W3C Editor's Draft

OWL 2 Web Ontology Language New Features and Rationale

W3C Editor's Draft 14 September 2009

This version:
http://www.w3.org/2007/OWL/draft/ED-owl2-new-features-20090914/
Latest editor's draft:
http://www.w3.org/2007/OWL/draft/owl2-new-features/
Previous version:
http://www.w3.org/2007/OWL/draft/ED-owl2-new-features-20090611/ (color-coded diff)
Editors:
Christine Golbreich, University of Versailles Saint-Quentin and LIRMM
Evan K. Wallace, National Institute of Standards and Technology (NIST)
Contributors:
Peter F. Patel-Schneider, Bell Labs Research, Alcatel-Lucent

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

Abstract

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. The OWL 2 Document Overview describes the overall state of OWL 2, and should be read before other OWL 2 documents.
This document is a simple introduction to the new features of the OWL 2 Web Ontology Language, including an explanation of the differences between the initial version of OWL and OWL 2. The document also presents the requirements that have motivated the design of the main new features, and their rationale from a theoretical and implementation perspective.

W3C Editor's Draft

Status of this Document

May Be Superseded

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

Summary of Changes

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

- A note was added pointing out that a property being asymmetric is a much stronger notion than its being non-symmetric.
- A note on the origin of the profile names was added.

Please Comment By 12 October 2009

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

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

No Endorsement

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

Patents

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

W3C Editor's Draft

Table of Contents

1 Introduction

This document provides an overview of the main new features of OWL 2 and their rationale. These features were determined based on real applications and user and tool-developer experience, some of which has been documented in the OWLED Workshop Series. The inclusion of the features is supported by use cases provided to the W3C OWL Working Group, some of which are listed in the Section 7. This document also describes and motivates some of the other design decisions that were made during the development of OWL 2 or purposefully retained from OWL Web Ontology Language (OWL 1), particularly the various concrete syntaxes for OWL 2, and the relationship of OWL 2 with RDF (Section 4). OWL 2 extends OWL 1 and inherits the language features, design decisions, and use cases for OWL 1. This document thus forms an extension of the Use Cases and Requirements that underlie OWL 1 [*OWL Use Cases and Requirements*].

OWL 2 adds several new features to OWL 1, including increased expressive power for properties, extended support for datatypes, simple metamodeling capabilities, extended annotation capabilities, and keys (Section 2). OWL 2 also defines several profiles – OWL 2 language subsets that may better meet certain performance requirements or may be easier to implement (Section 3).

2 Features & Rationale

The new features of OWL 2 are presented here, organized in the following categories:

1. syntactic sugar to make some common statements easier to say,
2. new constructs that increase expressivity,
3. extended support for datatypes,
4. simple metamodeling capabilities,
5. extended annotation capabilities,
6. other innovations, and minor features.

Each feature is described in a common pattern as follows:

- a brief sentence explaining why the new feature was added,
- a feature description including a informal meaning, informal syntax, and a simple example issued from Use Cases,
- the theoretical and implementation implications of the new feature, and
- links to related use cases.

Readers may selectively show or hide the Examples and the Functional Syntax (FSS) or the RDF Syntax in the Examples by toggling the buttons below .

Hide Examples | Hide FSS in Examples | Show RDF in Examples

2.1 Syntactic sugar

OWL 2 adds syntactic sugar to make some common patterns easier to write. Since all these constructs are simply shorthands, they do not change the expressiveness, semantics, or complexity of the language. Implementations, however, may prefer to take special notice of these constructs for more efficient processing.

2.1.1 F1: DisjointUnion

While OWL 1 provides means to define a set of subclasses as a disjoint and complete covering of a superclass by using several axioms, this cannot be done concisely.

DisjointUnion defines a class as the union of other classes, all of which are pairwise disjoint. It is a shorthand for separate axioms making the classes pairwise disjoint and one setting up the union class. Normative Syntax Direct Semantics RDF-Based Semantics

DisjointUnion ({ A } C CE_1 ... CE_n) where C is a class, CE_i, $1 \leq i \leq n$ are class expressions, and { A } zero or more annotations.

Example:

- HCLS

DisjointUnion(*:BrainHemisphere :LeftHemisphere :RightHemisphere*) (*UC#2*)	A *:BrainHemisphere* is exclusively either a *:LeftHemisphere* or *:RightHemisphere* and cannot be both of them.
DisjointUnion(*:Lobe :FrontalLobe :ParietalLobe :TemporalLobe :OccipitalLobe :LimbicLobe*) (*UC#1*)	A *:Lobe* is exclusively either a *:FrontalLobe*, *:ParietalLobe*, *:TemporalLobe*, *:OccipitalLobe* or a *:LimbicLobe* and cannot be more than one of them.

- CHEMISTRY

DisjointUnion(*:AmineGroup :PrimaryAmineGroup :SecondaryAmineGroup :TertiaryAmineGroup*)(*UC#3*)	An *:AmineGroup* is exclusively either a *:PrimaryAmineGroup*, *:SecondaryAmineGroup* or a *:TertiaryAmineGroup* and cannot be both of them.

- AUTOMOTIVE

DisjointUnion(:CarDoor :FrontDoor :RearDoor :TrunkDoor) (*UC#4*)	A *:CarDoor* is exclusively either a *:FrontDoor*, a *:RearDoor* or a*:TrunkDoor* and not more than one of them.

Use Case #1 Use Case #2 Use Case #3 Use Case #4

2.1.2 F2: DisjointClasses

While OWL 1 provides means to state that two subclasses are disjoint, stating that several subclasses are pairwise disjoint cannot be done concisely.

DisjointClasses states that all classes from the set are pairwise disjoint. It is a shorthand for binary disjointness axioms between the classes. Normative Syntax Direct Semantics RDF-Based Semantics

DisjointClasses ({ A } CE_1 ... CE_n) where CE_i, $1 \leq i \leq n$ are class expressions, and { A } zero or more annotations.

Example:

- HCLS

DisjointClasses(*:UpperLobeOfLung* *:MiddleLobeOfLung* *:LowerLobeOfLung*) (*UC#2*)	*:UpperLobeOfLung* *:MiddleLobeOfLung* *:LowerLobeOfLung* are pairwise exclusive.
DisjointClasses(*:LeftLung* *:RightLung*) (*UC#2*)	Nothing can be both a *:LeftLung* and a *:RightLung*.

Note: The FMA uses a huge number of disjoint classes [*FMA* C]: 3736 of template *Left X vs Right X* (e.g., Left lung vs Right lung), 13989 classes *X of left Y vs X of right Y* (e.g., Skin of right breast vs Skin of left breast), 75 classes X of *male Y vs X of female Y* (e.g., Right side of male chest vs Right side of female chest).

Use Case #1 Use Case #2

2.1.3 F3: NegativeObjectPropertyAssertion and NegativeDataPropertyAssertion

While OWL 1 provides means to assert values of a property for an individual, it does not provide a construct for directly asserting values that an individual does not have (negative facts).

NegativeObjectPropertyAssertion (resp. **NegativeDataPropertyAssertion**) states that a given property does not hold for the given individuals (resp. literal). Normative Syntax Direct Semantics RDF-Based Semantics

NegativeObjectPropertyAssertion ({ A } OPE a_1 a_2) where OPE is an object property expression, a_1 a_2 are individuals, and {A} zero or more annotations.

NegativeDataPropertyAssertion`( { A } DPE a lt ) where DPE is a data property expression, a an individual, lt a literal, and {A} 0 or more annotations.`

Example:

- HCLS

```
NegativeObjectPropertyAssertion(
:livesIn :ThisPatient
:IleDeFrance ) (UC#9)
NegativeDataPropertyAssertion(
:hasAge :ThisPatient
5^^xsd:integer ) (UC#9)
```

:ThisPatient does not live in the *:IleDeFrance* region.

:ThisPatient is not five years old.

Use Case #9

2.2 New constructs for properties

OWL 1 was mainly focused on constructs for expressing information about classes and individuals, and exhibited some weakness regarding expressiveness for properties. OWL 2 offers new constructs for expressing additional restrictions on properties, new characteristics of properties, incompatibility of properties, property chains and keys.

2.2.1 F4: Self Restriction

OWL 1 does not allow for the definition of classes of objects that are related to themselves by a given property, for example the class of processes that regulate themselves. This "local reflexivity" is useful in many applications, particularly when global reflexivity does not hold for a property in general, but local reflexivity holds for some classes of object. The OWL 2 construct `ObjectHasSelf` allows local reflexivity to be used in class descriptions. Self restrictions are part of SROIQ [*SROIQ*], an extension of the description logic underlying OWL-DL (SHOIN) designed to provide additions requested by users, while not affecting its decidability and practicability. SROIQ is supported by several reasoners, including FaCT++, HermiT and Pellet [*TOOLS*].

A class expression defined using an **ObjectHasSelf** restriction denotes the class of all objects that are related to themselves via the given object property. Normative Syntax Direct Semantics RDF-Based Semantics

ObjectHasSelf `(OPE) where OPE is an object property expression.`

Example:

- HCLS

`SubClassOf( :AutoRegulatingProcess ObjectHasSelf( :regulate ) )`	Auto-regulating processes *regulate* themselves.
`SubClassOf( :Auto-Phosphorylating-Kinase ObjectHasSelf( :phosphorylate ))` (*UC#20*)	Auto-Phosphorylating-Kinases *phosphorylate* themselves.

Use Case #5 Use Case #3

2.2.2 F5: Property Qualified Cardinality Restrictions

While OWL 1 allows for restrictions on the number of instances of a property, e.g., for defining persons that have at least three children, it does not provide a means to restrain the *class* or *data range* of the instances to be counted (*qualified* cardinality restrictions), e.g., for specifying the class of persons that have at least three children who are girls. In OWL 2, *both* qualified and unqualified cardinality restrictions are possible. Qualified object and data cardinality restrictions are present in SROIQ and have been successfully implemented. They are already supported by various tools and reasoners (e.g.; Protégé 4, FACT++, HermiT, KAON2, PELLET and RACER) [*TOOLS*] [*OWL API*].

ObjectMinCardinality, **ObjectMaxCardinality**, and **ObjectExactCardinality** (respectively, **DataMinCardinality**, **DataMaxCardinality**, and **DataExactCardinality**) allow for the assertion of minimum, maximum or exact qualified cardinality restrictions, object (respectively, data) properties. Normative Syntax Direct Semantics RDF-Based Semantics

Object Property Cardinality Restrictions

ObjectMinCardinality `( n OPE [ CE ] )` where `n` is a non-negative integer, `OPE` an object property expression, and `[ CE ]` is zero or one class expression.

ObjectMaxCardinality `( n OPE [ CE ] )` where `n` is a non-negative integer, `OPE` an object property expression, and `[ CE ]` is zero or one class expression.

ObjectExactCardinality `( n OPE [ CE ] )` where n is a non-negative integer, OPE an object property expression, and [CE] is zero or one class expression.

Example:

- HCLS

`ObjectMinCardinality( 5 :hasDirectPart owl:Thing )`	Class of objects having at least 5 *direct part*.
`ObjectExactCardinality( 1 :hasDirectPart :FrontalLobe ) (UC#1)`	Class of objects having exactly one *direct part* of type *frontal lobe*.

In OWL 1 it is possible to express that a Brain Hemisphere has at least 5 direct parts but not that it has exactly one direct part of each specific type, *frontal, parietal, temporal, occipital, limbic lobe*, as needed in *UC#1*. In OWL 2 both statements are possible as shown in the examples above.

