OWL 2
The Next Generation

Ian Horrocks
<ian.horrocks@comlab.ox.ac.uk>
Information Systems Group
Oxford University Computing Laboratory

The Web Ontology Language OWL

- Motivated by **Semantic Web** activity

  Add meaning to web content by annotating it with terms defined in ontologies
- Developed by W3C WebOnt working group
  - Based on earlier languages **RDF**, **OIL** and **DAML+OIL**
  - Became a **recommendation** on 10 Feb 2004
- Supported by **tools and infrastructure**
  - APIs (e.g., OWL API, Thea, OWLink)
  - Development environments (e.g., Protégé, TopBraid Composer)
  - Reasoners & Information Systems (e.g., Pellet, HermiT, Quonto)
- Based on a **Description Logic** ($\mathcal{SHOIN}$)

Experience with OWL

- OWL playing **key role** in increasing number & range of applications
  - eScience, eCommerce, geography, engineering, defence, …
  - E.g., OWL tools used to identify and repair errors in a medical ontology:
    "***would have led to missed test results if not corrected***"
  - E.g., BBC World Cup website powered by RDF metadata and OWL ontology
- Experience of **OWL in use** has identified restrictions:
  - on expressivity
  - on scalability

  These restrictions are problematic in some applications
- **Research** has now shown how some restrictions can be overcome
- W3C OWL WG has updated OWL accordingly; result called OWL 2
- OWL 2 is now a **W3C Recommendation** (supersedes OWL)

OWL 2 in a Nutshell

- **Extends OWL** with a small but useful set of features
  - That are needed in applications
  - For which semantics and reasoning techniques are well understood
  - That tool builders are willing and able to support
- Adds **profiles**
  - Language subsets with useful computational properties
- Is **fully backwards compatible** with OWL:
  - Every OWL ontology is a valid OWL 2 ontology
  - Every OWL 2 ontology not using new features is a valid OWL ontology
- Already supported by popular **OWL tools** & infrastructure:
  - Protégé, HermiT, Pellet, FaCT++, OWL API

What's New in OWL 2?

Four kinds of new feature:

- **Increased expressive power**
  - qualified cardinality restrictions, e.g.:
    - persons having two friends who are republicans
  - property chains, e.g.:
    - the brother of your parent is your uncle
  - local reflexivity restrictions, e.g.:
    - narcissists love themselves
  - reflexive, irreflexive, and asymmetric properties, e.g.:
    - nothing can be a proper part of itself (irreflexive)
  - disjoint properties, e.g.:
    - you can't be both the parent of and child of the same person
  - keys, e.g.:
    - country + license plate constitute a unique identifier for vehicles

What’s New in OWL 2?

Four kinds of new feature:

- **Extended Datatypes**

What's New in OWL 2?

Four kinds of new feature:

- **Extended Datatypes**
  - Much wider range of XSD Datatypes supported, e.g.:

    Integer, string, boolean, real, decimal, float, datatime, …

What’s New in OWL 2?

Four kinds of new feature:

- **Extended Datatypes**
  - Much wider range of XSD Datatypes supported, e.g.:

    Integer, string, boolean, real, decimal, float, datatime, …
  - User-defined datatypes using facets, e.g.:

max weight of an airmail letter:
xsd:integer maxInclusive ”20"^^xsd:integer

What's New in OWL 2?

Four kinds of new feature:

- **Extended Datatypes**
  - Much wider range of XSD Datatypes supported, e.g.:

    Integer, string, boolean, real, decimal, float, datatime, …
  - User-defined datatypes using facets, e.g.:

max weight of an airmail letter:
xsd:integer maxInclusive ”20"^^xsd:integer

format of Italian registration plates:
xsd:string xsd:pattern "[A-Z]{2} [0-9]{3}[A-Z]{2}

What's New in OWL 2?

