axioms.
- For the case of data ranges (step 10) it is sufficient to note that placing universal property restrictions on arbitrary (simple or complex) property expressions is allowed in OWL 2 DL. The rest of the argumentation follows the lines of the treatment of class expressions in step 9.
- For the treatment of RDF sequences in step 11: First, the enumeration class expressions being constructed from the RDF sequences are syntactically valid in OWL 2 DL, since all enumerated entries are IRIs by construction. Second, there is no restriction in OWL 2 DL disallowing axioms that claim equivalence of enumeration class expressions with themselves. Third, punning in OWL 2 DL allows a given non-built-in IRI of any entity type to be additionally declared as a named individual. Forth, there is no OWL 2 DL restriction forbidding to add an entity declaration for a new (i.e. not elsewhere used) IRI and to assert the denotation of this new IRI to be equivalent to some existing property or class expression. Hence, the resulting ontologies still mutually meet all syntactic restrictions on OWL 2 DL ontologies.

C: The Resulting Pair of RDF Graphs is Balanced. All the conditions of *balanced* pairs of RDF graphs are met by the pair (G_1 , G_2) for the following reasons:

- Condition 1: It has already been shown in paragraph *B* that G_1 and G_2 mutually meet the restrictions on OWL 2 DL ontologies.
- Conditions 2.1 and 2.2 on nodes in G_1 and G_2 are met by steps 5 and 6, respectively.
- Condition 3 on ontology headers in G_2 is satisfied by step 4, always applying an anonymous ontology header.
- Conditions 4.1, 4.3 and 4.4 on annotations, deprecation statements and annotation property axioms in G_2, respectively, are all satisfied by step 3.
- Condition 4.2 on statements with ontology properties is implicitly satisfied by step 4, since the substitution of the ontology header in G_2 removes all existing statements with ontology properties.
- Condition 5.1 on entity declarations in G_2 being reflected in G_1 is satisfied by step 7.
- Conditions 5.2, 5.3 and 5.4 on property, class and data range expressions in G_2, respectively, being reflected in G_1 are met by steps 8, 9 and 10, respectively.
- Condition 5.5 on RDF sequences in G_2 being reflected in G_1 is satisfied by step 11.

D: The Resulting Ontologies are semantically equivalent with the Original Ontologies under the OWL 2 Direct Semantics. F(G_1) is semantically equivalent with F(G_1^*), since F(G_1) differs from F(G_1^*) only by (potentially):

- additional entity declarations (steps 5, 7 and 11), which have no formal meaning under the OWL 2 Direct Semantics;
- additional annotations (step 6), which have no formal meaning;
- additional tautological axioms (steps 8, 9, 10 and 11), which do not change the formal meaning;

F(G_2) is semantically equivalent with F(G_2^*), since F(G_2) differs from F(G_2^*) only by (potentially):

- differently labeled anonymous individuals (step 1), by which the formal meaning under the OWL 2 Direct Semantics keeps unchanged, since anonymous individuals are existentially interpreted;
- missing annotations, deprecation statements and annotation property axioms (step 3), which have no formal meaning;
- a modified ontology header (step 4), which has no formal meaning;
- additional entity declarations (step 5), which have no formal meaning.

End of Proof for the Balancing Lemma.

In the following, the correspondence theorem will be proven.

Assume that the premises of the correspondence theorem are true for a given pair (G_1^* , G_2^*) of RDF graphs. This allows for applying the *balancing lemma*, which provides the existence of corresponding RDF graphs G_1 and G_2 that are OWL 2 DL ontologies in RDF graph form, and which meet the definition of *balanced* graph pairs. Let F(G_1) and F(G_2) be the corresponding OWL 2 DL ontologies in Functional Syntax form. Then, the claimed relationship 1 of the correspondence theorem follows directly from relationship 1 of the *balancing lemma* and from condition 1 of the definition of balanced graph pairs. Further, the claimed relationships 2 and 3 of the correspondence theorem follow directly from the relationships 2 and 3 of the *balancing lemma*, respectively.

The rest of this proof will treat the claimed relationship 4 of the correspondence theorem, which states that if F(G_1) OWL 2 Direct entails F(G_2) with respect to *F(D)*, then G_1 OWL 2 RDF-Based entails G_2 with respect to *D*. For this to see, an arbitrary OWL 2 RDF-Based interpretation *I* will be selected that OWL 2 RDF-Based satisfies G_1. For *I*, a closely corresponding OWL 2 Direct interpretation *J* will be constructed, and it will then be shown that *J* OWL 2 Direct satisfies F(G_1). Since it was assumed that F(G_1) OWL 2 Direct entails F(G_2), it will follow that *J* OWL 2 Direct satisfies F(G_2). Based on this result, it will then be possible to show that *I* also OWL 2 RDF-Based satisfies G_2. Since *I* was arbitrarily selected, this will mean that G_1 OWL 2 RDF-Based entails G_2.

Step 1: Selection of a Pair of Corresponding Interpretations.

Let F(G_1) OWL 2 Direct entail F(G_2) w.r.t. F(*D*), and let *I* be an OWL 2 RDF-Based interpretation of a vocabulary V^I w.r.t. *D*, such that *I* OWL 2 RDF-Based satisfies G_1.

Since the pair (G_1 , G_2) is *balanced*, there exist *entity declarations* in F(G_1) for each entity type of every non-built-in IRI occurring in G_1: For each entity declaration of the form "`Declaration(`*T*(*u*)`)`" in F(G_1), such that *T* is the entity type for some IRI *u*, a typing triple of the form "*u* `rdf:type` *t*" exists in G_1, where *t* is the vocabulary class IRI representing the part of the universe of *I* that corresponds to *T*. Since *I* OWL 2 RDF-Based satisfies G_1, all these declaration typing triples are OWL 2 RDF-Based satisfied by *I*, and thus all non-built-in IRIs in G_1 are instances of all their declared parts of the universe of *I*.

The vocabulary V^J := ($V^J{}_C$, $V^J{}_{OP}$, $V^J{}_{DP}$, $V^J{}_I$, $V^J{}_{DT}$, $V^J{}_{LT}$, $V^J{}_{FA}$) of the OWL 2 Direct interpretation *J* w.r.t. the datatype map *F(D)* is now constructed as follows.

