OWL 2
The Next Generation

Ian Horrocks
<ian.horrocks@comlab.ox.ac.uk>
Information Systems Group
Oxford University Computing Laboratory
What is an Ontology?
What is an Ontology?

A model of (some aspect of) the world
What is an Ontology?

A model of (some aspect of) the world

• Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
What is an Ontology?

A model of (some aspect of) the world

• Introduces **vocabulary** relevant to domain, e.g.:
  – Anatomy
  – Cellular biology
What is an Ontology?

A model of (some aspect of) the world

• Introduces **vocabulary**
  relevant to domain, e.g.:
  – Anatomy
  – Cellular biology
  – Aerospace
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace
  - Dogs
What is an Ontology?

A model of (some aspect of) the world

• Introduces **vocabulary**
  relevant to domain, e.g.:
  – Anatomy
  – Cellular biology
  – Aerospace
  – Dogs
  – Hotdogs
  – …
What is an Ontology?

A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain
- Specifies **meaning** of terms

Heart is a muscular organ that is part of the circulatory system
What is an Ontology?
A model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain
- Specifies **meaning** of terms
  
  Heart is a muscular organ that is part of the circulatory system

- **Formalised** using suitable logic

\[
\forall x. [\text{Heart}(x) \rightarrow \text{MuscularOrgan}(x) \land \\
\exists y. [\text{isPartOf}(x, y) \land \\
\text{CirculatorySystem}(y)]]
\]
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** (**SHOIN**)
Description Logics (DLs)

• Fragments of first order logic designed for KR

• Desirable computational properties
  – Decidable (essential)
  – Low complexity (desirable)

• Succinct and quantifier free syntax

\[ \forall x. [\text{Heart}(x) \rightarrow \text{MuscularOrgan}(x) \land \exists y. [\text{isPartOf}(x, y) \land \text{CirculatorySystem}(y)]] \]

\[ \text{Heart} \sqsubseteq \text{MuscularOrgan} \sqcap \exists \text{isPartOf}. \text{CirculatorySystem} \]
Description Logics (DLs)

DL Knowledge Base (KB) consists of two parts:

- Ontology (aka TBox) axioms define terminology (schema)
  
  \[
  \text{Heart} \sqsubseteq \text{MuscularOrgan} \sqsubseteq \\
  \exists \text{isPartOf}.\text{CirculatorySystem} \\
  \text{HeartDisease} \equiv \text{Disease} \sqsubseteq \\
  \exists \text{affects}.\text{Heart} \\
  \text{VascularDisease} \equiv \text{Disease} \sqsubseteq \\
  \exists \text{affects}.(\exists \text{isPartOf}.\text{CirculatorySystem})
  \]

- Ground facts (aka ABox) use the terminology (data)

  \[
  \text{John} : \text{Patient} \sqsubseteq \\
  \exists \text{suffersFrom}.\text{HeartDisease}
  \]
What are Ontologies Good For?

- Coherent **user-centric view** of domain
  - Help identify and resolve disagreements

- Ontology-based **Information Systems**
  - View of data that is independent of logical/physical schema
  - Queries use terms familiar to users
  - Answers reflect knowledge & data, e.g.:
    - “Patients suffering from Vascular Disease”
  - Query navigation/refinement
  - Incomplete and semi-structured data
  - Integration of heterogeneous sources

Now... *that* should clear up a few things around here
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”

• 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 is called OWL 2

• OWL 2 is now a **Proposed Recommendation**
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, datetime, …
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, datetime, ...
  - 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, datetime, …
  - 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.:
    
    \[
    \text{SnowLeopard} \text{ subClassOf BigCat} \quad \text{(i.e., a class)}
    \]
    
    \[
    \text{SnowLeopard} \text{ type EndangeredSpecies} \quad \text{(i.e., an individual)}
    \]
  - Annotations of axioms as well as entities, e.g.:
    
    \[
    \text{SnowLeopard} \text{ type EndangeredSpecies ("source: WWF")}
    \]
  - 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

```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