Four kinds of new feature:

- **Metamodelling and annotations**
  - Restricted form of metamodelling via "punning", e.g.:
    - SnowLeopard subClassOf BigCat (i.e., a class)
    - SnowLeopard type EndangeredSpecies (i.e., an individual)
  - Annotations of axioms as well as entities, e.g.:
    - SnowLeopard type EndangeredSpecies ("source: WWF")
  - Even annotations of annotations

What's New in OWL 2?

Four kinds of new feature:

- **Syntactic sugar**
  - Disjoint unions, e.g.:

    Element is the DisjointUnion of Earth Wind Fire Water

    i.e., Element is equivalent to the union of Earth Wind Fire Water

    Earth Wind Fire Water are pair-wise disjoint
  - Negative assertions, e.g.:

    Mary is not a sister of Ian

    21 is not the age of Ian

Alternative Syntaxes

- Normative exchange syntax is RDF/XML

```
<owl:Class rdf:about="#Heart">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#MuscularOrgan"/>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isPartOf"/>
                    <owl:someValuesFrom rdf:resource="#CirculatorySystem"/>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
```

Alternative Syntaxes

- Normative exchange syntax is RDF/XML
- Functional syntax mainly intended for language spec

```
EquivalentClasses(Heart
   ObjectIntersectionOf(ObjectSomeValuesFrom(isPartOf CirculatorySystem)
      MuscularOrgan))
```

Alternative Syntaxes

- Normative exchange syntax is RDF/XML
- Functional syntax mainly intended for language spec
- XML syntax for interoperability with XML toolchain

```
<EquivalentClasses>
    <Class URI="Heart"/>
    <ObjectIntersectionOf>
        <Class URI="MuscularOrgan"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty URI="isPartOf"/>
            <Class URI="CirculatorySystem"/>
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

Alternative Syntaxes

- Normative exchange syntax is RDF/XML
- Functional syntax mainly intended for language spec
- XML syntax for interoperability with XML toolchain
- Manchester syntax for better readability