- The set $V^J{}_C$ of classes contains all IRIs in V^I that are declared as classes in F(G_1), together with all the required class names listed in Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*].
- The set $V^J{}_{OP}$ of object properties contains all IRIs in V^I that are declared as object properties in F(G_1), together with all the required object property names listed in Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*].
- The set $V^J{}_{DP}$ of data properties contains all IRIs in V^I that are declared as data properties in F(G_1), together with all the required data property names listed in Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*].
- The set $V^J{}_I$ of individuals contains all IRIs in V^I that are declared as named individuals in F(G_1), and additionally all anonymous individuals occurring in F(G_1) and F(G_2).
- The set $V^J{}_{DT}$ of datatypes is defined according to Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] w.r.t. the datatype map F(*D*), together with all other IRIs in V^I that are declared as datatypes in F(G_1).
- The set $V^J{}_{LT}$ of literals is defined according to Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] w.r.t. the datatype map F(*D*).
- The set $V^J{}_{FA}$ of facet-literal pairs is defined according to Section 2.1 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*] w.r.t. the datatype map F(*D*).

The OWL 2 Direct interpretation $J := (\Delta_I , \Delta_D , \cdot^C , \cdot^{OP} , \cdot^{DP} , \cdot^I , \cdot^{DT} , \cdot^{LT} , \cdot^{FA})$ is now defined as follows. The object and data domains of J are identified with the universe IR and the set of data values LV of I, respectively, i.e., Δ_I := IR and Δ_D := LV. The class interpretation function $\cdot^C$, the object property interpretation function $\cdot^{OP}$, the data property interpretation function $\cdot^{DP}$, the datatype interpretation function $\cdot^{DT}$, the literal interpretation function $\cdot^{LT}$, and the facet interpretation function $\cdot^{FA}$ are defined according to Section 2.2 of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*]. Specifically, for every non-built-in IRI u occurring in F(G_1):

- If u is declared as a class, then set u^C := ICEXT(I(u)), since G_1 contains the triple "*u* `rdf:type owl:Class`", i.e., I(u) ∈ IC.
- If u is declared as an object property, then set u^{OP} := IEXT(I(u)), since G_1 contains the triple "*u* `rdf:type owl:ObjectProperty`", i.e., I(u) ∈ IP.
- If u is declared as a data property, then set u^{DP} := IEXT(I(u)), since G_1 contains the triple "*u* `rdf:type owl:DatatypeProperty`", i.e., I(u) ∈ IODP.
- If u is declared as a named individual, then set u^I := I(u), since G_1 contains the triple "*u* `rdf:type owl:NamedIndividual`", i.e., I(u) ∈ IR.
- If u is declared as a datatype, then set u^{DT} := ICEXT(I(u)), since G_1 contains the triple "*u* `rdf:type rdfs:Datatype`", i.e., I(u) ∈ IDC.

Notes:

- A *literal* occurring in G_1 is mapped by the reverse RDF mapping to the same literal in F(G_1), and the formal meaning of a well-formed literal is analog for both the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics.
- A *blank node b* occurring in G_1 that represents an *anonymous individual* is written as the same blank node b in F(G_1). Both the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics treat anonymous individuals in an analog way as *existential variables* defined locally to a given ontology, i.e. some individual x exists in the universe to which all occurrences of b in the ontology can be mapped (see Section 1.5 in the RDF Semantics [*RDF Semantics*] for the precise definition on how blank nodes are treated under the OWL 2 RDF-Based Semantics). Hence, the same mapping from b to x can be used with both I and J.
- G_1 may also contain declarations for *annotation properties*. Since annotation properties have no formal meaning under the OWL 2 Direct Semantics, the OWL 2 Direct interpretation J does not treat them.
- With the above definition it is possible for J to have a *nonseparated vocabulary* according to Section 5.9 of the OWL 2 Structural Specification [*OWL 2 Specification*]. Since G_1 is an OWL 2 DL ontology in RDF graph form, it is allowed that the same IRI u may be declared as one or more of an individual name, either a class name or a datatype name, and either an object property name or a data property name. For the OWL 2 RDF-Based interpretation I, the IRI u will always denote the same individual in the universe IR, where I(u) may additionally have a class extension or a property extension, or both. For the OWL 2 Direct interpretation J, however, u will denote as an individual name an element of Δ_I, as a class name a subset of Δ_I, as a datatype name a subset of Δ_D, as an object property name a subset of $\Delta_I \times \Delta_I$, and as a data property name a subset of $\Delta_I \times \Delta_D$.

Step 2: Satisfaction of F(G_1) by the OWL 2 Direct Interpretation.

Based on the premise that *I* OWL 2 RDF-Based satisfies G_1, it has to be shown that *J* OWL 2 Direct satisfies F(G_1). For this to hold, it will be sufficient that *J* OWL 2 Direct satisfies every axiom *A* occurring in F(G_1). Let g_A be the sub graph of G_1 that is mapped to *A* by the reverse RDF mapping. The basic idea can roughly be described as follows:

Since *I* is an OWL 2 RDF-Based interpretation, all the OWL 2 RDF-Based semantic conditions are met by *I*. Due to the close alignment between the definitions in the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics, OWL 2 RDF-Based semantic conditions exist that semantically correspond to the definition of the interpretation of the axiom *A*. In particular, the antecedent of one of these semantic conditions will become true, if the RDF-encoding of *A*, i.e. the graph g_A, is satisfied (in the case of an "if-and-only-if" semantic condition this will generally be the left-to-right direction of that condition). Now, all the RDF triples in g_A are OWL 2 RDF-Based satisfied by *I*, since *I* OWL 2 RDF-Based satisfies G_1. Hence, the antecedent of the semantic condition becomes true, and therefore its consequent becomes true as well. This will reveal a certain semantic relationship that according to *I* holds between the denotations of the IRIs, literals and anonymous individuals occurring in g_A, which, roughly speaking, expresses the meaning of the OWL 2 axiom *A*. Because of the close semantic correspondence of the OWL 2 Direct interpretation *J* to *I*, the analog semantic relationship holds according to *J* between the denotations of the IRIs, literals and anonymous individuals occurring in *A*. This semantic relationship turns out to be compatible with the formal meaning of the axiom *A* as specified by the OWL 2 Direct Semantics, i.e. *J* satisfies *A*.

This basic idea is now demonstrated in more detail for a single example axiom *A* in F(G_1), which can be taken as a hint on how a complete proof taking into account every feature of the OWL 2 RDF-Based Semantics could be constructed in principle.