```xml
<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 \(2\text{NEXPTIME-complete}\)

• 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 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:
  \[ A \subseteq B \quad A \cap B \subseteq C \quad A \subseteq \exists R.B \quad \exists R.B \subseteq C \]

- Saturate using inference rules:
  \[
  \begin{align*}
  A \subseteq B & \quad B \subseteq C \\
  & \quad \ \Rightarrow \quad A \subseteq C \\
  A \subseteq B & \quad A \subseteq C \\
  & \quad \ \Rightarrow \quad B \cap C \subseteq D \\
  \end{align*}
  \]

- Extension to Horn fragment requires (many) more rules
Saturation-based Technique (basics)

Example:

\[
\text{Organ Transplant} \equiv \text{Transplant} \land \exists \text{site.Organ} \\
\text{Heart Transplant} \equiv \text{Transplant} \land \exists \text{site.Heart} \\
\text{Heart} \subseteq \text{Organ}
\]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \land \exists \text{site.Organ}
\]

\[
\text{HeartTransplant} \equiv \text{Transplant} \land \exists \text{site.Heart}
\]

\[
\text{Heart} \subseteq \text{Organ}
\]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Organ}
\]

\[
\text{HeartTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Heart}
\]

\[
\text{Heart} \subseteq \text{Organ}
\]

\[
\text{OrganTransplant} \subseteq \text{Transplant}
\]

\[
\text{OrganTransplant} \subseteq \exists \text{site.Organ}
\]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
\text{Heart} \sqsubseteq \text{Organ}
\]

\[
\begin{align*}
\text{OrganTransplant} & \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} & \sqsubseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} & \sqsubseteq \text{SO} \\
\text{Transplant} \sqcap \text{SO} & \sqsubseteq \text{OrganTransplant}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
\text{Heart} \sqsubseteq \text{Organ}
\]

\[
\text{OrganTransplant} \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} \sqsubseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} \sqsubseteq \text{SO} \\
\text{Transplant} \sqcap \text{SO} \sqsubseteq \text{OrganTransplant}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
\quad \text{Heart} \sqsubseteq \text{Organ} \\
\text{OrganTransplant} & \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} & \sqsubseteq \exists \text{site.Organ} \\
\quad \exists \text{site.Organ} & \sqsubseteq \text{SO} \\
\text{Transplant} & \sqcap \text{SO} \sqsubseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \sqsubseteq \text{Transplant} \\
\text{HeartTransplant} & \sqsubseteq \exists \text{site.Heart} \\
\quad \exists \text{site.Heart} & \sqsubseteq \text{SH} \\
\text{Transplant} & \sqcap \text{SH} \sqsubseteq \text{HeartTransplant}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\text{OrganTransplant} \equiv \text{Transplant} \cap \exists \text{site.Organ} \\
\text{HeartTransplant} \equiv \text{Transplant} \cap \exists \text{site.Heart} \\
\text{Heart} \subseteq \text{Organ}
\]

\[
\text{OrganTransplant} \subseteq \text{Transplant} \\
\text{OrganTransplant} \subseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} \subseteq \text{SO} \\
\text{Transplant} \cap \text{SO} \subseteq \text{OrganTransplant} \\
\text{HeartTransplant} \subseteq \text{Transplant} \\
\text{HeartTransplant} \subseteq \exists \text{site.Heart} \\
\exists \text{site.Heart} \subseteq \text{SH} \\
\text{Transplant} \cap \text{SH} \subseteq \text{HeartTransplant}
\]
Saturation-based Technique (basics)