```
         Class: Heart
  EquivalentTo: MuscularOrgan
                that isPartOf CirculatorySystem
```

Profiles

- OWL only **useful in practice** if we can deal with large ontologies and/or large data sets
- Unfortunately, OWL is worst case highly intractable
  - OWL 2 ontology satisfiability is **2NEXPTIME-complete**
- Possible solution is **profiles**: language subsets with useful computational properties
- OWL defined one such profile: **OWL Lite**
  - Unfortunately, it isn't tractable either! (EXPTIME-complete)

Profiles

- OWL 2 defines three different tractable profiles:
  - **EL**: polynomial time reasoning for schema and data
    - Useful for ontologies with large conceptual part
  - **QL**: fast (logspace) query answering using RDBMs via SQL
    - Useful for large datasets already stored in RDBs
  - **RL**: fast (polynomial) query answering using rule-extended DBs
    - Useful for large datasets stored as RDF triples

OWL 2 EL

- A (near maximal) fragment of OWL 2 such that
  - Satisfiability checking is in PTime (**PTime-Complete**)
  - Data complexity of query answering also PTime-Complete
- Based on $\mathcal{EL}$ family of description logics
  - Existential (someValuesFrom) + conjunction
- Can exploit **saturation** based reasoning techniques
  - Computes classification in “one pass”
  - Computationally optimal
  - Can be extended to Horn fragment of OWL DL

Saturation-based Technique (basics)

- Normalise ontology axioms to standard form:

$$A \sqsubseteq B \quad A \sqcap B \sqsubseteq C \quad A \sqsubseteq \exists R.B \quad \exists R.B \sqsubseteq C$$

- Saturate using inference rules:

$$\frac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C} \qquad \frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C \quad \exists R.C \sqsubseteq D}{A \sqsubseteq D}$$

- Extension to Horn fragment requires (many) more rules

Saturation-based Technique

Performance with large bio-medical ontologies:

	GO	NCI	Galen v.0	Galen v.7	SNOMED
Concepts:	20465	27652	2748	23136	389472
FaCT++	15.24	6.05	465.35	—	650.37
HermiT	199.52	169.47	45.72	—	—
Pellet	72.02	26.47	—	—	—
CEL	1.84	5.76	—	—	1185.70
CB	1.17	3.57	0.32	9.58	49.44
Speed-Up:	1.57X	1.61X	143X	∞	13.15X

OWL 2 QL

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in **AC^0**
- Based on **DL-Lite** family of description logics
  - Existential (someValuesFrom) + conjunction (RHS only)
- Can exploit **query rewriting** based reasoning technique
  - Computationally optimal
  - Data storage and query evaluation can be delegated to standard RDBMS
  - Can be extended to more expressive languages (beyond AC^0) by delegating query answering to a Datalog engine

Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $\mathcal{Q}$, use $\mathcal{O}$ to rewrite $\mathcal{Q}$ as $\mathcal{Q}'$ s.t., for any set of ground facts $\mathcal{A}$:
  - $\text{ans}(\mathcal{Q}, \mathcal{O}, \mathcal{A}) = \text{ans}(\mathcal{Q}', \emptyset, \mathcal{A})$
- Resolution based query rewriting
  - **Clausify** ontology axioms
  - **Saturate** (clausified) ontology and query using resolution
  - **Prune** redundant query clauses

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$

$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \land \mathsf{Patient}(y)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$

$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$

$$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$$

$$Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)$$

$$Q(x) \leftarrow \mathsf{Doctor}(x)$$

$$Q(x) \leftarrow \mathsf{Consultant}(x)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$

$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$

~~$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$~~

~~$Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)$~~

$$Q(x) \leftarrow \mathsf{Doctor}(x)$$

$$Q(x) \leftarrow \mathsf{Consultant}(x)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$
$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \land \mathsf{Patient}(y)$$
$$\require{cancel}\cancel{Q(x) \leftarrow \mathsf{Doctor}(x) \land \mathsf{Patient}(f(x))}$$
$$\cancel{Q(x) \leftarrow \mathsf{treats}(x, f(x)) \land \mathsf{Doctor}(x)}$$
$$Q(x) \leftarrow \mathsf{Doctor}(x)$$
$$Q(x) \leftarrow \mathsf{Consultant}(x)$$

- For DL-Lite, result is a union of conjunctive queries

Query Rewriting Technique (basics)

- Data can be stored/left in **RDBMS**
- Relationship between ontology and DB defined by **mappings**, e.g.:

Doctor	$\mapsto$	SELECT Name FROM Doctor
Patient	$\mapsto$	SELECT Name FROM Patient
treats	$\mapsto$	SELECT DName, PName FROM Treats

- UCQ translated into **SQL query**:

```
SELECT Name FROM Doctor UNION
SELECT DName FROM Treats, Patient WHERE PName=Name
```

OWL 2 RL

- A (near maximal) fragment of OWL 2 such that
  - Can be implemented using standard rule engines
- Closely related to **Description Logic Programms (DLP)**
  - No "existentials" on RHS
  - Suffices to consider Herbrand models
- Can provide **correctness guarantees**
  - For conformant ontologies and atomic queries
  - In other cases results may be incomplete

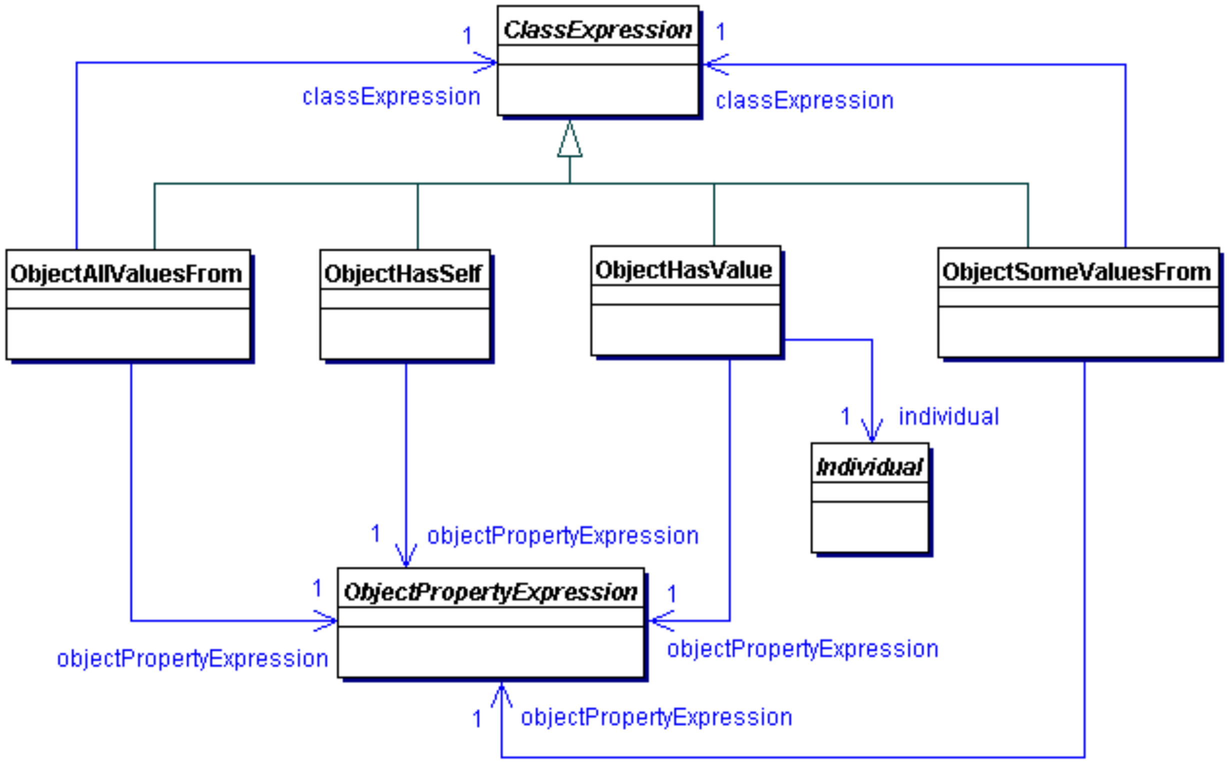
Last but not Least

Better quality spec

Last but not Least

Better quality spec

- Syntax spec uses UML (as well as functional syntax)



Last but not Least

Better quality spec

- Syntax spec uses UML (as well as functional syntax)
- Deterministic and bi-directional RDF mapping
- Fully formed XML and human readable syntaxes
- Several user facing documents, including Quick Ref

OWL 2 Web Ontology Language Quick Reference Guide

http://www.w3.org/2007/OWL/refcard

1 Names, Prefixes, and Notation

Names in OWL 2 are IRIs, often written in a shorthand prefix:local_name, where prefix: is a prefix name that expands to an IRI, and local_name is the remainder of the name. The prefix names in OWL 2 are:

Prefix Name	Expansion
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
xsd:	http://www.w3.org/2001/XMLSchema#

We use notation conventions in the following table*:

Letters	Meaning	Letters	Meaning
(a1 … an)	RDF list	n	non-negative integer**
_:a	anonymous individual (a blank node label)	ON	ontology name
_:x	blank node	P	object property expression
a	individual	p	prefix name
A	annotation property	PN	object property name
aN	individual name	R	data property
C	class expression	s	IRI or anonymous individual
CN	class name	t	IRI, anonymous individual, or literal
D	data range	U	IRI
DN	datatype name	v	literal
f	facet		

* All of the above can have subscripts.
** As a shorthand for "n"^^xsd:nonNegativeInteger

2 OWL 2 constructs and axioms

In the following tables, the three columns are:

Language Feature	Functional Syntax	RDF Syntax

For an OWL 2 DL ontology, there are additional global restrictions on axioms.

2.1 Class Expressions

Predefined and Named Classes

named class	CN	CN
universal class	owl:Thing	owl:Thing
empty class	owl:Nothing	owl:Nothing

Boolean Connectives and Enumeration of Individuals

intersection	ObjectIntersectionOf (C1…Cn)	_:x rdf:type owl:Class. _:x owl:intersectionOf (C1…Cn).
union	ObjectUnionOf (C1 … Cn)	_:x rdf:type owl:Class. _:x owl:unionOf (C1 … Cn).
complement	ObjectComplementOf (C)	_:x rdf:type owl:Class. _:x owl:complementOf C.
enumeration	ObjectOneOf(a1 … an)	_:x rdf:type owl:Class. _:x owl:oneOf (a1 … an).

Object Property Restrictions

universal	ObjectAllValuesFrom (P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:allValuesFrom C
existential	ObjectSomeValues From(P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:someValuesFrom C
individual value	ObjectHasValue(P a)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasValue a.
local reflexivity	ObjectHasSelf(P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasSelf "true"^^xsd:boolean.
exact cardinality	ObjectExactCardinality (n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:cardinality n.
qualified exact cardinality	ObjectExactCardinality (n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:qualifiedCardinality n. _:x owl:onClass C.