Example:

Let *A* be the following OWL 2 axiom in F(G_1):

A : `SubClassOf(ex:c1 ObjectUnionOf(ex:c2 ex:c3))`

and let g_A be the corresponding sub graph in G_1 that is being mapped to *A* via the reverse RDF mapping, namely

g_A :

```
ex:c1 rdfs:subClassOf _:x .
_:x rdf:type owl:Class .
_:x owl:unionOf ( ex:c2 ex:c3 ) .
```

Since the pair (G_1 , G_2) is *balanced*, G_1 contains the typing triples

```
ex:c1 rdf:type owl:Class .
ex:c2 rdf:type owl:Class .
ex:c3 rdf:type owl:Class .
```

that correspond to class entity declarations in F(G_1) for the IRIs "`ex:c1`", "`ex:c2`", and "`ex:c3`", respectively. All these declaration typing triples are OWL 2 RDF-Based satisfied by *I*, since it has been postulated that *I* OWL 2 RDF-Based satisfies G_1. Hence, by applying the semantics of `rdf:type` (see Section 4.1 of the RDF Semantics [*RDF Semantics*]), all the IRIs denote classes, precisely:

I(`ex:c1`) ∈ IC ,
I(`ex:c2`) ∈ IC , and
I(`ex:c3`) ∈ IC .

Since *I* is an OWL 2 RDF-Based interpretation, it meets all the OWL 2 RDF-Based semantic conditions, and since *I* OWL 2 RDF-Based satisfies G_1, all the triples in g_A are OWL 2 RDF-Based satisfied. This meets the left-to-right directions of the semantic conditions for subclass axioms ("`rdfs:subClassOf`", see Section 5.8) and union class expressions ("`owl:unionOf`", see Section 5.4), which results in the following semantic relationship that holds between the extensions of the classes above according to *I*:

ICEXT(*I*(`ex:c1`)) ⊆ ICEXT(*I*(`ex:c2`)) ∪ ICEXT(*I*(`ex:c3`)) .

By applying the definition of *J*, one can conclude that the following semantic relationship holds between the denotations of the class names occurring in *A* according to *J*:

(`ex:c1`) ^{C} ⊆ (`ex:c2`) ^{C} ∪ (`ex:c3`) ^{C} .

This semantic relationship is compatible with the formal meaning of the axiom *A* under the OWL 2 Direct Semantics. Hence, *J* OWL 2 Direct satisfies *A*.

Since *J* OWL 2 Direct satisfies F(G_1), and since it has been postulated that F(G_1) OWL 2 Direct entails F(G_2), it follows that *J* OWL 2 Direct satisfies F(G_2).

Step 3: Satisfaction of **G₂** ***by the OWL 2 RDF-Based Interpretation.***

The last step will be to show that *I* OWL 2 RDF-Based satisfies G_2. For this to hold, *I* needs to OWL 2 RDF-Based satisfy every triple occurring in G_2. The basic idea can roughly be described as follows:

First: According to the "*semantic conditions for ground graphs*" in Section 1.4 of the RDF Semantics specification [*RDF Semantics*], all the IRIs and literals used in RDF triples in G_2 need to be in the vocabulary V^I of *I*. This is true for the following reason: Since the pair (G_1 , G_2) is *balanced*, all IRIs and literals occurring in G_2 do also occur in G_1. Since *I* satisfies G_1, all IRIs and literals in G_1, including those in G_2, are contained in V^I due to the semantic conditions for ground graphs.

Second: If a set of RDF triples encodes an OWL 2 language construct that is not interpreted by the OWL 2 Direct Semantics, such as annotations, then G_2 should contain such a set of RDF triples only if they are also included in G_1. The reason is that with such triples there will, in general, exist OWL 2 RDF-Based interpretations only satisfying the graph G_1 but not G_2, which will render the pair (G_1 , G_2) into a nonentailment (an exception are RDF triples that are true under every OWL 2 RDF-Based interpretation). Since the pair (G_1 , G_2) is *balanced*, G_2 will not contain the RDF encoding for any *annotations*, statements with *ontology properties*, *deprecation* statements or *annotation property axioms*. Hence, there are no corresponding RDF triples that need to be satisfied by *I*.

Third: Since G_2 is an OWL 2 DL ontology in RDF graph form, the graph is partitioned by the reverse RDF mapping [*OWL 2 RDF Mapping*] into sub graphs corresponding to either *ontology headers*, *entity declarations* or *axioms*, where axioms may further consist of different kinds of *expressions*, such as Boolean class expressions. It has to be shown that all the triples in each such sub graph are OWL 2 RDF-Based satisfied by *I*.

For ontology headers: Let *A* be the ontology header of F(G_2) and let g_A be the corresponding sub graph of G_2. Since the pair (G_1 , G_2) is *balanced*, g_A is encoded as a single RDF triple of the form "*x* `rdf:type owl:Ontology`", where *x* is either an IRI or a blank node. Since G_1 is an OWL 2 DL ontology in RDF graph form, G_1 also contains the encoding of an ontology header including a triple g_1 of the form "*y* `rdf:type owl:Ontology`", where *y* is either an IRI or a blank node. Since *I* OWL 2 RDF-Based satisfies G_1, g_1 is satisfied by *I*. If both *y* and *x* are IRIs, then, due to balancing, *x* equals *y*, and therefore g_A equals g_1, i.e. g_A is OWL 2 RDF-Based satisfied by *I*. Otherwise, balancing forces *x* to be a blank node, i.e. *x* is treated as an existential variable under the OWL 2 RDF-Based Semantics according to the *"semantic conditions for blank nodes"* [*RDF Semantics*]. From this observation, and from the premise that *I* satisfies g_1, it follows that g_A is OWL 2 RDF-Based satisfied by *I*.

For entity declarations: Let *A* be an entity declaration in F(G_2), and let g_A *be the corresponding sub graph of* G_2*.* Since the pair (G_1 , G_2) is *balanced*, *A* occurs in F(G_1), and hence g_A is a sub graph of G_1. Since *I* OWL 2 RDF-Based satisfies G_1, *I* OWL 2 RDF-Based satisfies g_A.