Example:

\[ \text{Organ Transplant} \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \]
\[ \text{Heart Transplant} \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \]
\[ \text{Heart} \sqsubseteq \text{Organ} \]

\[ \text{Organ Transplant} \sqsubseteq \text{Transplant} \]
\[ \text{Organ Transplant} \sqsubseteq \exists \text{site.Organ} \]
\[ \exists \text{site.Organ} \sqsubseteq \text{SO} \]
\[ \text{Transplant} \sqcap \text{SO} \sqsubseteq \text{Organ Transplant} \]
\[ \text{Heart Transplant} \sqsubseteq \text{Transplant} \]
\[ \text{Heart Transplant} \sqsubseteq \exists \text{site.Heart} \]
\[ \exists \text{site.Heart} \sqsubseteq \text{SH} \]
\[ \text{Transplant} \sqcap \text{SH} \sqsubseteq \text{Heart Transplant} \]
\[ \text{Heart} \sqsubseteq \text{Organ} \]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
& \quad \text{Heart} \sqsubseteq \text{Organ}
\end{align*}
\]

\[
\begin{align*}
\text{OrganTransplant} & \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} & \sqsubseteq \exists \text{site.Organ} \\
& \quad \exists \text{site.Organ} \sqsubseteq \text{SO} \\
\text{Transplant} \sqcap \text{SO} & \sqsubseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \sqsubseteq \text{Transplant} \\
\text{HeartTransplant} & \sqsubseteq \exists \text{site.Heart} \\
& \quad \exists \text{site.Heart} \sqsubseteq \text{SH} \\
\text{Transplant} \sqcap \text{SH} & \sqsubseteq \text{HeartTransplant} \\
& \quad \text{Heart} \sqsubseteq \text{Organ}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} &\equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} &\equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
\text{Heart} &\subseteq \text{Organ}
\end{align*}
\]

\[
\begin{align*}
A &\subseteq \exists R.B \\
B &\subseteq C \\
\exists R.C &\subseteq D \\
A &\subseteq D \quad \text{HeartTransplant} \subseteq \text{SO}
\end{align*}
\]

\[
\begin{align*}
\text{OrganTransplant} &\subseteq \text{Transplant} \\
\text{OrganTransplant} &\subseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} &\subseteq \text{SO} \\
\text{Transplant} &\sqcap \text{SO} \subseteq \text{OrganTransplant} \\
\text{HeartTransplant} &\subseteq \text{Transplant} \\
\text{HeartTransplant} &\subseteq \exists \text{site.Heart} \\
\exists \text{site.Heart} &\subseteq \text{SH} \\
\text{Transplant} &\sqcap \text{SH} \subseteq \text{HeartTransplant} \\
\text{Heart} &\subseteq \text{Organ}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} &\equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} &\equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
\text{Heart} &\sqsubseteq \text{Organ} \\
A &\subseteq B \\
A &\subseteq C \\
B \sqcap C &\subseteq D \\
A &\subseteq D \\
\text{HeartTransplant} &\sqsubseteq \text{SO}
\end{align*}
\]
Saturation-based Technique (basics)

Example:

\[
\begin{align*}
\text{OrganTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site.Organ} \\
\text{HeartTransplant} & \equiv \text{Transplant} \sqcap \exists \text{site.Heart} \\
\text{Heart} & \sqsubseteq \text{Organ} \\
\text{OrganTransplant} & \sqsubseteq \text{Transplant} \\
\text{OrganTransplant} & \sqsubseteq \exists \text{site.Organ} \\
\exists \text{site.Organ} & \sqsubseteq \text{SO} \\
\text{Transplant} \sqcap \text{SO} & \sqsubseteq \text{OrganTransplant} \\
\text{HeartTransplant} & \sqsubseteq \text{Transplant} \\
\text{HeartTransplant} & \sqsubseteq \exists \text{site.Heart} \\
\exists \text{site.Heart} & \sqsubseteq \text{SH} \\
\text{Transplant} \sqcap \text{SH} & \sqsubseteq \text{HeartTransplant} \\
\text{Heart} & \sqsubseteq \text{Organ}
\end{align*}
\]
### Saturation-based Technique