- CHEMISTRY

`ObjectMaxCardinality( 3 :boundTo :Hydrogen) (UC#3)`	Class of objects *bound to* at most three different *:Hydrogen*

- AUTOMOTIVE

`ObjectMaxCardinality( 5 :hasPart :Door ) (UC#4)`	Class of objects having at most 5 *:Door*
`ObjectExactCardinality( 2 :hasPart :RearDoor ) (UC#4)`	Class of objects having exactly 2 *:RearDoor*

Data Property Cardinality Restrictions

DataMinCardinality `( n DPE [ DR ] )` where n is a non-negative integer, DPE a data property expression, and [DR] is zero or one or one data range.

DataMaxCardinality `( n DPE [ DR ] )` where n is a non-negative integer, DPE a data property expression, and [DR] is zero or one or one data range.

DataExactCardinality `( n DPE [ DR ] )` where n is a non-negative integer, DPE a data property expression, and [DR] is zero or one or one data range.

Example:

- HCLS

`DataMaxCardinality( 1 :hasSSN )`	Each individual has at most one Social Security Number

Use Case #1 Use Case #2 Use Case #3, Use Case #4 Use Case #8

2.2.3 F6: Reflexive, Irreflexive, and Asymmetric Object Properties

While OWL 1 allows assertions that an object property is symmetric or transitive, it is impossible to assert that the property is reflexive, irreflexive or asymmetric.

The OWL 2 construct **ReflexiveObjectProperty** allows it to be asserted that an object property expression is globally reflexive - that is, the property holds for all individuals. Normative Syntax Direct Semantics RDF-Based_Semantics

ReflexiveObjectProperty `( { A } OPE )` where `OPE` is an object property expression and `{ A }` zero or more annotations.

Example:

- HCLS

`ReflexiveObjectProperty( :sameBloodGroup ) (UC#9)`	Everything has the same blood group as itself.
`ReflexiveObjectProperty( :part_of ) (UC#2)`	Everything is *:part_of* itself

Note: There are different interpretations of the mereological relations. For example OBO (Use Case #5) states that *:part_of* is reflexive while the mereological relation *anatomicalPartOf* between anatomical entities is asserted to be irreflexive in Use Case #1.

The OWL 2 construct **IrreflexiveObjectProperty** allows it to be asserted that an object property expression is irreflexive - that is, the property does not hold for any individual. Normative Syntax Direct Semantics RDF-Based_Semantics

IrreflexiveObjectProperty `( { A } OPE )` where `OPE` is an object property expression and `{ A }` zero or more annotations.

Example:

- HCLS

`IrreflexiveObjectProperty( :proper_part_of ) (UC#5)`	Nothing can be a proper part of itself.
`IrreflexiveObjectProperty( :boundBy ) (UC#1)`	Nothing can be bound by itself.

- EARTH AND SPACE

`IrreflexiveObjectProperty( :flowsInto )(UC#6)`	Nothing can flow into itself.

Note: The given examples correspond to the statements about mereological and topological properties *anatomicalPartOf :boundBy* in the given Use Cases, e.g.; Use Case #1. Other applications may, however, use these terms for properties with different characteristics.

The OWL 2 construct **AsymmetricObjectProperty** allows it to be asserted that an object property expression is asymmetric - that is if the property expression OPE holds between the individuals `x` and `y`, then it cannot hold between `y` and `x`. Note that asymmetric is stronger than simply not symmetric. Normative Syntax Direct Semantics RDF-Based_Semantics

AsymmetricObjectProperty `( { A } OPE ) where OPE is an object property expression and { A } zero or more annotations.`

Example:

- HCLS

`AsymmetricObjectProperty( :proper_part_of )(UC#8)`	The property *:proper_part_of* is asymmetric.

These constructs are part of SROIQ and have been implemented in SROIQ reasoners such as FaCT++, HermiT and Pellet.

Use Case #5 Use Case #6 Use Case #8

Note: Many use cases illustrate the desirability for Reflexivity, Irreflexivity, Asymmetry or Local Relexivity. The usefulness of these features was explicitly mentioned by the Health Care and Life Sciences interest group in their last call comment. The Semantic Web Deployment Working Group (SWD) also explicitly mentioned the potential usefulness of reflexivity and asymmetry e.g., for specifying application-specific specializations of SKOS semantic relations (see comment from the SWD). For example, in mereology, the *partOf* relation is defined to be transitive, reflexive, and antisymmetric. Many applications that describe complex structures, e.g., in life sciences or systems engineering, require extensive use of part-whole relations axiomatized in this way. Other relations encountered in ontology modeling also require such axiomatizations, possibly with different characteristics (e.g., [*OBO*] [*RO*]). Examples include proper part of and locative relations (typically

transitive and irreflexive), causal relations (typically transitive and irreflexive) and membership relations (typically irreflexive). Another example is the skos:broader relationship. The SKOS specification [*SKOS*] makes no statements regarding the reflexivity or irreflexivity of skos:broader to allow both interpretations: for example, it should be considered *reflexive* for a direct translation of an inferred OWL subclass hierarchy, but *irreflexive* for most thesauri or classification schemes. OWL 2 reflexivity/irreflexivity allows one of these two features to be added on demand. Self restrictions are even more fine grained, allowing skos:broader to be made only *locally reflexive* or *irreflexive* w.r.t. a given skos:Concept (via a SubClassOf axiom) .

2.2.4 F7: Disjoint Properties

While OWL 1 provides means to state the disjointness of classes, it is impossible to state that properties are disjoint.

The OWL 2 construct **DisjointObjectProperties** allows it to be asserted that several object properties are pairwise incompatible (exclusive); that is, two individuals cannot be connected by two different properties of the set. This construct is part of SROIQ and has been implemented in SROIQ reasoners. Normative Syntax Direct Semantics RDF-Based Semantics

DisjointObjectProperties ({ A } OPE_1 ... OPE_n) where OPE_i, $1 \leq i \leq n$ are object property expressions and { A } zero or more annotations.

Example:

- HCLS

`DisjointObjectProperties( :connectedTo :contiguousWith ) (UC#1)`	*:connectedTo* and *:contiguousWith* are exclusive properties.

Note: Use Case #1 defines two anatomical entities related by a third anatomical entity as *connected*, while when they are adjacent, they are said to be *contiguous*.

DisjointDataProperties allows it to be asserted that several data properties are pairwise incompatible (exclusive). Normative Syntax Direct Semantics RDF-Based Semantics

DisjointDataProperties ({ A } DPE_1 ... DPE_n) where DPE_i, 1 ≤ i ≤ n are data property expressions and { A } zero or more annotations.

Example:

DisjointDataProperties(*:startTime :endTime*)	Start time of something, e.g., surgery, must be different from its end time.

Use Case #1 Use Case #2 Use Case #3

2.2.5 F8: Property Chain Inclusion

OWL 1 does not provide a means to define properties as a composition of other properties, as uncle could be defined; hence, it is not possible to propagate a property (e.g.; *locatedIn*) along another property (*e.g.; partOf*). The OWL 2 construct `ObjectPropertyChain` in a `SubObjectPropertyOf` axiom allows a property to be defined as the composition of several properties. Such axioms are known as *complex role inclusions* in SROIQ (which also defines regularity conditions necessary for decidability), and have been implemented in SROIQ reasoners. Normative Syntax Direct Semantics RDF-Based Semantics

An axiom `SubObjectPropertyOf` (**ObjectPropertyChain**(OPE_1 ... OPE_n) OPE) states that any individual `x` connected with an individual `y` by a chain of object properties expressions OPE_1, ..., OPE_n is necessary connected with `y` by the object property `OPE`.

`SubObjectPropertyOf` ({ A } **ObjectPropertyChain** (OPE_1 ... OPE_n) OPE) where OPE_i, 1 ≤ i ≤ n are object property and { A } zero or more annotations.

Example:

- HCLS

SubPropertyOf(ObjectPropertyChain(*:locatedIn :partOf*) *:locatedIn*) (UC#7)	If x is *located in* y and y is *part of* z then x is *located in* z, for example a disease located in a part is located in the whole.

Use Case #1 Use Case #5 Use Case #7 Use Case #8

2.2.6 F9: Keys

OWL 1 does not provide a means to define keys. However, keys are clearly of vital importance to many applications in order to uniquely identify individuals of a given class by values of (a set of) key properties. The OWL 2 construct `HasKey` allows keys to be defined for a given class. While in OWL 2 key properties are not required to be functional or total properties, it is always possible to separately state that a key property is functional, if desired. Keys in OWL 2 are a form of DL Safe rule [*DL-Safe*]. They have been implemented in HermiT, KAON2 and Pellet, and can be added to other reasoners.

An **HasKey** axiom states that each *named* instance of a class is uniquely identified by a (data or object) property or a set of properties - that is, if two named instances of the class coincide on values for each of key properties, then these two individuals are the same. Normative Syntax Direct Semantics RDF-Based Semantics

HasKey ({ A } CE (OPE_1 ... OPE_m) (DPE_1 ... DPE_m)) where CE is a class expression, OPE_i , $1 \leq i \leq m$ are object property expressions DPE_j, $1 \leq j \leq n$ are data property expression and { A } zero or more annotations.

Example:

- HCLS

```
HasKey( :RegisteredPatient
:hasWaitingListN )

ClassAssertion(
:RegisteredPatient
:ThisPatient )
DataPropertyAssertion(
:hasWaitingListN :ThisPatient
"123-45-6789" )
```

Each registered patient - [on the ABM national organ waiting list] - is uniquely identified by his waiting list number (*UC#9*)

:ThisPatient is a *:RegisteredPatient*.

:ThisPatient has the the waiting list number "123-45-6789".

In this example, since *:hasWaitingListN* is a key for the class *:RegisteredPatient*, the number "123-45-6789" uniquely identifies *:ThisPatient*. The axiom HasKey(*:RegisteredPatient :hasWaitingListN*) only states that two different patients who have got a number assigned cannot have the same number on the waiting list: if the values of *:hasWaitingListN* were the same for two named instances of the class *:RegisteredPatient*, these two individuals would be equal. An HasKey axiom is similar to an `InverseFunctionalProperty` axiom, the main difference being that it is applicable only to individuals that are explicitly named. It does not state that each registered patient has at least or at most one value of

:hasWaitingListN. The inference that each patient who has a *:hasWaitingListN* belongs to the class *:RegisteredPatient* cannot be drawn.

`HasKey( :Transplantation :donorId :recipientId :ofOrgan )`	Each Transplantation is uniquely identified by a donor, a recipient, and an organ (*UC#9*)

A set of several properties is needed to identify a transplantation: indeed a donor may provide several organs to a single person, e.g., a kidney and a liver, or the same kind of organ to two recipients, e.g., a kidney, or different organs to different recipients.

Use Case #2 Use Case #7 Use Case #9

2.3 Extended datatype capabilities

2.3.1 F10: Extra Datatypes and Datatype Restrictions

OWL 1 provides support for only integers and strings as datatypes and does not support any subsets of these datatypes. For example, one could state that every person has an age, which is an integer, but could not restrict the range of that datatype to say that adults have an age greater than 18. OWL 2 provides new capabilities for datatypes, supporting a richer set of datatypes and restrictions of datatypes by facets, as in XML Schema.

OWL 2 datatypes include a) various kinds of numbers, adding support for a wider range of XML Schema Datatypes (double, float, decimal, positiveInteger, etc.) and providing its own datatypes, e.g., owl:real; b) strings with (or without) a Language Tag (using the rdf:PlainLiteral datatype); and c) boolean values, binary data, IRIs, time instants, etc.

DatatypeRestriction also makes it possible to specify restrictions on datatypes by means of constraining *facets* that constrain the range of values allowed for a given datataype, by length (for strings) e.g., minLength, maxLength, and minimum/ maximum value, e.g., minInclusive, maxInclusive. Extended datatypes are allowed in many description logics and are supported by several reasoners. Normative Syntax Direct Semantics RDF-Based Semantics

DatatypeRestriction $(\text{DT } F_1 \text{ lt}_1 \ldots F_n \text{ lt}_n)$ where DT is a unary datatype, $1 \leq i \leq n$ $\langle F_i \text{ lt}_i \rangle$ are pairs of constraining facet and literal.