For axioms: Let *A* be an axiom in F(G_2), and let g_A be the corresponding sub graph of G_2. Since *I* is an OWL 2 RDF-Based interpretation, all the OWL 2 RDF-Based semantic conditions are met by *I*. Due to the close alignment between the definitions in the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics, OWL 2 RDF-Based semantic conditions exist that semantically correspond to the definition of the interpretation of the axiom *A*. In particular, the consequent of one of these semantic conditions corresponds to the RDF-encoding of *A*, i.e. the graph g_A, except for declaration typing triples, for which satisfaction has already been shown (in the case of an "if-and-only-if" semantic condition this will generally be the right-to-left direction of that condition). Hence, in order to show that g_A is OWL 2 RDF-Based satisfied by *I*, it will be sufficient to show that the antecedent of this semantic condition is true. In general, several requirements have to be met to ensure this:

Requirement 1: The denotations of all the non-built-in IRIs in g_A have to be contained in the appropriate part of the universe of *I*. This can be shown as follows. For every non-built-in IRI *u* occurring in g_A, *u* also occurs in *A*. Since the pair (G_1 , G_2) is *balanced*, there are entity declarations in F(G_2) for all the entity types of *u*, each being of the form *D* := "`Declaration(`*T*(*u*)`)`" for some entity type *T*. From the reverse RDF mapping follows that for each such declaration *D* a typing triple *d* exists in G_2, being of the form *d* := "*u* `rdf:type` *t*", where *t* is the vocabulary class IRI representing the part of the universe of *I* that corresponds to the entity type *T*. It has already been shown that, for *D* being an entity declaration in F(G_2) and *d* being the corresponding sub graph in G_2, *I* OWL 2 RDF-Based satisfies *d*. Hence, *I*(*u*) is an individual contained in the appropriate part of the universe.

Requirement 2: For every expression *E* occurring in *A*, with the RDF encoding g_E in g_A, an individual has to exist in the universe of *I* that appropriately represents the denotation of *E*. Since *I* is an OWL 2 RDF-Based interpretation, all the OWL 2 RDF-Based semantic conditions are met by *I*. Due to the close alignment between the definitions in the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics, OWL 2 RDF-Based semantic conditions exist that semantically correspond to the definition of the interpretation of the expression *E*. In particular, the antecedent of one of these semantic conditions will become true, if the RDF-encoding of *E*, i.e. the graph g_E, is satisfied (in the case of an "if-and-only-if" semantic condition this will generally be the left-to-right direction of that condition). Now, since the pair (G_1 , G_2) is *balanced*, g_E also occurs in G_1. So, since *I* OWL 2 RDF-Based satisfies G_1, g_E is OWL 2 RDF-Based satisfied by *I*. Hence, the antecedent of the semantic condition becomes true, and therefore its consequent becomes true as well. This will result in the existence of an individual with the required properties, when taking into account existential blank node semantics.

Requirement 3: A semantic relationship has to hold between the denotations of the IRIs, literals and anonymous individuals occurring in g_A with respect to *I*, which, roughly speaking, expresses the meaning of the OWL 2 axiom *A*. This is the case for the following reasons: First, the literals and anonymous individuals occurring in *A* and g_A, respectively, are interpreted in an analog way under the OWL 2 Direct Semantics and the OWL 2 RDF-Based Semantics. Second, it was assumed that the OWL 2 Direct interpretation *J* OWL 2 Direct satisfies *A*, and therefore a semantic relationship with the desired properties holds with respect to *J*. Third, *J* has been defined in close correspondence to *I*, so that for the semantic relationship expressed by *J* an analog semantic relationship holds with respect to *I*.

This basic idea is now demonstrated in more detail for a single example axiom *A* in F(G_2), which can be taken as a hint on how a complete proof taking into account every feature of the OWL 2 RDF-Based Semantics could be constructed in principle.

Example:

Let *A* be the following OWL 2 axiom in F(G_2):

A : `SubClassOf(ex:c1 ObjectUnionOf(ex:c2 ex:c3))`

and let g_A be the corresponding sub graph in G_2 that is being mapped to *A* via the reverse RDF mapping, namely

g_A :

```
ex:c1 rdfs:subClassOf _:x .
_:x rdf:type owl:Class .
_:x owl:unionOf ( ex:c2 ex:c3 ) .
```

First, since the pair (G_1 , G_2) is *balanced*, G_2 contains the typing triples

```
ex:c1 rdf:type owl:Class .
ex:c2 rdf:type owl:Class .
ex:c3 rdf:type owl:Class .
```

that correspond to class entity declarations in F(G_2) for the IRIs "`ex:c1`", "`ex:c2`", and "`ex:c3`", respectively. All these declaration typing triples are OWL 2 RDF-Based satisfied by *I*, since due to balancing the typing triples exist in G_1 as well, and since it has been postulated that *I* OWL 2 RDF-Based satisfies all triples in G_1. Hence, by applying the semantics of `rdf:type` (see Section 4.1 of the RDF Semantics [*RDF Semantics*]), all the

W3C Recommendation

IRIs denote classes, and therefore the denotations of the IRIs are included in the appropriate part of the universe of *I*, precisely:

I(ex:c1) ∈ IC ,
I(ex:c2) ∈ IC , and
I(ex:c3) ∈ IC .

Second, g_A contains the sub graph g_E, given by

g_E :

```
_:x rdf:type owl:Class .
_:x owl:unionOf ( c2 c3 ) .
```

which corresponds to the union class expression *E* in *A*, given by

E : `ObjectUnionOf(ex:c2 ex:c3)`

Since the pair (G_1 , G_2) is [*balanced*], g_E occurs as a sub graph of G_1 as well. g_E contains blank nodes and, since *I* satisfies G_1, the semantic conditions for RDF graphs with blank nodes apply (see [Section 1.5 of the RDF Semantics] [[*RDF Semantics*]]). This provides the existence of a mapping *B* from blank(g_E) to IR, where blank(g_E) is the set of all blank nodes occurring in g_E. It follows that the extended interpretation *I*+*B* OWL 2 RDF-Based satisfies all the triples in g_E. Further, since *I* is an OWL 2 RDF-Based interpretation, *I* meets all the OWL 2 RDF-Based semantic conditions. Thus, the left-to-right direction of the semantic condition for union class expressions ("`owl:unionOf`", see [Section 5.4]) applies, providing:

[*I*+*B*](_:x) ∈ IC ,
ICEXT([*I*+*B*](_:x)) = ICEXT(*I*(ex:c2)) ∪ ICEXT(*I*(ex:c3)) .

Third, since the OWL 2 Direct interpretation *J* OWL 2 Direct satisfies *A*, the following semantic relationship holds between the denotations of the class names in *A* according to *J*:

(ex:c1) ^{C} ⊆ (ex:c2) ^{C} ∪ (ex:c3) ^{C} .

By applying the definition of the OWL 2 Direct interpretation *J*, one can conclude that the following semantic relationship holds between the extensions of the classes above according to *I*:

ICEXT(*I*(ex:c1)) ⊆ ICEXT(*I*(ex:c2)) ∪ ICEXT(*I*(ex:c3)) .