Performance with large bio-medical ontologies:

<table>
<thead>
<tr>
<th></th>
<th>GO</th>
<th>NCI</th>
<th>Galen v.0</th>
<th>Galen v.7</th>
<th>SNOMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts:</td>
<td>20465</td>
<td>27652</td>
<td>2748</td>
<td>23136</td>
<td>389472</td>
</tr>
<tr>
<td>FACT++</td>
<td>15.24</td>
<td>6.05</td>
<td>465.35</td>
<td>—</td>
<td>650.37</td>
</tr>
<tr>
<td>HERMIT</td>
<td>199.52</td>
<td>169.47</td>
<td>45.72</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PELLET</td>
<td>72.02</td>
<td>26.47</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CEL</td>
<td>1.84</td>
<td>5.76</td>
<td>—</td>
<td>—</td>
<td>1185.70</td>
</tr>
<tr>
<td>CB</td>
<td>1.17</td>
<td>3.57</td>
<td>0.32</td>
<td>9.58</td>
<td>49.44</td>
</tr>
<tr>
<td>Speed-Up:</td>
<td>1.57X</td>
<td>1.61X</td>
<td>143X</td>
<td>∞</td>
<td>13.15X</td>
</tr>
</tbody>
</table>
**OWL 2 QL**

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in $\text{AC}^0$
- 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 $\text{AC}^0$) 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:

\[
\text{Doctor } \subseteq \exists \text{treats.Patient} \\
\text{Consultant } \subseteq \text{Doctor}
\]

\[Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y)\]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \subseteq \exists \text{treats.Patient} \\
\text{Consultant} & \subseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x) \\
\end{align*}
\]

\[Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y)\]
Query Rewriting Technique (basics)

• Example:

\[
\begin{align*}
\text{Doctor} & \sqsubseteq \exists \text{treats.Patient} \\
\text{Consultant} & \sqsubseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y)\]
Query Rewriting Technique (basics)

- Example:

\[ \text{Doctor} \subseteq \exists \text{treats.Patient} \]
\[ \text{Consultant} \subseteq \text{Doctor} \]

\[ \text{treats}(x, f(x)) \leftarrow \text{Doctor}(x) \]
\[ \text{Patient}(f(x)) \leftarrow \text{Doctor}(x) \]
\[ \text{Doctor}(x) \leftarrow \text{Consultant}(x) \]

\[ Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \]
\[ Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \]
Query Rewriting Technique (basics)

- Example:

\[
\begin{align*}
\text{Doctor} & \subseteq \exists \text{treats.Patient} \\
\text{Consultant} & \subseteq \text{Doctor} \\
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x))
\end{align*}
\]
Query Rewriting Technique (basics)

- Example:

\[\text{Doctor} \sqsubseteq \exists \text{treats.Patient}\]
\[\text{Consultant} \sqsubseteq \text{Doctor}\]

\[
\begin{align*}
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[ \text{Doctor} \sqsubseteq \exists \text{treats.Patient} \]
\[ \text{Consultant} \sqsubseteq \text{Doctor} \]

\[ \text{treats}(x, f(x)) \leftarrow \text{Doctor}(x) \]
\[ \text{Patient}(f(x)) \leftarrow \text{Doctor}(x) \]
\[ \text{Doctor}(x) \leftarrow \text{Consultant}(x) \]

\[ Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \]
\[ Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \]
\[ Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \]
Query Rewriting Technique (basics)