Example:

- HCLS

`DatatypeRestriction(xsd:integer minInclusive 18)` *(UC#9)*	new datatype with a lower bound of 18 on the XML Schema datatype xsd:integer

This datatype is needed for example to define patients under 18 (children) who depend on a hospital's pediatric services while those over 18 (adults) depend on adult services.

Use Case #9 Use Case #11 Use Case #12 Use Case #18 Use Case #19

2.3.2 F11: N-ary Datatypes

In OWL 1 it is not possible to represent relationships between values for one object, e.g., to represent that a square is a rectangle whose length equals its width. N-ary datatype support was **not** added to OWL 2 because there were issues regarding just what support should be added. However, OWL 2 includes syntactic constructs needed for n-ary datatypes, to provide a common basis for extensions. The Data Range Extension: Linear Equations note proposes an extension to OWL 2 for defining data ranges in terms of linear (in)equations with rational coefficients.

Example:

- HCLS

`DataAllValuesFrom (` *:admissionTemperature* *:currentTemperature* `DataComparison(Arguments(x y) leq( x y ))))` *(UC#11)*	individuals whose *:admissionTemperature* is less than or equal to their *:currentTemperature*.

Use Case #10 Use Case #11

2.3.3 Datatype Definitions

OWL 1 allows a new class to be defined by a class description, but it does not offer means to explicitly define a new datatype. For ease of writing, reading, and maintaining ontologies, OWL 2 provides a new construct to define datatypes; this is particularly useful if the same datatype is used multiple times in an ontology.

DatatypeDefinition allows to explicitly name a new datatype. Normative Syntax Direct Semantics RDF-Based Semantics

DatatypeDefinition `( { A } DT DR ), where DT is a datatype, DR a data range and { A } zero or more annotations.`

Example:

- HCLS

```
DatatypeDefinition( :adultAge
DatatypeRestriction(xsd:integer
minInclusive 18)(UC#9)
```

An adult age is defined by using a lower bound of 18 with the XML Schema datatype xsd:integer

Use Case #9

2.3.4 Data Range Combinations

While OWL 1 allows a new class to be constructed by combining classes, it does not provide means to construct a new datatype by combining other ones. In OWL 2 it is possible to define new datatypes in this way.

In OWL 2, combinations of data ranges can be constructed using intersection (**DataIntersectionOf**), union (**DataUnionOf**), and complement (**DataComplementOf**) of data ranges.

DataIntersectionOf ({ A } DR_1 ... DR_n) where DR_i where $1 \leq i \leq n$, are data ranges and { A } zero or more annotations.

DataUnionOf ({ A } DR_1 ... DR_n) where DR_i where $1 \leq i \leq n$, are data ranges and { A } zero or more annotations.

DataComplementOf ({ A } DR) where DR_i where $1 \leq i \leq n$, are data ranges and { A } zero or more annotations.

Example:

```
DataComplementOf( :adultAge )
```

This data range contains all literals that are not a positive integer greater or equal to 18

Use Case #9

2.4 Simple metamodeling capabilities

2.4.1 F12: Punning

OWL 1 DL required a strict separation between the names of, e.g., classes and individuals. OWL 2 DL relaxes this separation somewhat to allow different uses of the same term, e.g., Eagle, to be used for both a class, the class of all Eagles, and an individual, the individual representing the species Eagle belonging to the (meta)class of all plant and animal species. However, OWL 2 DL still imposes certain restrictions: it requires that a name cannot be used for both a class and a datatype and that a name can only be used for one kind of property. The OWL 2 Direct Semantics treats the different uses of the same name as completely separate, as is required in DL reasoners.

Example:

- Telecom

`Declaration( Class( :Person ) ) (UC#13) (1)`	*:Person* is declared to be a class
`ClassAssertion( :Service :s1 ) (2)`	*:s1* is an individual of *:Service*.
`ObjectPropertyAssertion( :hasInput :s1 :Person )(3)`	the individual *:s1* is connected by *:hasInput* to the individual *:Person*.

The same term ':Person' denotes both a class in (1) and an individual in (3). This is possible in OWL 2 thanks to *punning* (Class ↔ Individual).

- Collaborative environment (Wiki)

`Declaration( Class( :Deprecated_Properties ) ) (UC#14)(1)`	*:Deprecated_Properties* is declared to be a *Class*
`Declaration( ObjectProperty( :is_located_in ) ) (2)`	*:is_located_in* is declared to be an *ObjectProperty*
`ClassAssertion( :Deprecated_Properties :is_located_in ) (3)`	*:is_located_in* is an individual of *:Deprecated_Properties*.

The same term 'is_located_in' denotes both a property (2) and an individual (3). This is possible in OWL 2 thanks to *punning* (Property ↔ Individual).

Use Case #14 could also be represented using an annotation *deprecated property* on the property *:is_located_in*, which might be more intuitive or better modeling.

- UML Design

`Declaration( Class( :Person ) ) Declaration( Class( :Company ) ) (UC#15) (1)`	*:Person* and *:Company* are declared to be classes.
`SubClassOf ( :PersonCompany :Association) (2)`	*:PersonCompany* denotes a subclass of an *:Association used to model an association between classes :Person and Company as a class.*
`ObjectPropertyDomain( :PersonCompany :Person )(3)`	The domain of the property *:PersonCompany* is *:Person*.
`ObjectPropertyRange( :PersonCompany :Company )(4)`	The range of the property *:PersonCompany* is *:Company*.

The same term *:PersonCompany* denotes both a class (2) and an ObjectProperty(3, 4). This is possible in OWL 2 thanks to *punning* (Class ↔ ObjectProperty).

Use Case #12 Use Case #13 Use Case #14 Use Case #15

2.5 Extended Annotations

OWL 1 allowed extralogical annotations, such as a label or a comment, to be given for each ontology entity, but did not allow annotations of axioms, e.g., giving information about who asserted an axiom or when. OWL 2 allows for annotatins on ontologies, entities, anonymous individuals, axioms, and annotations themselves.

2.5.1 F13: Annotations

Annotations on ontology entities and anonymous individuals OWL 2 provides the construct **AnnotationAssertion** for annotation of ontology entities (such as classes or properties) and anonymous individuals. These annotations carry no semantics in the OWL 2 Direct Semantics, allowing the direct use of DL reasoners.

AnnotationAssertion `( { A } AP s v ) where AP is an annotation property, s is an IRI or an anonymous individual, v is a literal, an IRI, or an anonymous individual and {A} are 0 or more annotations (of the annotation assertion)`

Example:

- HCLS

```
AnnotationAssertion
(rdfs:label CARO:0000003
"anatomical structure" )
(UC#5)
```

The IRI *CARO:0000003* of CARO ontology is annotated by the human-readable label *"anatomical structure"*, as a value of the rdfs:label annotation property

```
AnnotationAssertion
(FMA:UWDAID FMA:Heart 7088 )
(UC#2)
```

The IRI *FMA:Heart* of the FMA is annotated by the integer *7088* (its FMA Id), as a value of the annotation property *FMA:UWDAID*.

Annotations on Axioms, Annotations, Ontologies OWL 2 provides the construct **Annotation** for annotations of axioms and ontologies. It can also be used for annotations of annotations themselves. These annotations carry no semantics in the OWL 2 Direct Semantics, allowing the direct use of DL reasoners.

Annotation `( {A} AP v )` where `AP` is an annotation property, `v` is a literal, an IRI, or an anonymous individual and `{A}` are 0 or more annotations.

Example:

- HCLS

```
SubClassOf( Annotation(
rdfs:comment "Middle lobes of
lungs are necessarily right
lobes since left lungs do not
have middle lobe.")
:MiddleLobe :RightLobe )
(UC#2)
```

The comment "Middle lobes of lungs are necessarily right lobes" is an annotation of the subclass axiom which explains why *:MiddleLobe* is a subclass of *:RightLobe*.

Use Case #2 Use Case #5 Use Case #12 Use Case #19

2.5.2 Axioms about annotation properties

Annotation properties can be given domains (**AnnotationPropertyDomain**) and ranges (**AnnotationPropertyRange**) and participate in an annotation property hierarchy (**SubAnnotationPropertyOf**). These special axioms have no semantic meaning in the OWL 2 Direct Semantics, but carry the standard RDF semantics in the RDF-based Semantics (via the mapping to RDF vocabulary).

Subproperty of Annotation Property

SubAnnotationPropertyOf ({ A } AP_1 AP_2) where AP_1 and AP_2 are annotation properties, and {A} are 0 or more annotations.

Example:

- HCLS

SubAnnotationPropertyOf (*:narrow_synonym :synonym*) (*UC#5*)	The property *:narrow synonym* is a subproperty of *:synonym*. OBO ontologies, in particular the Gene Ontology, distinguish different kinds of synonyms: exact_synonym, narrow_synonym, broad_synonym.

Domain of Annotation Property

AnnotationPropertyDomain ({ A } AP U) where AP is an annotation property, U is an IRI and {A} are 0 or more annotations.

Example:

- HCLS

AnnotationPropertyDomain (*FMA:UWDAID* *FMA:AnatomicalEntity*)(UC#2)	Only *FMA: AnatomicalEntity* can have an FMA:UWDAID (that is, an FMA ID)

Range of Annotation Property

AnnotationPropertyRange ({ A } AP U) where AP is an annotation property, U is an IRI and {A} are 0 or more annotations.

Example:

- HCLS

AnnotationPropertyRange (*FMA:UWDAID* xsd:positiveInteger) (UC#2)	The ID of an *FMA: AnatomicalEntity* is a positive integer

Use Case #2 Use Case #5

2.6 Other Innovations

2.6.1 F14: Declarations

In OWL 1, an entity such as a class or an object property could be used in an ontology without any prior announcement, so there was no way of ensuring that entity names matched in different axioms. In practice, if an entity name was mistyped in an axiom, there was no way of catching the error. In OWL 2 a declaration signals that an entity is part of the vocabulary of an ontology. A declaration also associates an entity category (class, datatype, object property, data property, annotation property, or individual) with the declared entity. Declarations are not always necessary (see Syntax). Declarations do not affect the meaning of OWL 2 ontologies and thus do not have an effect on reasoning. Implementations may choose to check that every name is declared if desired.

Declaration `( A E )` where A is an annotation and E an entity.

Example:

- TOOLS

The following declarations state that the IRI *:Person* is used as a class and the IRI *:Peter* as an individual.

`Declaration( Class( :Person ) ) (UC#17)`	*:Person* is declared to be a class
`Declaration( NamedIndividual( :Peter ) )`	*:Peter* is declared to be an individual

- HCLS

`Declaration( Class( CARO:0000003 ) ) (UC#5)`	*CARO:0000003* is declared to be a class

Use Case #17 Use Case #5

2.6.2 Top and Bottom Properties

While OWL 1 had only top and bottom predefined entities for classes, the two classes owl:Thing and owl:Nothing, OWL 2 also provides top and bottom object and data properties, namely **owl:topObjectProperty**, **owl:bottomObjectProperty**, **owl:topDataProperty**, and **owl:bottomDataProperty**.

- all pairs of individuals are connected by `owl:topObjectProperty`
- no individuals are connected by `owl:bottomObjectProperty`.
- all possible individuals are connected with all literals by `owl:topDataProperty`

- no individual is connected by `owl:bottomDataProperty` to a literal.

2.6.3 IRIs

Uniform Resource Locators (URIs) were used in OWL 1 to identify classes, ontologies, and other ontology elements. URIs are strings formed using a subset of ASCII. This was quite limiting, particularly with respect to non-English language names as ASCII only includes letters from the English alphabet. To support broad international needs, OWL 2 uses Internationalized Resource Identifiers (IRIs) [*RFC3987*] for identifying ontologies and their elements.