maximum cardinality	ObjectMaxCardinality (n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:minCardinality n.
qualified maximum cardinality	ObjectMaxCardinality (n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:minQualifiedCardinality n. _:x owl:onClass C.
minimum cardinality	ObjectMinCardinality (n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:maxCardinality n.
qualified minimum cardinality	ObjectMinCardinality (n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:maxQualifiedCardinality n. _:x owl:onClass C.

Data Property Restrictions

universal	DataAllValuesFrom (R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:allValuesFrom D.
existential	DataSomeValuesFrom (R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:someValuesFrom D.
literal value	DataHasValue (R v)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:hasValue v.
exact cardinality	DataExactCardinality (n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:cardinality n.
qualified exact cardinality	DataExactCardinality (n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:qualifiedCardinality n. _:x owl:onDataRange D.
maximum cardinality	DataMaxCardinality (n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxCardinality n.
qualified maximum cardinality	DataMaxCardinality (n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxQualifiedCardinality n. _:x owl:onDataRange D.
minimum cardinality	DataMinCardinality (n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minCardinality n.
qualified minimum cardinality	DataMinCardinality (n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minQualifiedCardinality n. _:x owl:onDataRange D.

Restrictions Using n-ary Data Range

In the following table 'Dn' is an n-ary data range.

n-ary universal	DataAllValuesFrom (R1 … Rn Dn)	_:x rdf:type owl:Restriction. _:x owl:onProperties (R1 … Rn). _:x owl:allValuesFrom Dn.
n-ary existential	DataSomeValuesFrom (R1 … Rn Dn)	_:x rdf:type owl:Restriction. _:x owl:onProperties (R1 … Rn). _:x owl:someValuesFrom Dn.

2.2 Properties

Object Property Expressions

named object property	PN	PN
universal object property	owl:topObjectProperty	owl:topObjectProperty
empty object property	owl:bottomObjectProperty	owl:bottomObjectProperty
inverse property	ObjectInverseOf(PN)	_:x owl:inverseOf PN

Data Property Expressions

named data property	R	R
universal data property	owl:topDataProperty	owl:topDataProperty
empty data property	owl:bottomDataProperty	owl:bottomDataProperty

2.3 Individuals & Literals

named individual	aN	aN
anonymous individual	_:a	_:a
literal (datatype value)	"abc"^^DN	"abc"^^DN

2.4 Data Ranges

Data Range Expressions

named datatype	DN	DN
data range complement	DataComplementOf (D)	_:x rdf:type rdfs:Datatype. _:x owl:datatypeComplementOf D.
data range intersection	DataIntersectionOf (D1…Dn)	_:x rdf:type rdfs:Datatype. _:x owl:intersectionOf (D1…Dn).
data range union	DataUnionOf (D1…Dn)	_:x rdf:type rdfs:Datatype. _:x owl:unionOf (D1…Dn).
literal enumeration	DataOneOf (v1 … vn)	_:x rdf:type rdfs:Datatype. _:x owl:oneOf (v1 … vn).
datatype restriction	DatatypeRestriction (DN f1 v1 … fn vn)	_:x rdf:type rdfs:Datatype. _:x owl:onDatatype DN. _:x owl:withRestrictions (_:x1 … _:xn). _:xj fj vj. j=1…n

2.5 Axioms

Class Expression Axioms