Finally, combining all intermediate results gives

I(ex:c1) ∈ IC ,
[*I*+*B*](_:x) ∈ IC ,
ICEXT(*I*(ex:c1)) ⊆ ICEXT([*I*+*B*](_:x)) .

Therefore, all the premises are met to apply the right-to-left direction of the semantic condition for subclass axioms ("`rdfs:subClassOf`", see [Section 5.8]), which results in

(I(`ex:c1`) , [I+B](`_:x`)) ∈ IEXT(I(`rdfs:subClassOf`)) .

So, the remaining triple

```
ex:c1 rdfs:subClassOf _:x .
```

in g_A is OWL 2 RDF-Based satisfied by I+B, where "`_:x`" is the root blank node of the union class expression g_E. Hence, w.r.t. existential blank node semantics, I OWL 2 RDF-Based satisfies all the triples in g_A.

To conclude, for any OWL 2 RDF-Based interpretation I that OWL 2 RDF-Based satisfies G_1, I also OWL 2 RDF-Based satisfies G_2. Hence, G_1 OWL 2 RDF-Based entails G_2, and therefore relationship 4 of the correspondence theorem holds. ***Q.E.D.***

8 Appendix: Comprehension Conditions (Informative)

The correspondence theorem in Section 7.2 shows that it is possible for the OWL 2 RDF-Based Semantics to reflect all the entailments of the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*], provided that one allows for certain "harmless" syntactic transformations on the RDF graphs being considered. This makes numerous potentially desirable and useful entailments available that would otherwise be outside the scope of the OWL 2 RDF-Based Semantics, for the technical reasons discussed in Section 7.1. It seems natural to ask for similar entailments even when an entailment query does not consist of OWL 2 DL ontologies in RDF graph form. However, the correspondence theorem does not apply to such cases, and thus the OWL 2 Direct Semantics cannot be taken as a reference frame for "desirable" and "useful" entailments, or for when a graph transformation can be considered "harmless" or not.

As discussed in Section 7.1, a core obstacle for the correspondence theorem to hold was the RDF encoding of OWL 2 expressions, such as union class expressions, when they appear on the right hand side of an entailment query. Under the OWL 2 RDF-Based Semantics it is not generally ensured that an individual exists, which represents the denotation of such an expression. The "*comprehension conditions*" defined in this section are additional semantic conditions that provide the necessary individuals for *every* sequence, class and property expression. By this, the combination of the normative semantic conditions of the OWL 2 RDF-Based Semantics (Section 5) and the comprehension conditions can be regarded to "simulate" the semantic expressivity of the OWL 2 Direct Semantics on entailment queries consisting of *arbitrary* RDF graphs.

The combined semantics is, however, not primarily intended for use in actual implementations. The comprehension conditions add significantly to the complexity and expressivity of the basic semantics and, in fact, have proven to lead to formal inconsistency. But the combined semantics can still be seen as a generalized reference frame for "desirable" and "useful" entailments, and this can be used, for example, to evaluate methods that syntactically transform *unrestricted* entailment queries in order to receive additional entailments under the OWL 2 RDF-Based Semantics. Such a concrete method is, however, outside the scope of this specification.

Note: The conventions in the introduction of Section 5 ("Semantic Conditions") apply to the current section as well.

W3C Recommendation

8.1 Comprehension Conditions for Sequences

Table 8.1 lists the comprehension conditions for sequences, i.e. RDF lists. These comprehension conditions provide the existence of sequences built from any finite combination of individuals contained in the universe.

Table 8.1: Comprehension Conditions for Sequences

if	then exists z_1 , … , z_n ∈ IR
a_1 , … , a_n ∈ IR	(z_1 , a_1) ∈ IEXT(*I*(`rdf:first`)) , (z_1 , z_2) ∈ IEXT(*I*(`rdf:rest`)) , … , (z_n , a_n) ∈ IEXT(*I*(`rdf:first`)) , (z_n , *I*(`rdf:nil`)) ∈ IEXT(*I*(`rdf:rest`))

8.2 Comprehension Conditions for Boolean Connectives

Table 8.2 lists the comprehension conditions for Boolean connectives (see Section 5.4 for the corresponding semantic conditions). These comprehension conditions provide the existence of complements for any class and datatype, and of intersections and unions built from any finite set of classes contained in the universe.

Table 8.2: Comprehension Conditions for Boolean Connectives

if	then exists *z* ∈ IR
s sequence of c_1 , … , c_n ∈ IC	(*z* , *s*) ∈ IEXT(*I*(`owl:intersectionOf`))
s sequence of c_1 , … , c_n ∈ IC	(*z* , *s*) ∈ IEXT(*I*(`owl:unionOf`))
c ∈ IC	(*z* , *c*) ∈ IEXT(*I*(`owl:complementOf`))
d ∈ IDC	(*z* , *d*) ∈ IEXT(*I*(`owl:datatypeComplementOf`))

8.3 Comprehension Conditions for Enumerations

Table 8.3 lists the comprehension conditions for enumerations (see Section 5.5 for the corresponding semantic conditions). These comprehension conditions provide the existence of enumeration classes built from any finite set of individuals contained in the universe.

Table 8.3: Comprehension Conditions for Enumerations

if	then exists *z* ∈ IR
s sequence of a_1 , … , a_n ∈ IR	(*z* , *s*) ∈ IEXT(*I*(`owl:oneOf`))

8.4 Comprehension Conditions for Property Restrictions

Table 8.4 lists the comprehension conditions for property restrictions (see Section 5.6 for the corresponding semantic conditions). These comprehension conditions provide the existence of cardinality restrictions on any property and for any nonnegative integer, as well as value restrictions on any property and on any class contained in the universe.

W3C Recommendation

Note that the comprehension conditions for self restrictions constrains the right hand side of the produced `owl:hasSelf` assertions to be the Boolean value `"true"^^xsd:boolean`. This is in accordance with Table 13 in Section 3.2.4 of the OWL 2 RDF Mapping [*OWL 2 RDF Mapping*].

Implementations are *not* required to support the comprehension conditions for `owl:onProperties`, but *may* support them in order to realize *n-ary dataranges* with arity ≥ 2 (see Sections 7 and 8.4 of the OWL 2 Structural Specification [*OWL 2 Specification*] for further information).