2.6.4 Imports and Versioning

In OWL 1 ontologies can be stored as Semantic Web documents, and ontologies can import other ontologies. OWL 2 makes it clear that this importing is by the location of the ontology document.

OWL 2 also clears up the relationship between an ontology name (IRI) and its location and, in response to several requests, provides a simple versioning mechanism by means of version names (IRIs). Each OWL 2 ontology may have an ontology IRI, which is used to identify the ontology. An OWL 2 ontology may also have a version IRI, which is used to identify a particular version of the ontology.

An OWL 2 ontology is stored at its version IRI and one of the ontologies that have the ontology IRI is stored at the ontology IRI as well. If it does not matter which of the versions is desired then importing can use the ontology IRI, but if a particular version is desired then the version IRI is used.

Ontology `( [O [ V ]] { Import ( O' ) } { A } { AX } ) where [O] and [V] are zero or one ontology and version IRIs, {Import(O')} are 0 or more imports, O' is an ontology IRI, {A} are 0 or more annotations and {AX} are 0 or more axioms.`

The ontology is stored at its version IRI V. One of the versions using the ontology IRI O should also be stored at O; this is considered to be the current version of the ontology.

2.7 Minor features

Some other changes have been introduced in the OWL 2 syntax, but these are not changes in the expressive power with respect to OWL 1.

2.7.1 Anonymous Individuals

In OWL 1, anonymous individuals were introduced as individuals without identifiers.

Example:

`Individual(value( :city :Paris ) value( :region :IleDeFrance ))`	This axiom does not contain an individual name for the subject of the *:city* and *:region* triples, so the introduced individual is an anonymous individual.

In contrast, in OWL 2 anonymous individuals are identified using node IDs.

Example:

`ObjectPropertyAssertion( :city _:a1 :Paris ) (UC#9)`	This axiom introduces an explicit anonymous individual _:a1 for this unknown address which is in the city of Paris ...
`ObjectPropertyAssertion( :region _:a1 :IleDeFrance )`	and in the region of IleDeFrance

This change was mainly motivated by a requirement related to the new functional syntax. While patterns using blank nodes could be specified without node IDs because of the (nested) frame structure of Abstract syntax constructions, this cannot be done in the functional syntax. There is no change in expressive capability. Nothing changed on the RDF side, and the treatment of anonymous individuals in OWL 2 is fully backwards compatible with that in OWL 1. In the example above, the "_:a1" simply represents a blank node in the RDF graph.

Use Case #9

2.7.2 Inverse Properties

In OWL 1, all properties are atomic, but it is possible to assert that some object property is the inverse of another property. In OWL 2, property expressions such as `ObjectInverseOf( P )` can be directly used in class expressions. This makes writing ontologies easier by avoiding the need to name an inverse.

An inverse object property expression **ObjectInverseOf**(`P`) connects an individual a_1 with a_2 if and only if the object property `P` connects a_2 with a_1.

ObjectInverseOf`( P )` `where P is an object property.`

Example:

`ObjectInverseOf( :partOf )`	this expression represents the inverse property of *:partOf*

An inverse object properties axiom **InverseObjectProperties**(OPE_1 OPE_2) states that two properties are inverse.

InverseObjectProperties (OPE_1 OPE_2) where OPE_1 and OPE_2 are object property expressions.

Example:

The following is an example of an OWL 1 inverse property axiom.

`ObjectProperty( :hasPart inverse :partOf )`	*:hasPart* has an inverse property named *:partOf*.

This can be represented in OWL 2 by the following axiom stating that `:hasPart` is an inverse of `:partOf`.

`EquivalentProperties( :hasPart ObjectInverseOf( :partOf ) )`	*:partOf* is the same as the inverse property of *:hasPart*.

As such axioms are quite common, OWL 2 provides the following syntactic shortcut as well.

`InverseObjectProperties( :hasPart :partOf )`	*:hasPart* and *:partOf* are inverse properties.

3 Profiles

3.1 F15: OWL 2 EL, OWL 2 QL, OWL 2 RL

OWL 1 defined two major dialects, OWL DL and OWL Full, and one syntactic subset (OWL Lite). However, it turned out that this was not sufficient to address requirements later identified by deployments of OWL ontologies.

- Many applications, particularly in the life sciences, use very large ontologies, e.g.; the FMA, NCI Thesaurus, SNOMED CT, Gene Ontology and some OBO ontologies. Such ontologies often need to represent (rather) complex entities (e.g.; anatomical entities composed of parts connected in complex ways) or to allow the propagation of properties (e.g.; location of diseases from parts to whole); they also have a huge

number of classes, and heavy use is made of classification in order to facilitate development and maintenance. Applications are, therefore, mainly concerned with language scalability and reasoning performance problems (see, e.g., issues surrounding the FMA [*FMA*]), and are willing to trade off some expressiveness in return for computational guarantees, particularly w.r.t. classification.
- Many applications involving classical databases are concerned with interoperability of OWL with database technologies and tools. While the ontologies used in such applications are typically relatively lightweight, they are often used to query very large sets of individuals stored in standard relational databases. There is, therefore, a requirement to access such data directly via relational queries (e.g., SQL).
- Other applications are concerned with interoperability of the ontology language with rules and existing rule engines. While the ontologies used in such applications are again typically relatively lightweight, they may be used to query large datasets, and it may be useful or necessary to operate directly on data in the form of RDF triples. Typical cases include both OWL applications that are willing to trade the full expressivity of the language for efficiency, and RDF(S) applications that need some added expressivity from OWL 2.

In order to address the above requirements, OWL 2 defines three different profiles : OWL 2 EL, OWL 2 QL, and OWL 2 RL — sublanguages (syntactic subsets) of OWL 2 with useful computational properties (e.g., reasoning complexity in range of LOGSPACE to PTIME) or implementation possibilities (e.g., fragments implementable using RDBs). They are briefly described below; for a complete description, see Profiles [*OWL 2 Profiles*].

3.1.1 OWL 2 EL

OWL 2 EL captures the expressive power used by many large-scale ontologies, e.g.; SNOMED CT, and the NCI thesaurus.

OWL 2 EL places several syntactic restrictions on the language:

- Restrictions on constructs: OWL 2 EL supports existential quantification to a class expression or a data range, existential quantification to an individual (ObjectHasValue) or a literal (DataHasValue), self-restriction, enumerations involving a single individual or a single literal, intersection of classes and data ranges. Missing features include universal quantification to a class expression or a data range, cardinality restrictions (min, max and exact), disjunction (ObjectUnionOf, DisjointUnion, and DataUnionOf), class negation and many other features; a complete list of missing features is given in OWL 2 Profiles [*OWL 2 Profiles*].
- Restrictions on axioms: OWL 2 EL supports most axioms e.g., subClass, equivalentClass, class disjointness, range and domain, object property inclusion (SubObjectPropertyOf), possibly involving property chains, and data property inclusion (SubDataPropertyOf)transitive properties, keys (HasKey), ….

- It should be noted that in addition to syntactic restrictions, OWL 2 EL extends the global restrictions on axioms defined in the OWL 2 Structural Specification [*OWL 2 Specification*] with an additional condition (see 2.2.6 Global Restrictions in OWL 2 Profiles [*OWL 2 Profiles*]).

As a result of these restrictions, OWL 2 EL reasoners (e.g., CEL [*CEL*]) can exploit reasoning algorithms, including query answering algorithms, whose complexity is known to be worst-case polynomial (see Computational Properties in OWL 2 Profiles [*OWL 2 Profiles*]). The EL acronym reflects the profile's basis in the EL family of description logics [*EL++*] [*EL++ Update*], logics that provide only Existential quantification.

3.1.2 OWL 2 QL

OWL 2 QL captures the expressive power typically used in simple ontologies like thesauri, and (most of) the expressive power of ER/UML schemas.

OWL 2 QL places several syntactic restrictions on the language:

- Restrictions on constructs: features include a limited form of existential restrictions, subClass, equivalentClass, disjointness, range and domain, symmetric properties, etc. Missing features are existential quantification to a class expression or a data range, self-restriction, existential quantification to an individual or a literal, enumeration of individuals and literals, universal quantification to a class expression or a data range, cardinality restrictions (min, max and exact), disjunction (ObjectUnionOf, DisjointUnion, and DataUnionOf, property inclusions (SubObjectPropertyOf involving property chains), functional and inverse-functional properties, transitive properties, reflexive properties, irreflexive properties, asymmetric properties, keys; a complete list of missing features is given in OWL 2 Profiles [*OWL 2 Profiles*].
- Restrictions on axioms: OWL 2 QL supports the same class axioms as in the structural specification [*OWL 2 Specification*], except **DisjointUnion** which is disallowed.

These restrictions enable a tight integration with RDBMSs, and reasoners can be implemented on top of standard relational databases. This profile is, therefore, particularly well suited to applications requiring only relatively lightweight ontologies, but with very large number of individuals, and where it is useful or necessary to access the data *directly* via relational queries (e.g., SQL). Reasoning, including query answering, can be efficiently implemented using query rewriting techniques, and its complexity is known to be worst case NLogSpace (see Computational Properties in OWL 2 Profiles [*OWL 2 Profiles*]). the QL acronym reflects the fact that query answering can be implemented by rewriting queries into a standard relational Query Language.

3.1.3 OWL 2 RL

OWL 2 RL is designed to accommodate both OWL 2 applications that can trade the full expressivity of the language for efficiency, and RDF(S) applications that need some added expressivity from OWL 2. This is achieved by defining a syntactic subset of OWL 2 which is amenable to implementation using rule-based technologies.

OWL 2 RL places several syntactic restrictions on the language:

- Restrictions on constructs: most OWL 2 class expressions constructs are supported, but with their use restricted to certain syntactic positions (see Table 2 in OWL 2 Profiles [*OWL 2 Profiles*]). For example, neither existential quantification to a class nor unions of class expressions (`ObjectUnionOf`) are allowed on the right hand side of axioms.
- Restrictions on axioms: OWL 2 RL supports all axioms of OWL 2, except disjoint unions of classes, reflexive object property axioms, and negative object and data property assertions.

These restrictions allow OWL 2 RL to be implemented using rule-based technologies such as rule extended DBMSs, and results in the complexity of reasoning, including query answering, being worst-case polynomial (see Computational Properties in OWL 2 Profiles [*OWL 2 Profiles*]). Rule-based implementations can operate directly on RDF triples (e.g., Oracle's OWL Prime [*OWL Prime*]) and so can be applied to an arbitrary RDF graph, i.e., to any OWL 2 ontology. In this case, only correct answers to queries will be computed (reasoning will be *sound*), but it is not guaranteed to obtain all correct answers (it may not be *complete*). The profile was inspired by DLP [*DLP*] and pD* [*pD**], and the RL acronym reflects the fact that reasoning can be implemented using a standard Rule Language.

Use Case #2 Use Case #3 Use Case #4 Use Case #8 Use Case #16

3.2 Which profile to choose ?

Application developers may ask themselves which profile best suits their needs. The choice between the different profiles mainly depends on the expressiveness required by the application, the priority given to reasoning on classes or data, the size of datasets and importance of scalability, etc. The following suggestions may be useful:

- Users requiring a scalable profile for large but (rather) simple ontologies and good time performance for ontology (TBox/schema) reasoning may want to consider OWL 2 EL.
- Users requiring a profile that can easily interoperate with relational database systems, and where scalable reasoning on large datasets is the most important task may want to consider OWL 2 QL.

- Users requiring a profile that can easily interoperate with rules engines and rule extended DBMSs, and where scalable reasoning on large datasets is the most important task may want to consider OWL 2 RL.