- Example:

\[
\begin{align*}
\text{Doctor} & \subseteq \exists \text{treats.Patient} \\
\text{Consultant} & \subseteq \text{Doctor}
\end{align*}
\]

\[
\begin{align*}
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

• Example:

\[
\text{Doctor} \subseteq \exists \text{treats.Patient}
\]

\[
\text{Consultant} \subseteq \text{Doctor}
\]

\[
\text{treats}(x, f(x)) \leftarrow \text{Doctor}(x)
\]

\[
\text{Patient}(f(x)) \leftarrow \text{Doctor}(x)
\]

\[
\text{Doctor}(x) \leftarrow \text{Consultant}(x)
\]

\[
Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y)
\]

\[
Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x))
\]

\[
Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x)
\]

\[
Q(x) \leftarrow \text{Doctor}(x)
\]
Query Rewriting Technique (basics)

- Example:

\[
\text{Doctor} \subseteq \exists \text{treats.Patient}
\]

\[
\text{Consultant} \subseteq \text{Doctor}
\]

\[
\begin{align*}
\text{treats}(x, f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) & \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]

\[
\begin{align*}
Q(x) & \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) & \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) & \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Doctor}(x) \\
Q(x) & \leftarrow \text{Consultant}(x)
\end{align*}
\]
Query Rewriting Technique (basics)

- Example:

\[
\text{Doctor} \sqsubseteq \exists \text{treats.Patient}
\]
\[
\text{Consultant} \sqsubseteq \text{Doctor}
\]

\[
\text{treats}(x, f(x)) \leftarrow \text{Doctor}(x)
\]
\[
\text{Patient}(f(x)) \leftarrow \text{Doctor}(x)
\]
\[
\text{Doctor}(x) \leftarrow \text{Consultant}(x)
\]

\[
Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y)
\]
\[
Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x))
\]
\[
Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x)
\]
\[
Q(x) \leftarrow \text{Doctor}(x)
\]
\[
Q(x) \leftarrow \text{Consultant}(x)
\]
Query Rewriting Technique (basics)

• Example:

\[
\text{Doctor} \subseteq \exists \text{treats.Patient} \\
\text{Consultant} \subseteq \text{Doctor}
\]

\[
treats(x, f(x)) \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) \leftarrow \text{Consultant}(x)
\]

\[
Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) \leftarrow \text{Doctor}(x) \\
Q(x) \leftarrow \text{Consultant}(x)
\]
Query Rewriting Technique (basics)

• Example:

\[
\text{Doctor} \sqsubseteq \exists \text{treats.Patient} \\
\text{Consultant} \sqsubseteq \text{Doctor}
\]

\[
\text{treats}(x, f(x)) \leftarrow \text{Doctor}(x) \\
\text{Patient}(f(x)) \leftarrow \text{Doctor}(x) \\
\text{Doctor}(x) \leftarrow \text{Consultant}(x)
\]

\[
Q(x) \leftarrow \text{treats}(x, y) \land \text{Patient}(y) \\
Q(x) \leftarrow \text{Doctor}(x) \land \text{Patient}(f(x)) \\
Q(x) \leftarrow \text{treats}(x, f(x)) \land \text{Doctor}(x) \\
Q(x) \leftarrow \text{Doctor}(x) \\
Q(x) \leftarrow \text{Consultant}(x)
\]

• 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 $\rightarrow$ SELECT Name FROM Doctor
  - Patient $\rightarrow$ SELECT Name FROM Patient
  - treats $\rightarrow$ 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
1 Names, Prefixes, and Notation

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

Prefix Name Expansion
ntrip http://w3.org/2006/02/02 rdf-syntax
rdfs http://w3.org/2000/01/rdf-schema
owl http://www.w3.org/2002/07/owl
xsd http://www.w3.org/2000/01/xyd

We use notation conventions in the following table:

<table>
<thead>
<tr>
<th>Letters</th>
<th>Meaning</th>
<th>Letters</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>blank node</td>
<td>p</td>
<td>property object expression</td>
</tr>
<tr>
<td>a</td>
<td>individual</td>
<td>a</td>
<td>prefix name</td>
</tr>
<tr>
<td>A</td>
<td>annotation property</td>
<td>P</td>
<td>property object expression</td>
</tr>
<tr>
<td>R</td>
<td>property</td>
<td>x</td>
<td>blank node</td>
</tr>
<tr>
<td>C</td>
<td>class</td>
<td>a</td>
<td>individual</td>
</tr>
<tr>
<td>CN</td>
<td>class name</td>
<td>i</td>
<td>anonymous individual</td>
</tr>
<tr>
<td>D</td>
<td>data range</td>
<td>x</td>
<td>blank node</td>
</tr>
<tr>
<td>DN</td>
<td>data type name</td>
<td>i</td>
<td>anonymous individual</td>
</tr>
<tr>
<td>f</td>
<td>from</td>
<td>x</td>
<td>blank node</td>
</tr>
</tbody>
</table>

*All of the above can have subscripts. **A shorthand for "x**" is "x+" or "x-neg" or "x-integer".

2.2 Properties

Object Property Expressions

<table>
<thead>
<tr>
<th>named object property</th>
<th>OWL</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>named object property</td>
<td>owl:topObjectProperty</td>
<td>owl:topObjectProperty</td>
</tr>
<tr>
<td>inverted object property</td>
<td>owl:inverseObjectProperty</td>
<td>owl:inverseObjectProperty</td>
</tr>
</tbody>
</table>

Data Property Expressions

<table>
<thead>
<tr>
<th>named data property</th>
<th>OWL</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>named data property</td>
<td>owl:topDataProperty</td>
<td>owl:topDataProperty</td>
</tr>
<tr>
<td>inverted data property</td>
<td>owl:inverseDataProperty</td>
<td>owl:inverseDataProperty</td>
</tr>
</tbody>
</table>

2.3 Individuals & Literals

<table>
<thead>
<tr>
<th>named individual</th>
<th>OWL</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>named individual</td>
<td>owl:topIndividual</td>
<td>owl:topIndividual</td>
</tr>
<tr>
<td>literal</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

2.4 Data Ranges

Data Range Expressions

<table>
<thead>
<tr>
<th>named data range</th>
<th>OWL</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>named data range</td>
<td>owl:topDataRange</td>
<td>owl:topDataRange</td>
</tr>
</tbody>
</table>

2.5 Axioms

Class Expression Axioms

<table>
<thead>
<tr>
<th>subclass</th>
<th>equivalent classes</th>
<th>subclass Axiom (C1, C2)</th>
<th>C1</th>
<th>- C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>equivalent classes</td>
<td>owl:equivalentClass</td>
<td>owl:equivalentClass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disjoint classes</td>
<td>DisjointClasses</td>
<td>C1</td>
<td>- C2</td>
<td></td>
</tr>
<tr>
<td>disjoint classes</td>
<td>DisjointClasses</td>
<td>C1</td>
<td>- C2</td>
<td></td>
</tr>
<tr>
<td>disjoint union</td>
<td>DisjointUnion</td>
<td>C1</td>
<td>- C2</td>
<td></td>
</tr>
<tr>
<td>disjoint union</td>
<td>DisjointUnion</td>
<td>C1</td>
<td>- C2</td>
<td></td>
</tr>
</tbody>
</table>

Object Property Axioms

| subproperty | SubObjectPropertyOf | P1 |- | P2 |
|-------------|---------------------|-----|-----|
| property chain | EquivalentObjectProperty | P1 |- | P2 |
| property domain | ObjectPropertyDomain | P1 |- | P2 |
| property range | EquivalentObjectProperty | P1 |- | P2 |
| disjoint properties | DisjointObjectProperties | P1 |- | P2 |
| disjoint properties | DisjointObjectProperties | P1 |- | P2 |
| inverse properties | InverseObjectProperties | P1 |- | P2 |
| inverse properties | InverseObjectProperties | P1 |- | P2 |
# OWL 2 Documentation Roadmap

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

Any questions?

Resources:

• OWL 2 Proposed Recommendation