Table 8.4: Comprehension Conditions for Property Restrictions

if	then exists z ∈ IR
c ∈ IC , p ∈ IP	(z , c) ∈ IEXT(I(`owl:someValuesFrom`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
c ∈ IC , s sequence of p_1 , … , p_n ∈ IP , n ≥ 1	(z , c) ∈ IEXT(I(`owl:someValuesFrom`)) , (z , s) ∈ IEXT(I(`owl:onProperties`))
c ∈ IC , p ∈ IP	(z , c) ∈ IEXT(I(`owl:allValuesFrom`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
c ∈ IC , s sequence of p_1 , … , p_n ∈ IP , n ≥ 1	(z , c) ∈ IEXT(I(`owl:allValuesFrom`)) , (z , s) ∈ IEXT(I(`owl:onProperties`))
a ∈ IR , p ∈ IP	(z , a) ∈ IEXT(I(`owl:hasValue`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
p ∈ IP	(z , I(`"true"^^xsd:boolean`)) ∈ IEXT(I(`owl:hasSelf`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , p ∈ IP	(z , n) ∈ IEXT(I(`owl:minCardinality`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , p ∈ IP	(z , n) ∈ IEXT(I(`owl:maxCardinality`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , p ∈ IP	(z , n) ∈ IEXT(I(`owl:cardinality`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , c ∈ IC , p ∈ IP	(z , n) ∈ IEXT(I(`owl:minQualifiedCardinality`)) , (z , c) ∈ IEXT(I(`owl:onClass`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , d ∈ IDC , p ∈ IODP	(z , n) ∈ IEXT(I(`owl:minQualifiedCardinality`)) , (z , d) ∈ IEXT(I(`owl:onDataRange`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , c ∈ IC , p ∈ IP	(z , n) ∈ IEXT(I(`owl:maxQualifiedCardinality`)) , (z , c) ∈ IEXT(I(`owl:onClass`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))

W3C Recommendation

n ∈ INNI , d ∈ IDC , p ∈ IODP	(z , n) ∈ IEXT(I(`owl:maxQualifiedCardinality`)) , (z , d) ∈ IEXT(I(`owl:onDataRange`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , c ∈ IC , p ∈ IP	(z , n) ∈ IEXT(I(`owl:qualifiedCardinality`)) , (z , c) ∈ IEXT(I(`owl:onClass`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))
n ∈ INNI , d ∈ IDC , p ∈ IODP	(z , n) ∈ IEXT(I(`owl:qualifiedCardinality`)) , (z , d) ∈ IEXT(I(`owl:onDataRange`)) , (z , p) ∈ IEXT(I(`owl:onProperty`))

8.5 Comprehension Conditions for Datatype Restrictions

Table 8.5 lists the comprehension conditions for datatype restrictions (see Section 5.7 for the corresponding semantic conditions). These comprehension conditions provide the existence of datatypes built from restricting any datatype contained in the universe by any finite set of facet-value pairs contained in the facet space (see Section 4.1) of the original datatype.

The set IFS is defined in Section 5.7.

Table 8.5: Comprehension Conditions for Datatype Restrictions

if	then exists z ∈ IR , s sequence of z_1 , … , z_n ∈ IR
d ∈ IDC , f_1 , … , f_n ∈ IODP , v_1 , … , v_n ∈ LV , (f_1 , v_1) , … , (f_n , v_n) ∈ IFS(d)	(z , d) ∈ IEXT(I(`owl:onDatatype`)) , (z , s) ∈ IEXT(I(`owl:withRestrictions`)) , (z_1 , v_1) ∈ IEXT(f_1) , … , (z_n , v_n) ∈ IEXT(f_n)

8.6 Comprehension Conditions for Inverse Properties

Table 8.6 lists the comprehension conditions for inverse property expressions. These comprehension conditions provide the existence of an inverse property for any property contained in the universe.

Inverse property expressions can be used to build axioms with anonymous inverse properties, such as in the graph

```
_:x owl:inverseOf ex:p .
_:x rdfs:subPropertyOf owl:topObjectProperty .
```

Note that, to some extent, the OWL 2 RDF-Based Semantics already covers the use of inverse property expressions by means of the semantic conditions of inverse property axioms (see Section 5.12), since these semantic conditions also apply to an existential variable on the left hand side of an inverse property axiom. Nevertheless, not all relevant cases are covered by this semantic condition. For example, one might expect the above example graph to be generally true. However, the normative semantic conditions do not permit this conclusion, since it is not ensured that for every property p there is an individual in the universe with a property extension being inverse to that of p.

Table 8.6: Comprehension Conditions for Inverse Properties

if	then exists *z* ∈ IR
p ∈ IP	(*z* , *p*) ∈ IEXT(*I*(`owl:inverseOf`))

9 Appendix: Changes from OWL 1 (Informative)

This section lists relevant differences between the OWL 2 RDF-Based Semantics and the original specification of the *OWL 1 RDF-Compatible Semantics* [*OWL 1 RDF-Compatible Semantics*]. Significant effort has been spent in keeping the design of the OWL 2 RDF-Based Semantics as close as possible to that of the OWL 1 RDF-Compatible Semantics. While this aim was achieved to a large degree, the OWL 2 RDF-Based Semantics actually deviates from its predecessor in several aspects. In most cases this is because of serious technical problems that would have arisen from a conservative semantic extension. Not listed are the new language constructs and the new datatypes of OWL 2.

The following markers are used:

- **[DEV]**: a deviation from OWL 1 that breaks backward compatibility
- **[EXT]**: a backward compatible extension to OWL 1
- **[NOM]**: a change of the nomenclature originally used in OWL 1
- **[DPR]**: a feature of OWL 1 that has been deprecated as of OWL 2

Generalized Graph Syntax [EXT]. The OWL 2 RDF-Based Semantics allows RDF graphs to contain *IRIs* [*RFC 3987*] (see Section 2.1), whereas the OWL 1 RDF-Compatible Semantics was restricted to RDF graphs with *URIs* [*RFC 2396*]. This change is in accordance with the rest of the OWL 2 specification (see Section 2.4 of the OWL 2 Structural Specification [*OWL 2 Specification*]). In addition, the OWL 2 RDF-Based Semantics is now explicitly allowed to be applied to RDF graphs containing "*generalized*" *RDF triples*, i.e. triples that can consist of IRIs, literals or blank nodes in all three positions (Section 2.1), although implementations are not required to support this. In contrast, the OWL 1 RDF-Compatible Semantics was restricted to RDF graphs conforming to the RDF Concepts specification [*RDF Concepts*]. These limitations of the OWL 1 RDF-Compatible Semantics were actually inherited from the RDF Semantics specification [*RDF Semantics*]. The relaxations are intended to warrant interoperability with existing and future technologies and tools. Both changes are compatible with OWL 1, since all RDF graphs that were legal under the OWL 1 RDF-Compatible Semantics are still legal under the OWL 2 RDF-Based Semantics.