Note that OWL 2 QL and OWL 2 RL are both well suited to applications where relatively lightweight ontologies are used with very large datasets. The choice of which to use may depend on the type of data to be processed: if it is useful or necessary to access the data directly via relational queries (e.g., SQL), then OWL 2 QL may be preferred; if it is useful or necessary to operate directly on data in the form of RDF triples, then OWL 2 RL may be preferred.

4 Other Design Choices and Rationale

While OWL 2 is fully backwards compatible with OWL 1, its conceptual design is slightly different, in particular regarding OWL 2 syntax.

4.1 Syntax

There are various syntaxes available to serialize and exchange OWL 2 ontologies. The primary exchange syntax for OWL 2 is the RDF/XML Syntax [*RDF/XML*], which is the only syntax that MUST be supported by implementations. As explained below, the main purpose of the Functional Syntax [*OWL 2 Specification*] is to specify the structure of the language. OWL/XML [*OWL 2 XML*] is an XML serialization motivated by the desire for better interoperability with XML based tools and languages.

Normative syntax
The only required exchange syntax for OWL 2 ontologies is RDF/XML, as clearly stated in Section 2.1 of the Conformance and Test Cases document [*OWL 2 Conformance*]:

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

Functional Syntax
The grammar of OWL 1 was defined by the Abstract Syntax (AS). The Functional Syntax (FS) plays a similar role for OWL 2: it defines the grammar of the language. But OWL 2 is specified not only in terms of a grammar but also of structure. Indeed, in addition to the Functional Syntax, OWL 2 has introduced the *structural specification* to precisely specify the conceptual structure of OWL 2 ontologies. The structural specification is defined using the Unified Modeling Language (UML). It uses a very simple form of UML diagrams that are expected to be easily understandable by readers familiar with object-oriented systems. The structural

specification provides a normative abstract model for all the syntaxes of OWL 2, normative and non-normative. It is independent of any concrete exchange syntaxes for OWL 2 ontologies. The Functional Syntax closely follows the structural specification. Clarity and readability of the syntax were important factors in the design of the Functional Syntax. The functional-style syntax has been introduced to allow for easy writing of OWL 2 axioms. Another benefit of the OWL 2 Functional Syntax is that it is closer to the syntax used in first order logic, which makes various specification issues as well as relating OWL 2 constructs to the general literature easier. It is one among several syntaxes for OWL 2 (e.g., RDF/XML, Manchester syntax).

OWL 1 provides a frame-like syntax that allows several features of a class, property or individual to be defined in a single axiom at once. This may cause problems in practice. First, it bundles many different aspects of the given entity into a single axiom. While this may be convenient when ontologies are being designed, it is not convenient for manipulating them programmatically. In fact, most implementations of OWL 1 break such axioms apart into several "atomic" axioms, each dealing with only a single feature of the entity. However, this may cause problems with round-tripping, as the structure of the ontology may be destroyed in the process. Second, this type of axiom is often misinterpreted as a declaration and unique "definition" of the given entity. In OWL 1, however, entities may be used without being the subject of any such axiom, and there may be many such axioms relating to the same entity. OWL 2 has addressed these problems in several ways. First, the frame-like notation has been dropped in favor of a more fine-grained structure of axioms: each axiom describes just one feature of the given entity. Second, OWL 2 provides explicit declarations, and an explicit definition of the notion of structural consistency. Although OWL 2 is more verbose, this is not expected to lead to problems given that most OWL ontologies are created using ontology engineering tools.

Example:

The following is an example of an OWL 1 frame-like axiom.

```
ObjectProperty(
:partOf
ObjectInverseOf(
:containedIn )
inverseFunctional
transitive

Annotation(
rdfs:comment "an
object is a part of
another object."))
```

The property *:partOf* has an inverse property named *containedIn*, is an inverse functional and transitive property, and has the human-friendly comment "Specifies that an object is a part of another object."

Example:

This can be represented in OWL 2 using the following axioms.

`Declaration( ObjectProperty( :partOf ) )`	Declaration of the object property *:partOf*
`AnnotationAssertion( rdfs:comment :partOf "partOf means that an object is a part of another object." )`	This assertion provides a comment on the property *:partOf* which is "*partOf* means that an object is a part of another object."
`InverseObjectProperties( :partOf :containedIn )`	*:partOf* and *:containedIn* are inverse properties
`InverseFunctionalObjectProperty( :partOf )`	*:partOf* is an inverse functional property
`TransitiveObjectProperty( :partOf )`	*:partOf* is a transitive property

Concerning the abstract syntax (AS) in OWL 2, if AS is used as an exchange syntax, then OWL 1 ontologies written in AS may be input to OWL 2 tools and remain valid ontologies. But it should be emphasized that this is an issue of the tool providers: the only required exchange syntax for OWL 2 ontologies being RDF/XML, it is up to the tools to decide whether they would accept ontologies serialized in AS (or in FS, for that matter).

OWL/XML Syntax

The OWL Working Group has defined an XML syntax for OWL 2 based on XML Schema [*XML Schema*], called the XML_Serialization, or OWL/XML [*OWL 2 XML*]. This syntax mirrors the structural specification of OWL 2 [*OWL 2 Specification*]. The XML syntax is motivated by the desire to support OWL users who want better interoperability with XML based tools and languages, for example WSDL, XSLT/XQuery/XPath, or schema-aware editors. This is a standard format that OWL tool vendors may optionally support to provide access to the extensive tool chain available for XML schemas. Thus OWL tool developers and users using tools from these vendors will be be able to write XPath, XSLT, XQuery and CSS to work with OWL. This was very difficult to do using the RDF/XML format which was the only XML format available for OWL 1. An additional benefit is that XML data can be exposed to RDF/OWL applications using GRDDL. The introduction of OWL/XML also provides a more comfortable avenue for the XML-savvy user to understand OWL and makes OWL more appealing to those organizations and individuals who have made considerable investment in XML tooling and training. An open source toolkit is already available for conversion between this format and the required exchange form RDF/XML. Thus OWL/XML integrates with existing OWL 1 tooling and data, while not breaking interoperability among tools.

4.2 Backward Compatibility

The overall structure of OWL 2 has not changed compared to OWL 1 — almost all the building blocks of OWL 2 were already present in OWL 1, albeit possibly under different names.

- In OWL 1, the abstract syntax (see Section 2 of the OWL 1 Semantics [*OWL 1 Semantics*]) played the role of both the structure and the functional syntax in OWL 2 [*OWL 2 Specification*]. The OWL 2 functional syntax differs in form from the OWL 1 abstract syntax, but its role within the overall structure of OWL is identical: it specifies the structure of the language. The OWL 2 functional syntax is much closer to the RDF graph representation and can capture more RDF graphs; it also has a direct correspondence to the structural specification in UML [*UML*].
- Like OWL 1, OWL 2 specifies a precise mapping from ontology structures (represented using the abstract/functional syntax) to RDF graphs. OWL 2, however, also benefits from an explicitly specified mapping from RDF graphs back to ontology structures [*OWL 2 RDF Mapping*].
- The two semantics (Direct [*OWL 2 Direct Semantics*] and RDF-Based [*OWL 2 RDF-Based Semantics*]) of OWL 2 have their direct counterparts in OWL 1, under the names Direct Model-Theoretic Semantics and RDF-Compatible Model-Theoretic Semantics respectively [*OWL 1 Semantics*].
- An XML Presentation Syntax was also available for OWL 1 [*OWL 1 XML Syntax*] (although not as a Recommendation). On the other hand, the Manchester syntax [*OWL 2 Manchester Syntax*] did not exist for OWL 1.
- OWL 1 defined one sub-language (OWL Lite), where OWL 2 defines three (EL, QL, and RL) [*OWL 2 Profiles*]. OWL Lite has not been re-specified for OWL 2, but because of backward compatibility, OWL Lite ends up as a sub-language of OWL 2.

The central role of RDF/XML as the only required exchange syntax for OWL 2 tools and the relationships between the Direct and RDF-Based semantics (i.e., the correspondence theorem) have not changed. More importantly, backwards compatibility with OWL 1 is complete, both syntactically and semantically.

- Just as in OWL 1, OWL 2 can handle all RDF graphs. The vocabulary that is given special meaning in OWL 2 includes the special vocabulary of OWL 1. However, the use of owl:DataRange, while still possible, is now deprecated — rdfs:Datatype should be used instead.
- The direct semantics for OWL 2 [*OWL 2 Direct Semantics*] is almost completely compatible with the direct semantics for OWL 1 [*OWL 1 Semantics*]. The only difference is that annotations are semantics-free in the direct semantics for OWL 2. It is highly unlikely, however, that users will notice this difference: firstly, the semantics given to annotations in the OWL 1 direct semantics was extremely weak and unlikely to lead to any significant entailments; and secondly, OWL 1 tools using the direct semantics typically treat annotations as though they are semantics-free.
- The RDF-based semantics for OWL 2 [*OWL 2 RDF-Based Semantics*] is completely compatible with the RDF-based semantics for OWL 1 [*OWL 1*

Semantics]. Some of the details of this semantics have changed, but the set of inferences is the same.

- The treatment of importing in RDF documents has changed slightly in OWL 2 if the RDF graphs are to be conformant OWL 2 DL ontology documents [*OWL 2 Conformance*]. In OWL 1, importing happened first, so the entire merged graph was considered as one unit [*OWL 1 Semantics*]. In OWL 2, the individual documents are considered separately in most cases [*OWL 2 Specification*]. This means that OWL 1 DL RDF documents that do not have a well-specified ontology header may need to be slightly modified to be conforming OWL 2 DL ontology documents.

5 Recapitulatory Table

This table provides a summary of the main new features with an example for each. It summarizes the relations between Use Cases (column 1), Features (column 2) and Examples (column 3). For each use case one specific feature, noted by name in bold, is selected. The corresponding example is given (column 3) and the reference from which it is issued appears in bold (column 4). The other features that the use case is concerned with are noted by numbers F1 to F15. (The choice of examples aims at reconciling an easy understandable illustration for each feature, a variety of domains, and real examples from papers available online).

Use Case	Feature(s)	Example

UC#1	**DisjointUnion** F2 F5 F7 F8 F11	`DisjointUnion(:Lobe :FrontalLobe :ParietalLobe :TemporalLob`
		
		`:Lobe is a disjoint union of :FrontalLobe :FrontalLobe :ParietalLob :TemporalLobe :Oc`

UC#2	**DisjointClasses** F1 F2 F5 F7 F9	`DisjointClasses( :LeftLung :RightLung )`
		
		`a :Lung cannot be :LeftLung and :RightLung`

UC#20	**Local reflexivity**	`ObjectHasSelf( :phosphorylates)`
		
		`class of all individuals that :phosphorylates themselves`

UC#4	**Qualified Cardinality** F1 F15	`ExactCardinality( 2 :hasPart :RearDoor )`
		
		`Class of objects having exactly 2 :RearDoor`

UC#5	**Asymmetric property** F6 F8 F13	`AsymmetricProperty( :proper_part_of)`
		
		`if p is a proper part of q then q cannot be a proper part o`

UC#6	**Irreflexive property**	IrreflexiveProperty(*:flowsInto*) Nothing *:flowsInto* itself.
UC#7	**Property chain** F9	SubPropertyOf(ObjectPropertyChain(*:locatedIn :partOf*) *:l* anything *:locatedIn* a part is *:locatedIn* the whole, e.g., a
UC#8	**Reflexive property** F5 F8	ReflexiveProperty(*:partOf*) [*Part Whole*] argues about *partOf* as a reflexive property e. a car".
UC#9	**Negative property** F9 F10	NegativePropertyAssertion(*:hasAge :ThisPatient* 5^^*xsd:inte* *This patient* is not five years old.
UC#10	**N-ary**	AllValuesFrom(*:testDate :enrollmentDate* x > y + 30) individuals whose *:testDate* is superior to their *:enrollmen*
UC#11	**N-ary** F10	AllValuesFrom(*:admissionTemperature :currentTemperature* x individuals whose *:admissionTemperature* is inferior to *:cur*
UC#12	**Datatype restriction** F5 F12 F13	DatatypeRestriction(xsd:integer minInclusive 18) new datatype with a lower bound of 18 on the XML Schema dat describe the class *Adult*.
UC#13	**metamodeling**	Declaration(Class(*:Person*)) *:Person* is declared to be a class ClassAssertion(*:Service :s1*) *:s1* is an instance of *:Service* PropertyAssertion(*:hasInput :s1 :Person*) *:s1* has input *:Person* *this is an example of punning for Class ↔ Individual.*
UC#14	**metamodeling**	Declaration(ObjectProperty(*:is_located_in*))