subclass	SubClassOf(C1 C2)	C1 rdfs:subClassOf C2.
equivalent classes	EquivalentClasses (C1 … Cn)	Cj owl:equivalentClass Cj+1. j=1…n-1
disjoint classes	DisjointClasses(C1 C2)	C1 owl:disjointWith C2.
pairwise disjoint classes	DisjointClasses (C1 … Cn)	_:x rdf:type owl:AllDisjointClasses. _:x owl:members (C1 … Cn).
disjoint union	DisjointUnionOf (CN C1 … Cn)	CN owl:disjointUnionOf (C1 … Cn).

Object Property Axioms

subproperty	SubObjectPropertyOf (P1 P2)	P1 rdfs:subPropertyOf P2.
property chain inclusion	SubObjectPropertyOf (ObjectPropertyChain (P1 … Pn) P)	P owl:propertyChainAxiom (P1 … Pn).
property domain	ObjectPropertyDomain (P C)	P rdfs:domain C.
property range	ObjectPropertyRange(P C)	P rdfs:range C.
equivalent properties	EquivalentObjectProperties (P1 … Pn)	Pj owl:equivalentProperty Pj+1. j=1…n-1
disjoint properties	DisjointObjectProperties (P1 P2)	P1 owl:propertyDisjointWith P2.
pairwise disjoint properties	DisjointObjectProperties (P1 … Pn)	_:x rdf:type owl:AllDisjointProperties. _:x owl:members (P1 … Pn).
inverse properties	InverseObjectProperties (P1 P2)	P1 owl:inverseOf P2.

OWL 2 Documentation Roadmap

Part	Type	Document
1	For Users	Document Overview. A quick overview of the OWL 2 specification that includes a description of its relationship to OWL 1. This it the starting point and primary reference point for OWL 2.
2	Core Specification	Structural Specification and Functional-Style Syntax defines the constructs of OWL 2 ontologies in terms of both their structure and a functional-style syntax, and defines OWL 2 DL ontologies in terms of global restrictions on OWL 2 ontologies.
3	Core Specification	Mapping to RDF Graphs defines a mapping of the OWL 2 constructs into RDF graphs, and thus defines the primary means of exchanging OWL 2 ontologies in the Semantic Web.
4	Core Specification	Direct Semantics defines the meaning of OWL 2 ontologies in terms of a model-theoretic semantics.
5	Core Specification	RDF-Based Semantics defines the meaning of OWL 2 ontologies via an extension of the RDF Semantics.
6	Core Specification	Conformance provides requirements for OWL 2 tools and a set of test cases to help determine conformance.
7	Specification	Profiles defines three sub-languages of OWL 2 that offer important advantages in particular applications scenarios.
8	For Users	OWL 2 Primer provides an approachable introduction to OWL 2, including orientation for those coming from other disciplines.
9	For Users	OWL 2 New Features and Rationale provides an overview of the main new features of OWL 2 and motivates their inclusion in the language.
10	For Users	OWL 2 Quick Reference Guide provides a brief guide to the constructs of OWL 2, noting the changes from OWL 1.
11	Specification	XML Serialization defines an XML syntax for exchanging OWL 2 ontologies, suitable for use with XML tools like schema-based editors and XQuery/XPath.
12	Specification	Manchester Syntax (WG Note) defines an easy-to-read, but less formal, syntax for OWL 2 that is used in some OWL 2 user interface tools and is also used in the Primer.
13	Specification	Data Range Extension: Linear Equations (WG Note) specifies an optional extension to OWL 2 which supports advanced constraints on the values of properties.

Thank you for listening

Any questions?

Resources:

- OWL 2 Recommendation
  - http://www.w3.org/TR/owl2-overview/