Facets for Datatypes [EXT]. The basic definitions of a *datatype* and a *D-interpretation*, as defined by the RDF Semantics specification and as applied by the OWL 1 RDF-Compatible Semantics, have been extended to take into account *constraining facets* (see Section 4), in order to allow for *datatype restrictions* as specified in Section 5.7. This change is compatible with OWL 1, since Section 5.1 of the RDF Semantics specification explicitly allows for extending the minimal datatype definition provided there.

Correspondence Theorem and Comprehension Conditions [DEV]. The semantic conditions of the OWL 1 RDF-Compatible Semantics included a set of so called *"comprehension conditions"*, which allowed to prove the original *"correspondence theorem"* stating that every entailment of OWL 1 DL was also an entailment of OWL 1 Full. The document at hand adds comprehension conditions for the new language constructs of OWL 2 (see Section 8). However, the comprehension conditions are *not* a normative aspect of the OWL 2 RDF-Based Semantics anymore. It has turned out that combining the comprehension conditions with the normative set of semantic conditions in Section 5 would lead to formal inconsistency of the resulting semantics (Issue 119). In addition, it became clear that a correspondence theorem along the lines of the original theorem would not work for the relationship between the OWL 2 RDF-Based Semantics and the OWL 2 Direct Semantics [*OWL 2 Direct Semantics*], since it is not possible to "balance" the differences between the two semantics solely by means of additional comprehension conditions (see Section 7.1). Consequently, the correspondence theorem of the OWL 2 RDF-Based Semantics (Section 7.2) follows an alternative approach that replaces the use of the comprehension conditions and can be seen as a technical refinement of an idea originally discussed by the WebOnt Working Group (email). This change is an *incompatible deviation* from OWL 1, since certain aspects of the originally normative definition of the semantics have been removed.

Flawed Semantics of Language Constructs with Argument Lists [DEV]. In the OWL 1 RDF-Compatible Semantics, the semantic conditions for unions, intersections and enumerations of classes were defined in a flawed form, which lead to formal inconsistency of the semantics (Issue 120; see also this unofficial problem description). The affected semantic conditions have been revised; see Section 5.4 and Section 5.5. This change is an *incompatible deviation* from OWL 1, since the semantics has formally been weakened in order to eliminate a source of inconsistency.

Incomplete Semantics of `owl:AllDifferent` [EXT]. The OWL 1 RDF-Compatible Semantics missed a certain semantic condition for axioms based on the vocabulary term "`owl:AllDifferent`" (see also this unofficial problem description). The missing semantic condition has been added to the OWL 2 RDF-Based Semantics (see Section 5.10). This change is compatible with OWL 1, since the semantics has been conservatively extended.

Aligned Semantics of `owl:DataRange` and `rdfs:Datatype` [EXT]. The class `owl:DataRange` has been made an *equivalent class* to `rdfs:Datatype` (see Section 5.2). The main purpose for this change was to allow for the deprecation of the term `owl:DataRange` in favor of `rdfs:Datatype`. This change is compatible with OWL 1 according to an analysis of the relationship between the two classes in the OWL 1 RDF-Compatible Semantics (email).

Ontology Properties as Annotation Properties [EXT]. Several properties that have been ontology properties in OWL 1, such as `owl:priorVersion`, have now been specified as both ontology properties and annotation properties, in order to be in line with the rest of the OWL 2 specification (see Section 5.5 of the OWL 2 Structural Specification [*OWL 2 Specification*]). This change is compatible with OWL 1, since the semantics has been conservatively extended: all the ontology properties of OWL 1 are still ontology properties in OWL 2.

Nonempty Data Value Enumerations [DEV]. The semantic condition for enumerations of data values in Section 5.5 is now restricted to *nonempty* sets of data values. This prevents the class `owl:Nothing` from unintentionally becoming an instance of the class `rdfs:Datatype`, as analyzed in (email). This restriction of the semantics is an *incompatible deviation* from OWL 1. Note, however, that it is still possible to define a datatype as an empty enumeration of data values, as explained in Section 5.5.

Terminological Clarifications [NOM]. This document uses the term "*OWL 2 RDF-Based Semantics*" to refer to the specified semantics only. According to Section 2.1, the term "*OWL 2 Full*" refers to the language that is determined by the set of all RDF graphs (also called "*OWL 2 Full ontologies*") being interpreted using the OWL 2 RDF-Based Semantics.

OWL 1 has not been particularly clear on this distinction. Where the OWL 1 RDF-Compatible Semantics specification talked about "*OWL Full interpretations*", "*OWL Full satisfaction*", "*OWL Full consistency*" and "*OWL Full entailment*", the OWL 2 RDF-Based Semantics Specification talks in Section 4 about "*OWL 2 RDF-Based interpretations*", "*OWL 2 RDF-Based satisfaction*", "*OWL 2 RDF-Based consistency*" and "*OWL 2 RDF-Based entailment*", respectively, since these terms are primarily meant to be related to the semantics rather than the whole language.

Modified Abbreviations [NOM]. The names "R_I", "P_I", "C_I", "EXT_I", "$CEXT_I$", "S_I", "L_I" and "LV_I", which have been used in the OWL 1 RDF-Compatible Semantics specification, have been replaced by the corresponding names defined in the RDF Semantics document [*RDF Semantics*], namely "IR", "IP", "IC", "IEXT", "ICEXT", "IS", "IL" and "LV", respectively. Furthermore, all uses of the IRI mapping "IS" have been replaced by the more general interpretation mapping "*I*", following the conventions in the RDF Semantics document. These changes are intended to support the use of the OWL 2 RDF-Based Semantics document as an incremental extension of the RDF Semantics document. Names for the "parts of the universe" that were exclusively used in the OWL 1 RDF-Compatible Semantics document, such as "IX" or "IODP", have not been changed. Other abbreviations, such as "IAD" for the class extension of `owl:AllDifferent`, have in general not been reused in the document at hand, but the explicit nonabbreviated form, such as "IEXT(*I*(`owl:AllDifferent`))", is used instead.

Modified Tuple Notation Style [NOM]. Tuples are written in the form "(…)" instead of "< … >", as in the other OWL 2 documents.

Deprecated Vocabulary Terms [DPR]. The following vocabulary terms have been deprecated as of OWL 2 by the Working Group, and *should not* be used in new ontologies anymore:

- `owl:DataRange` (per resolution of Issue 29)

10 Appendix: Change Log (Informative)

10.1 Changes Since Recommendation

This section summarizes the changes to this document since the Recommendation of 27 October, 2009.