		:is_located_in is declared to be an *ObjectProperty* ClassAssertion(*:Deprecated_Properties :is_located_in*) *:is_located_in* is an individual of the class *:Deprecated_Pr* *this is an example of punning for Property ↔Individual.*
UC#15	**metamodeling**	Declaration(Class(*:Person*)) Declaration(Class(*:Compan* *:Person* and *:Company* are declared to be classes SubClassOf (*:PersonCompany :Association*)) association between classes *:Person* and *:Company* PropertyDomain(*:PersonCompany :Person*) The domain of the property *:PersonCompany* is *:Person.* PropertyRange(*:PersonCompany :Company*) The range of the property *:PersonCompany* is *:Company.* *this is an example of punning for Class ↔ ObjectProperty.*
UC#16	**Profiles**	*This Use Case motivates a profile e.g., OWL QL, where conjunctive query answe* *conventional relational database systems*
UC#17	**Declaration**	Declaration(Class(*:Person*)) *:Person* is declared to be a class.
UC#18	**Datatype** F5	DatatypeRestriction(xsd:integer minInclusive "18000"^^xsd: "19600"^^xsd:integer) *The data range for atmosphere above 18000 [feet] and below*
UC#19	**Annotation** F10	SubClassOf(*rdfs:comment* ("data generated by the LogParser ObserverLog") :LogInformation :Information) *This is an example of an annotation of axioms*

Legend:

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Disjoint Union	Disjoint Classes	Negative Property Assertion	Local reflexivity	Qualified Cardinality	Reflexive, Irreflexive, Asymmetric	Disjoint properties	Property chain inclusion	Keys	Datatype restriction	N-ary datatype

6 References

[OWL 2 Specification]
OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. W3C Editor's

Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-syntax/.

[OWL 2 Direct Semantics]
OWL 2 Web Ontology Language: Direct Semantics Boris Motik, Peter F. Patel-Schneider, Bernardo Cuenca Grau, eds. W3C Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-direct-semantics-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-direct-semantics/.

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

[OWL 2 RDF Mapping]
OWL 2 Web Ontology Language: Mapping to RDF Graphs Peter F. Patel-Schneider, Boris Motik, eds. W3C Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-mapping-to-rdf-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-mapping-to-rdf/.

[OWL 2 Profiles]
OWL 2 Web Ontology Language: Profiles Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, Carsten Lutz, eds. W3C Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-profiles-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-profiles/.

[OWL 2 Conformance]
OWL 2 Web Ontology Language: Conformance Michael Smith, Ian Horrocks, Markus Krötzsch, Birte Glimm, eds. W3C Editor's Draft, 14 September 2009, http://www.w3.org/2007/OWL/draft/ED-owl2-conformance-20090914/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-conformance/.

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

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

[OWL 1 XML Syntax]
OWL Web Ontology Language: XML Presentation Syntax. Masahiro Hori, Jérôme Euzenat and Peter F. Patel-Schneider, eds., W3C Note, 11 June 2003.

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

[RDF/XML]
RDF/XML Syntax Specification (Revised). Dave Beckett and Brian McBride, eds., W3C Recommendation 10 February 2004.

[OWL Use Cases and Requirements]
OWL Web Ontology Language: Use Cases and Requirements Jeff Heflin, ed. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-webont-req-20040210/. Latest version available at http://www.w3.org/TR/webont-req/.

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

[SHOIQ]
A Tableaux Decision Procedure for SHOIQ. Horrocks, I., and Sattler, U. In Proc. of 19th International Joint Conference on Artificial Intelligence (IJCAI 2005) (2005), Morgan Kaufmann, Los Altos.).

[Next Steps]
Next Steps to OWL. B. Cuenca Grau, I. Horrocks, B. Parsia, P. Patel-Schneider, and U. Sattler. In Proc. of OWL: Experiences and Directions, CEUR, 2006.

[Syntax Problem]
Problem with OWL Syntax. Boris Motik and I. Horrocks, OWLED 2006, 2006.

[CEL]
CEL—A Polynomial-time Reasoner for Life Science Ontologies. F. Baader, C. Lutz, and B. Suntisrivaraporn. In U. Furbach and N. Shankar, editors, Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06), volume 4130 of Lecture Notes in Artificial Intelligence, pages 287–291. Springer-Verlag, 2006.

[SNOMED EL+]
Replacing SEP-Triplets in SNOMED CT using Tractable Description Logic Operators. B. Suntisrivaraporn, F. Baader, S. Schulz, K. Spackman, AIME 2007

[EL++]
Pushing the EL Envelope. Franz Baader, Sebastian Brandt, and Carsten Lutz. In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005), 2005.

[EL++ Update]
Pushing the EL Envelope Further. Franz Baader, Sebastian Brandt, and Carsten Lutz. In Proc. of the Washington DC workshop on OWL: Experiences and Directions (OWLED08DC), 2008.

[DL-Lite]
Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati. J. of Automated Reasoning 39(3):385–429, 2007.

[DLP]
Description Logic Programs: Combining Logic Programs with Description Logic. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. in Proc. of the 12th Int. World Wide Web Conference (WWW 2003), Budapest, Hungary, 2003. pp.: 48–57

[pD*]
Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Herman J. ter Horst. J. of Web Semantics 3(2–3):79–115, 2005.

[OWLPrime]
Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. Zhe Wu Eadon, G. Das, S. Chong, E.I. Kolovski, V. Annamalai, M. Srinivasan, J. Oracle, Nashua, NH; Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, pages 1239-1248, Cancun, 2008.

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

[Datatype]
OWL Datatypes: Design and Implementation. Boris Motik, Ian Horrocks, ISWC 2008, Karlsruhe, Deutshland, 2008.

[XML Schema]
W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. Shudi Gao, C. M. Sperberg-McQueen, and Henry S. Thompson, eds. W3C Candidate Recommendation, 30 April 2009, http://www.w3.org/TR/2009/CR-xmlschema11-1-20090430/. Latest version available as http://www.w3.org/TR/xmlschema11-1/.

[DL-Safe]
Query Answering for OWL-DL with Rules. Boris Motik, Ulrike Sattler and Rudi Studer. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 3(1):41–60, 2005.

[UML]
OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. Object Management Group, OMG Available Specification, November 2007, http://www.omg.org/docs/formal/07-11-04.pdf.

7 Appendix: Use Cases

7.1 Use Cases ↔ Features

Use Case	Disjoint Union	Disjoint Classes	Negative property	Local reflexivity	Qualified Cardinality	Reflex., Irrefl., Asymm.	Disjoint properties	Property chain	Keys	Datatyp restricti
UC#1	*	*	-	-	*	-	*	*	-	-
UC#2	*	*	-	-	*	-	*	-	*	-
UC#3	*	*	-	-	*	-	-	-	-	-
UC#4	*	-	-	-	*	-	-	-	-	-
UC#5	-	-	-	*	-	*	-	*	-	-
UC#6	-	-	-	-	-	*	-	-	-	-
UC#7	-	-	-	-	-	-	-	*	*	-
UC#8	-	-	-	-	*	*	-	*	-	-
UC#9	-	-	*	-	-	-	-	-	*	*

UC#10	-	-	-	-	-	-	-	-	-	-
UC#11	-	-	-	-	-	-	-	-	-	*
UC#12	-	-	-	-	*	-	-	-	-	*
UC#13	-	-	-	-	-	-	-	-	-	-
UC#14	-	-	-	-	-	-	-	-	-	-
UC#15	-	-	-	-	-	-	-	-	-	-
UC#16	-	-	-	-	-	-	-	-	-	-
UC#17	-	-	-	-	-	-	-	-	-	-
UC#18	-	-	-	-	*	-	-	-	-	*
UC#19	-	-	-	-	-	-	-	-	-	*

The following list of Use Cases is not exhaustive. Use Cases included in that list are only some among many that motivated the OWL 2 *new* features - whatever user/implementor/theoretical reasons - that appear, at this time, accepted by the Working Group for OWL 2. Some other extensions pointed out in the papers (such as rules, default, etc.), possibly needed in the future, are indicated within brackets.

All use cases are presented using the following pattern: *Overview, Features, Example for, References*. The *Overview* only gives a general description of the use cases. *Features* lists several features required by the use case after the paper. *Example* points to a feature and short example which has been selected to illustrate a specific new feature of OWL 2. This same information can be seen in an abbreviated form in Table 3.2. For an easy access, *References* points to the related papers available online which URL is provided in the bibliography of the Appendix.

7.2 Use Case #1 - Brain image annotation for neurosurgery [HCLS]

Overview: The system being developed concerns the preparation of surgical procedures in neurosurgery. Specifically, the aim is to assist a user in labelling the cortical gyri and sulci in the region surrounding a lesion whose resection is the primary objective. Providing anatomical landmarks, especially in eloquent cortex, is highly important for surgery. Brain image annotation is also useful for documentation of clinical cases, which then enables retrieval of similar cases for decision support in future procedures. A shared ontology of brain anatomy is also needed to integrate multiple distributed image sources indexed by anatomical features. This is useful for large-scale federated systems for statistical analysis of brain images of major brain pathologies.

Features: **Disjoint Union, Disjoint Classes, Qualified Cardinality Restrictions, Disjoint Properties, Property chain inclusion axioms, [N-ary], [Rules]**

Example for: **Disjoint Union**

- E.g.; Lobe is a disjoint union
  of :FrontalLobe :ParietalLob :TemporalLobe :OccipitalLobe
  and :LimbicLobe.

References: [*MEDICAL REQ*] [*Ontology with rules*] [*Brain Imaging*]

7.3 Use Case #2 – The Foundational Model of Anatomy [HCLS]

Overview: The Foundational Model of Anatomy (FMA) is the most comprehensive ontology of human 'canonical' anatomy. Anatomy plays a prominent role in biomedicine, and many biomedical ontologies and applications refer to anatomical entities. FMA is a tremendous resource in bioinformatics that facilitates sharing of information among applications that use anatomy knowledge. As its authors claim, the FMA is " ... a reference ontology in biomedical informatics for correlating different views of anatomy, aligning existing and emerging ontologies in bioinformatics ...". The Anatomy ontology, together with the Gene, and Disease reference ontologies constitute the backbone of the future Semantic Web for Life Sciences. But the FMA would benefit from new features of OWL to state that some properties are exclusive (e.g.; proper-part and boundBy). Since many biomedical ontologies and applications refer to the FMA anatomical entities through cross-references, keys would also be useful.

Features: **Disjoint Union, Disjoint Classes, Qualified Cardinality Restrictions, Disjoint Properties, Keys, Extended annotations, Profiles**

Example for: **Disjoint Classes**

- E.g.; Nothing can be both a *:LeftLung* and a *:RightLung*.

References: [*FMA*]

7.4 Use Case #3 - Classification of chemical compounds [HCLS]

Overview: Functional groups describe the semantics of chemical reactivity in terms of atoms and their connectivity, which exhibit characteristic chemical behavior when present in a compound. In this use case the authors take a first step towards designing an OWL-DL ontology of functional groups for the classification of chemical compounds, and highlight the capabilities and limitations of OWL 1 and the proposed OWL 1.1 in terms of domain requirements. They also describe the application of expressive features in the design of an ontology of basic relations and how an upper-level ontology can be used to guide the formulation of life science knowledge. They report on experiences to enhance existing ontologies so as to facilitate knowledge representation and question answering.