- With the publication of the XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes Recommendation of 5 April 2012, the elements of OWL 2 which are based on XSD 1.1 are now considered required, and the note detailing the optional dependency on the XSD 1.1 Candidate Recommendation of 30 April, 2009 has been removed from the "Status of this Document" section.

10.2 Changes Since Proposed Recommendation

This section summarizes the changes to this document since the Proposed Recommendation of 22 September, 2009.

- [editorial] Correction of grammar (punctuation, word order, etc.), mainly in the Introduction section.
- [editorial] Updated and corrected several hyperlinks.

W3C Recommendation

10.3 Changes Since Candidate Recommendation

This section summarizes the changes to this document since the Candidate Recommendation of 11 June, 2009.

- [resolution] Re-definition of several ontology properties to be both ontology properties and annotation properties, in order to align the RDF-Based Semantics with the rest of the OWL 2 specification, and in particular to avoid an equivocal definition of the OWL 2 RL/RDF rules (per WG resolution).
- [correction] Correction of the type of facets: Facets are intended to be data properties and have been used as such elsewhere in the document, but they were wrongly specified as unrestricted properties so far.
- [correction] Correction of a mismatch between the definition of D-interpretations in the document at hand and the RDF Semantics specification: according to the definition of simple interpretations, LV contains all plain literals in the vocabulary *V*. The missing reference to "*V*" has been added.
- [nonnormative] Correction of an error in the formulation of the correspondence theorem.
- [nonnormative] The section on Axiomatic Triples has been extended by an explicit set of axiomatic triples, based on the discussion in the rest of the section.
- [nonnormative] The section on Axiomatic Triples now explicitly mentions axiomatic triples for datatypes and facets corresponding to the semantic conditions for datatypes and facets, respectively.
- [nonnormative] Refinement of the proof for the correspondence theorem and correction of several errors. Motivated by these changes, the example in Section 7.1 has been slightly revised as well.
- [editorial] Added a description and ALT-attribute text to Figure 1 on the parts hierarchy.
- [editorial] Distinction between normative and nonnormative references, as in other OWL 2 documents.
- [editorial] Added some clarification to the introduction section.
- [editorial] Removed a redundant conclusion from the table presenting the semantic conditions for datatype restrictions, since this conclusion already follows from the semantic conditions for the vocabulary properties, and having the conclusion repeated would not match the general approach that is applied when presenting "if-then" semantic conditions in this document.
- [editorial] Reworded the description of the markers in the section on changes from OWL 1, and added a marker "DPR" for the deprecated features.
- [editorial] Changed the presentation style of references and citations to a form used in all OWL 2 documents.
- [editorial] Changed the presentation style for tuples from "⟨ … ⟩" to "(…)", to follow the conventions used in the other OWL 2 documents.
- [editorial] Numerous minor corrections and stylistic improvements.

10.4 Changes Since Last Call

This section summarizes the changes to this document since the Last Call Working Draft of 21 April, 2009.

- [resolution] Renamed the annotation vocabulary terms "`owl:subject`", "`owl:predicate`" and "`owl:object`" to "`owl:annotatedSource`", "`owl:annotatedProperty`" and "`owl:annotatedTarget`", respectively (per WG resolution).

W3C Recommendation

- [resolution] Replaced the datatype "`rdf:text`" by "`rdf:PlainLiteral`" (per WG resolution).
- [resolution] Replaced the facet "`rdf:langPattern`" by "`rdf:langRange`", following the same replacement in the original rdf:PlainLiteral specification.
- [correction] Changed the range of the property "`owl:annotatedProperty`" from IP to IR in order to avoid undesired semantic side effects from annotations. This was an oversight when the original semantic conditions for annotations of axioms and annotations were removed from the document.
- [nonnormative] The semantic conditions and comprehension conditions for the n-ary property restrictions have been changed to only cover property sequences of length greater than 0, since the meaning of an expression with an empty property set is not clear.
- [editorial] Explained the optional status of the semantic conditions concerned with the IRI "`owl:onProperties`", in accordance with the rest of the OWL 2 specification.
- [editorial] Shortened and clarified some section titles, moved the section on semantic conditions for sub property chains within Section 5, and aligned the entry order of all tables in Section 8 with those in Section 5.
- [editorial] Several clarifications, minor corrections and cosmetic changes.

11 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 Jie Bao (RPI), Ivan Herman (W3C/ERCIM), 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 Computing Laboratory), 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 Computing Laboratory), Elisa Kendall (Sandpiper Software), Markus Krötzsch (FZI), Carsten Lutz (Universität Bremen), Deborah L. McGuinness (RPI), Boris Motik (Oxford University Computing Laboratory), 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, and Vipul Kashyap.

12 References

12.1 Normative References

[OWL 2 Specification]
OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition) Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. W3C

W3C Recommendation

Recommendation, 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/. Latest version available at http://www.w3.org/TR/owl2-syntax/.

[RDF Concepts]
Resource Description Framework (RDF): Concepts and Abstract Syntax. Graham Klyne and Jeremy J. Carroll, eds. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Latest version available as http://www.w3.org/TR/rdf-concepts/.

[RDF Semantics]
RDF Semantics. Patrick Hayes, ed., W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/. Latest version available as http://www.w3.org/TR/rdf-mt/.

[RFC 2119]
RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. Network Working Group, S. Bradner. IETF, March 1997, http://www.ietf.org/rfc/rfc2119.txt

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

12.2 Nonnormative References

[OWL 2 Direct Semantics]
OWL 2 Web Ontology Language: Direct Semantics (Second Edition) Boris Motik, Peter F. Patel-Schneider, Bernardo Cuenca Grau, eds. W3C Recommendation, 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/. Latest version available at http://www.w3.org/TR/owl2-direct-semantics/.

[OWL 2 RDF Mapping]
OWL 2 Web Ontology Language: Mapping to RDF Graphs (Second Edition) Peter F. Patel-Schneider, Boris Motik, eds. W3C Recommendation, 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/. Latest version available at http://www.w3.org/TR/owl2-mapping-to-rdf/.

[OWL 1 RDF-Compatible Semantics]
OWL Web Ontology Language: Semantics and Abstract Syntax, Section 5. RDF-Compatible Model-Theoretic Semantics. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, eds., W3C Recommendation, 10 February 2004.

[RFC 2396]
RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax. T. Berners-Lee, R. Fielding, U.C. Irvine and L. Masinter. IETF, August 1998.