"Monocyclic and polycyclic ring structures are important parts of molecules that participate in several kinds of chemical reactions." A new OWL language feature such as qualified cardinality restriction would be helpful to describe the number and types of functional groups.

Features: **Disjoint Union, Disjoint Classes, Qualified Cardinality Restrictions, Profiles**

Example for: **Qualified Cardinality Restrictions**

- E.g.; for specifying the number and types of functional groups.

References: [*Chemistry*]

7.5 Use Case #4 - Querying multiple sources in an automotive company [Automotive]

Overview: Large companies often store information and knowledge in multiple information systems using various models and formats. The key objective in this use case is the retrieval of relevant information from multiple data and knowledge sources for a large automotive company. For this application a language with a profile facilitating querying multiple databases and easy representation of Parts Library ISO 13584 Standard (PLIB) ontologies of Products, which is particularly used for e-business catalogues, would be helpful.

Features: **Disjoint Union, Qualified Cardinality Restrictions, Profiles (OWL 2 QL)**

Example for: **Qualified Cardinality Restrictions**

- E.g.; the class of automobile having exactly 2 rear doors.

References: [*Auto*]

7.6 Use Case #5 - OBO ontologies for biomedical data integration [HCLS]

Overview: The Open Biomedical Ontologies (OBO) consortium is pursuing a strategy to facilitate the integration of biomedical data through their annotation using common controlled ontologies. Existing OBO ontologies, including the Gene Ontology, are undergoing coordinated reform, and new ontologies are being created on the basis of an evolving set of shared principles governing ontology development. The result is an expanding family of OBO ontologies designed to be interoperable and to incorporate accurate representations of biological reality. Within that effort the OBO ontology of relations is designed to define a set of basic relations with their semantics. OBO qualifies each relation using characteristics of being transitive, symmetric, reflexive, anti-symmetric. More generally OBO format offers constructs such as is_reflexive, is_symmetric, is_cyclic, is_anti_symmetric, etc. that are used in the OBO obtologies. Converting OBO ontologies requires the new OWL 2 property axioms reflexive, irreflexive, asymmetric to map corresponding OBO constructs, otherwise they should be transformed into annotations.

Features: **Local reflexivity, Reflexive, Irreflexive, Asymmetric, Property chain inclusion axioms, Declaration [Antisymmetric]**

Example for: **Asymmetric**

- E.g.; if p is a proper part of q then q cannot be a proper part of p.

References: [*OBO*] [*RO*] [*OBO2OWL*]

7.7 Use Case #6 – Spatial and topological relationships at the Ordnance Survey [Earth and Space]

Overview: Ordnance Survey is Britain's National Mapping Agency. It currently maintains a continuously updated database of the topography of Great Britain. The database includes around 440 million man-made and natural landscape features. These features include everything from forests, roads and rivers down to individual houses, garden plots, and even pillar boxes. In addition to this topographic mapping, entire new layers of information are progressively being added to the database, such as aerial photographic images which precisely match the mapping; data providing the addresses of all properties; and integrated transport information. For topological and spatial relationships, and in many other places, “we need to be able to say whether a property is reflexive, irreflexive, asymmetric or antisymmetric in order to capture the true intentions of our axioms”.

Features: **Reflexive, Irreflexive, Asymmetric, [Antisymmetric]**

Example for: **Irreflexive**

- E.g.; Nothing flows into itself.

References: [*Ordnance*]

7.8 Use Case #7 - The Systematized Nomenclature of Medicine [HCLS]

Overview: The Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT) is a work of clinical terminology with broad coverage of the domain of health care, and it has been selected as a national standard for use in electronic health applications in many countries, including the U.S., U.K., Canada, Australia, Denmark, and others. SNOMED was originally published in 1976, while SNOMED CT became available in 2002 as a major expansion resulting from the merger of SNOMED RT with the U.K.'s Clinical Terms version 3. A major distinguishing feature differentiating it from prior editions is the use of description logic (DL) to define and organize codes and terms. Another major distinguishing feature of SNOMED is its size and complexity. With over 350,000 concept codes, each representing a different class, it is an order of magnitude larger than the next largest DL-based ontology of which we are aware.

Without property chain inclusion axioms, adoption of OWL by the SNOMED community would have required awkward workarounds with their attendant

complications and complexities - effectively killing movement in that direction. With [them], we have a clear path to using OWL 2 for further development and integration with other biomedical ontologies. The required property chain inclusion axioms allow to encode inheritance of properties along another property, e.g., *part-of*, which is of utmost importance in anatomy. For example, with axioms such as `has-location ∘ proper-part-of < has-location` injury to finger can be inferred as injury to hand. As reported in [*SNOMED EL+*] by re-engineering SNOMED-CT in this way, the number of anatomical classes dropped from 54,380 to 18,125, and the time needed by the CEL reasoner [*CEL*] (version 0.94) from 900.15 seconds to 18.99 seconds.
Like the FMA, given the common use of cross-references between SNOMED and other biomedical ontologies via concepts ID, keys would be highly useful as well.

Features: **Property chain inclusion axioms, Keys, Profiles (OWL 2 EL)**

Example for: **Property chain**

- E.g.; anything located in a part is located in the whole

References: [*SNOMED REQ*]

7.9 Use Case #8 - Simple part-whole relations in OWL Ontologies [HCLS]

Overview: Representing part-whole relations is a very common issue for those developing ontologies for the Semantic Web. OWL does not provide any built-in primitives for part-whole relations (as it does for the subclass relation), but contains sufficient expressive power to capture most, but not all, of the common cases. The study of part-whole relations is an entire field in itself - "mereology" - this note is intended only to deal with straightforward cases for defining classes involving part-whole relations. Several extensions of whole needed for part-whole are discussed in this study, namely, needs of qualified cardinality restriction, reflexivity, propagation from parts to whole

Features: **Qualified cardinality restriction, Reflexivity, Property chain inclusion**

Example for: **Reflexive**

- E.g.; a frontal lobe is *part of* a brain hemisphere or a car is *part of* a car

Note: According to the definition given in OBO, the whole is being considered as a part [*Part Whole*] but there are controversial opinions asserting that 'part of' is not reflexive.

References: [*Part Whole*]

7.10 Use Case #9 - Kidney Allocation Policy in France [HCLS]

Overview: Allocation in France falls under the responsibility of the Agence de la biomedicine. It includes general rules such as: donor-recipient ABO blood group identity, unique registration on the national waiting list (a registration number is assigned at the registration of the waiting list which uniquely identifies a patient on the waiting list) and definition of some organ specific nation-wide allocation priorities. For each kidney recipient, minimal HLA matching and forbidden antigens can be specified. Pediatric recipients get a priority for pediatric donors. Kidneys are proposed by order of priority to (1) urgent patients, (2) patients with panel reactive antibodies level = 80% included in a specific acceptable antigen protocol or =1 HLA mismatch with the donor, then (3) zero mismatch patients, and (4) patients with low transplantation accessibility. Geographic criteria are involved: each region (of the transplant map), e.g., Ile de France, is supposed to take in charge only patients living in the region. This real-life application and allocation system show how distinguishing between adults and children has strong implications in health care: at hospital, patients under 18 (child) depend on pediatric services while over 18 (adult) depend on adult services; only children less than 16 years waiting for a transplant have a priority on the waiting list.

Features: **Negative Property Assertion, Datatypes restriction, Keys**

Example for: **NegativePropertyAssertion**

- E.g.; This patient is not 5 years old.

References: [*Agence Biomedecine*] [*Transplant Ontology*]

7.11 Use Case #10 – Eligibility Criteria for Patient Recruitment

Overview: This use case is based on an ongoing W3C task force on Clinical Observations Interoperability where the goal is to enable re-use and sharing of clinical data created in healthcare delivery in the Clinical Trials context. In particular the first application chosen to demonstrate feasibility of the interoperability approach is that of patient recruitment. In this case, a sample set of clinical trial protocols available from http://www.clinicaltrials.gov each of which contains a list of eligibility (inclusion and exclusion criteria). These eligibility criteria are used for identify eligible patients and potentially form conditions in a SPARQL query or could be represented as OWL classes. They also need to be mapped as per the discussion in the use case above. A list of requirements based on an analysis of these clinical trial protocols is available from http://esw.w3.org/topic/HCLS/ClinicalObservationsInteroperability?action=AttachFile&do=get&target=FunctionalRequirements_v1.xls

In particular, one of the clinical trials requires that the enrolment date of a clinical trial participant be within 30 days after the patient has been started on a particular therapy. This motivated the need for N-ary datatypes with inequality expressions.

Features: **[N-Ary]**

Example for: **N-ary Datatypes**

- E.g.; the enrolment date of a clinical trial participant should be within 30 days after the patient has been started on a particular therapy

7.12 Use Case #11 – Multiple UCs on datatype [HCLS]

Overview: [*N-ary*] presents many Use cases that would benefit from various datatype extensions

Features: **Datatypes restriction, [N-Ary]**

Example for: **N-ary Datatypes**

- E.g.; datatypes restrictions like intervals, or N-Ary datatype with inequality such as needed in Use Case #10.

References: [*N-ary*]

7.13 Use Case #12 – Protégé report on the experiences of OWL users [Tool]

Overview: [*Protege*] reported in 2005 on Protégé experiences with the development of OWL support, and on the experiences of the user community with OWL at that time. While the overall feedback from the community was positive, their experience suggested that there were considerable gaps between the user requirements, the expressivity of OWL, and users' understanding of OWL. To summarize, based on their experiences, Protégé developers suggested a number of extensions to a future version of OWL namely, Integration of user-defined datatypes (esp. for numeric ranges), Qualified Cardinality Restrictions, Management of disjointness (owl:AllDisjoint), More flexible annotation properties (at least as best practices). This report underlined that one of the omissions in the OWL language that users complain about most often is poor representation of numeric expressions. Almost all groups, except for those developing traditional medical terminologies, sorely need to be able to express quantitative information. Typical examples include the length between 1mm and 2mm, age greater than 18 years, pressure in the range of 1030mb to 1035mb. Such range declarations are needed to classify individuals and to build class definitions such as *Adult*, and should therefore be supported by reasoners. User base points out that the current OWL datatype formalism is much too weak to support most real world applications and that many potential users therefore cannot adopt OWL. *"The user communities anxiously await an extension to the OWL specification to represent user-defined datatypes with XML Schema facets such as xsd:minInclusive."* It also points out some limitations related to annotations or metamodeling from an implementors perspective: *"Despite the value of annotation properties, in OWL DL, properties that are declared as annotation properties are greatly limited in so far that they can neither have range or domain constraints, nor can they be arranged in sub-property hierarchies. This type of*

information about a property enables tools to control the values that annotation properties can acquire. Without range constraints it is difficult to provide the user with appropriate input widgets. In a similar sense, it is often helpful to declare meta-classes so that classes can be categorized into types and different interfaces be pro-vided for each type. Currently, using these features means that the ontology will be forced into OWL Full."

Features: **Qualified cardinality restriction, Datatypes restriction, Annotations, metamodeling**

Example for: **Extra Datatypes**

- E.g.; adults are individuals whose age is greater than 18 years.

References: [*Protege*]

7.14 Use Case #13 - Web service modeling [Telecom]

Overview: People often want to use a class to specify the value of some property. An example originating at the University of Karlsruhe [*Web Service*] is in service modeling. Services are modeled as instances of the :Service class. For each concrete service (i.e., for each instance of :Service), the users wanted to state what the input to the service is. Here is an example of a service description:

(1) :Service rdf:type owl:Class
(2) :Person rdf:type owl:Class
(3) s1 rdf:type :Service
(4) s1 :input :Person

s1 is an individual of the class :Service due to (1) and (3), and :Person is a class due to (2); hence, in (4) we have a relationship :input between an individual and a class. Hence, you need some kind of metamodeling to solve this problem. One way would be that the name 'Person' may refer both to Person as a class and as an individual denoting Person as a whole (Class ↔ Individual)

Features: **metamodeling**

Example for: **Simple metamodeling**

- E.g.; a class and an individual : *Person* may be used both for a class and an individual

References: [*Web Service*] [*Punning*]

7.15 Use Case #14 - Managing vocabulary in collaborative environments [Wiki]

Overview: It can be useful to relate schema elements (classes/properties) with each other in order to capture pragmatic relationships between them. An example observed in applications of Semantic MediaWiki (a simple but widely used OWL-based semantic content management system with light-weight expressiveness) [*OWL1.1 Wiki*] is that users wish to relate schema elements to indicate domain-specific relationships, and generally to organize ontological vocabulary. Examples are statements such as:

- "The property *is_located_in* is in the class *Deprecated_Properties* and was replaced by property *has_location*."
- "Objects of the class *City* should have a value for the property *population*." (expressed by relating class and property)

These are merely pragmatic descriptions, and no logical relationship on schema-level is intended. However, in collaborative vocabulary creation, it is relevant that users can express such intended relationships. An important aspect of Semantic MediaWiki is that users can also query for semantic information, and this is currently realized as intended by punning. Semantic MediaWiki has already been extended by using off-the-shelf OWL reasoners, and it would be desirable if such systems would be able to deal with the use of punning in such simple cases; (Class/Property ↔ Individual)

Features: **metamodeling**

Example for: **Simple metamodeling**

- E.g.; a property and an individual: to make a statement asserting that a property is an individual of the class Deprecated_properties

References: [*Wiki*] [*Punning*]

7.16 Use Case #15 - UML Association Class [Designer]

Overview: The Unified Modeling Language (UML) includes a modeling element known as an Association Class which combines the features of a UML Class and a UML Association (UML's construct for defining class to class relationships Association). The Association Class, e.g., the association between classes Person and Company, allows a modeler to define a relation as an association and reify it simultaneously. This is convenient when one wants to model attributes of relations themselves. One way to support such case might be Class and ObjectProperty punning (Class ↔ ObjectProperty).

Features: **metamodeling**

Example for: **Simple metamodeling**

- E.g.; an object property and a class: *PersonCompany* may be used both for an object property and a class.

References: [*UML Association Class*] [*Punning*]

7.17 Use Case #16 - Database federation [Designer]

Overview: Some life sciences application designer has been building a database federation scheme. The scheme involves designing an XML schema that describes the fields and values in a variety of databases, and associated query tools that, from a query interface, can write queries (in several variants of SQL) to databases that have relevant information. Those results are presented as a single integrated view. He hears that OWL and Semantic Web technologies might be a suitable technology for implementing the same functionality and making it available using Web standards, but doesn't know where to start. This application illustrates common needs of a wide community of users that would like to use their databases and can easily query them in a convivial way. This motivates a profile where conjunctive query answering is implemented using conventional relational database systems.

Features: **Profiles (OWL 2 QL)**

Example for: **Profiles**

- E.g.; OWL 2 QL profile to easily query a federation of databases in a convivial way

References: [*Who reads?*]

7.18 Use Case #17 - Tools developers [Tools]

Overview: A user adds an assertion to an ontology; however, he accidentally mistypes the IRI of an individual. It should be possible to detect this error by comparing the IRI of the individual in the axiom with the IRIs explicitly declared to be a part of the ontology: if the individual IRI has not been explicitly introduced as being in the ontology, the user should be given the opportunity to correct his error. Tools developers, such as those involved in the Protégé-OWL toolset architecture [*TOOLS*], have often expressed problems raised for e.g.; APIs [*OWL API*] due to lack of declarations. "The first problem is that OWL does not allow for explicit declarations—assertions that a certain class, property, or an individual exists in an ontology. This aspect of the OWL standard was often misinterpreted, which caused design errors in OWL APIs" [*Syntax Problem*].

Features: **Declaration**

Example for: **Declaration**

- E.g.; A person is declared to be a class of an ontology.

References: [*Syntax Problem*]

7.19 Use Case #18 - Virtual Solar Terrestrial Observatory [Earth and Space]

Overview: Numerous single discipline and multi-discipline virtual observatories (e.g., http://vsto.org , http://vmo.nasa.gov/) are beginning to use semantic technologies to provide data access and integration. A virtual observatory is a suite of software applications on a set of computers that allows users to uniformly find, access, and use resources (data, software, document, and image products and services using these) from a collection of distributed product repositories and service providers. A VO is a service that unites services and / or multiple repositories. from http://lwsde.gsfc.nasa.gov/VO_Framework_7_Jan_05.doc. Some Virtual Observatories are focusing quite heavily on provenance encoding at data ingest time (e.g., http://spcdis.hao.ucar.edu/). The Virtual Solar Terrestrial Observatory (VSTO) is a National Science Foundation and National Center for Atmospheric Research supported effort that allows researchers to find solar and solar-terrestrial data. It provides an ontology-enhanced interface to semantically-enhanced web services that help access a number of online repositories of scientific data. The background OWL ontology contains term descriptions for science terms including instruments, observatories, parameters, etc. Users essentially need to specify a description of the data they wish to retrieve which includes either a specific instrument class or a description of that class, a date range for the data taken, and the parameters. In order to specify that in relevant science terms, scientists need to be able to represent numerical ranges and comparisons going beyond the numeric support of OWL 1. The application also needs to expand to include spatial descriptions. It would use representational power if provided for spatial/geographic containment.

Requirements: **Qualified Cardinality, Datatype restriction, [Defaults]**

Example for: **Datatype restriction**

- E.g.; the range for atmosphere is above 18000 and below 19600 [feet]

References: [*VSTO*]

7.20 Use Case #19 – Semantic Provenance Capture [Earth and Space]

Overview: In an effort to provide better search capabilities over meta information in addition to scientific data, the SPCDIS effort is providing infrastructure to capture declarative descriptions of scientific provenance information at data ingest time. The initial domain of the effort is solar coronal physics. This effort requires (among other things) extended annotations as well as datatype restriction.

Features: **Datatype restriction, Extended Annotations**

Example for: **Extended annotation** to attach annotations

- E.g.; comments on axioms, such as a SubClass axiom, to express for instance that the the elements of the subclass are data generated by a log parser.

References: [*NCAR*]

7.21 Use Case #20 – Biochemical self-interaction [Chemical domain]

Overview: In Biochemistry, some biomolecules will chemical modify themselves in such a way that it has biologically important consequences. i) Protein kinases are enzymes capable of adding phosphate groups to certain amino acids found within target proteins. Some kinases, known as Auto-Phosphorylating Kinases, will add phosphate groups to certain target amino acids that are part of itself. ii) Ribozymes are catalytically active RNA molecules in which 7 natural types are known to cleave their own RNA sequences. Such cleavage may result in significant changes to viral replication, gene expression, and possibly the generation of different protein transcripts. Such catalytically active, self-cleaving RNA make up a subclass of ribozymes called Self-Cleaving Ribozymes. Such biochemical self-interaction can be captured by asserting local reflexivity of the properties.

Features: Local Reflexivity

Example for: **Local reflexivity**

- E.g.; An Auto-Phosphorylating Kinase (is a kinase that) :phosphorylates itself.

References: [*BIO*]

7.22 Use Cases Bibliography

[Medical Req]
Web ontology language requirements w.r.t expressiveness of taxonomy and axioms in medicine In Proc. of ISWC 2003
[Micro Theory]
Creation and Usage of a "Micro Theory" for Long Bone Fractures: An Experience ReportHoward Goldberg, Vipul Kashyap and Kent Spackman, In Proc. of KR-MED 2008..
[Ontology with Rules]
Ontology enriched by rules for identifying brain anatomical structures In RIF 2004, Washington, 2004. and Annex.
[Brain Imaging]
Towards an Hybrid System Using an Ontology Enriched by Rules for the Semantic Annotation of Brain MRI Images In Proc. of RR 2007
The Brain Anatomy Case Study In Proc. of Protege 2005.
[FMA]
The Foundational Model of Anatomy A

The Foundational Model of Anatomy B
The Foundational Model of Anatomy C.

[Chemistry]
Describing chemical functional groups in OWL-DL for the classification of chemical compounds Natalia Villanueva-Rosales and Michel Dumontier. In OWL: Experiences and Directions (OWLED 07), Innsbruck, Austria.
Modelling Life Sciences knowledge with OWL1.1 (OWLED 08 DC)

[Auto]
An exploratory study in an automotive company.

[OBO]
The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Barry Smith et al. .

[RO]
Relations in Biomedical Ontologies. .

[OBO2OWL]
OBO to OWL: Go to OWL1.1! (OWLED 07)
OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences In Proc. of ISWC 2007.

[Ordnance]
Experiences of using OWL at the Ordnance Survey.

[SNOMED REQ]
An examination of OWL and the requirements of a large health care terminology.

[Agence Biomedecine]
Changing Kidney Allocation Policy in France: the Value of Simulation.

[Transplant Ontology]
Construction of the dialysis and transplantation ontology.

[Little Web]
A little semantic web goes a long way in biology Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Stevens, R., Turi, D. In Proceedings of the 2005 International Semantic Web Conference (ISWC 2005), pp. 786-800. Springer, Berlin Heidelberg New York (2005).

[Part Whole]
Simple part-whole relations in OWL Ontologies Alan Rector, Chris Welty. W3C Editor's Draft 11 Aug 2005 .

[TOOLS]
Supporting Early Adoption of OWL 1.1 with Protege-OWL and FaCT++. Matthew Horridge and Dmitry Tsarkov and Timothy Redmond. In OWL: Experiences and Directions (OWLED 06), Athens, Georgia.

[OWL API]
Igniting the OWL 1.1 Touch Paper: The OWL API Matthew Horridge and Sean Bechhofer and Olaf Noppens (2007). In OWL: Experiences and Directions (OWLED 07), Innsbruck, Austria.

[Protege OWL]
The Protégé OWL Experience Holger Knublauch, Matthew Horridge, Mark Musen, Alan Rector, Robert Stevens, Nick Drummond, Phil Lord, Natalya F.

Noy2, Julian Seidenberg, Hai Wang. In OWL: Experiences and Directions (OWLED 05), Galway, Ireland, 2005.

[N-ary]
N-ary Data predicate use case.

[Web Service]
Preference-based Selection of Highly Configurable Web Services Steffen Lamparter, Anupriya Ankolekar, Stephan Grimm, Rudi Studer: WWW-07, Banff, Canada, 2007.

[Wiki]
Reusing Ontological Background Knowledge in Semantic Wikis Denny Vrandecic, Markus Krötzsch, Proceedings 1st Workshop on Semantic Wikis. Budva, Montenegro, June 2006 .

[UML Association Class]
Association.

[Punning]
Punning Use Cases.

[Who reads?]
Who reads our documents?
NIF
NIF Data-Integration slides

[VSTO]
The Virtual Solar-Terrestrial Observatory: A Deployed Semantic Web Application Case Study for Scientific Research McGuinness, D.L., Fox, P., Cinquini, L., West, P., Garcia, J., Benedict, J.L., Middleton, D..
VSTO2.
VMO.

[NCAR]
Semantic Provenance Capture in Data Ingest Systems .

[BIO]
Springer.
pnas.

[SKOS]
W3C Working Draft 29 August 2008 .

8 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 Elisa Kendall (Sandpiper Software), Peter F. Patel-Schneider (Bell Labs Research, Alcatel-Lucent) and Alan Ruttenberg (Science Commons) for their thorough reviews.

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