OWL 2
Web Ontology Language

Some material adapted from presentations by Ian Horrocks and by Feroz Farazi
Introduction

• OWL 2 extends OWL 1 and is backward compatible with it
• The new features of OWL 2 based on real applications, use cases and user experience
• Adopted as a W3C recommendation in December 2009
• All new features were justified by use cases and examples
Features and Rationale

• Syntactic sugar
• New constructs for properties
• Extended datatypes
• Punning
• Extended annotations
• Some innovations
• Minor features
Syntactic Sugar

• OWL 2 adds features that
  – Don’t change expressiveness, semantics, complexity
  – Makes some patterns easier to write
  – Allowing more efficient processing in reasoners

• New features include:
  – DisjointUnion
  – DisjointClasses
  – NegativeObjectPropertyAssertion
  – NegativeDataPropertyAssertion
Syntactic sugar: disJointUnion

• Need for disjointUnion construct
  – A :CarDoor is exclusively either
    • a :FrontDoor,
    • a :RearDoor or
    • a :TrunkDoor
    • and not more than one of them

• In turtle
  :CarDoor a owl:Class;
  owl:disjointUnionOf
    (:FrontDoor
    :RearDoor
    :TrunkDoor) .
Syntactic sugar: disJointUnion

• It’s common for a concept to have more than one decomposition into disjoint union sets
• E.g.: every person is either male or female (but not both) and also either a minor or adult (but not both)

    foaf:Person
    owl:disjointUnionOf (:MalePerson :FemalePerson);
    owl:disjointUnionOf (:Minor :Adult) .
Syntactic sugar: disJointClasses

• It’s common to want to assert that a set of classes are pairwise disjoint
• i.e., that no individual can be an instance of two of the classes in the set

[a owl:allDisjointClasses;
 owl:members (:faculty :staff :students)]
Syntactic sugar: negative assertions

• Asserts that a property doesn’t hold between two instances or between an instance and a literal
• NegativeObjectPropertyAssertion
  – Barack Obama was not born in Kenya
• NegativeDataPropertyAssertion
  – Barack Obama is not 60 years old
•Encoded using a “reification style”
Syntactic sugar: negative assertions

@prefix dbp: <http://dbpedia.org/resource/> .
@prefix dbpo: <http://dbpedia.org/ontology/> .

[a owl:NegativeObjectPropertyAssertion;
 owl:sourceIndividual dbp:Barack_Obama ;
 owl:assertionProperty dbpo:born_in ;

[a owl:NegativeDataPropertyAssertion;
 owl:sourceIndividual dbp:Barack_Obama ;
 owl:assertionProperty dbpo:age ;
 owl:targetIndividual "60" ] .
New property Features

• Self restriction
• Qualified cardinality restriction
• Object properties
• Disjoint properties
• Property chain
• keys
Self restriction

• Classes of objects that are related to themselves by a given property

• For example, the class of processes that regulate themselves

• It is also called local reflexivity

• An example: Auto-regulating processes regulate themselves

• narcissists are people who love themselves
Qualified cardinality restrictions

• Qualifies the instances to be counted
• Six varieties: \{Data | Object\}\{Min | Exact | Max\}Crdinality
• For example,
  – People with exactly three children who are girls
  – People with at least three names
  – Each individual has at most one SSN
Qualified cardinality restrictions

• Done via new properties with domain owl:Restriction, namely \{min\mid max\}\)QualifiedCardinality and onClass

• Example: people with exactly three children who are girls

  [a owl:restriction;
   owl:onProperty :has_child;
   owl:onClass [owl:subClassOf :FemalePerson;
               owl:subClassOf :Minor].
  QualifiedCardinality “3” .
Object properties

- ReflexiveObjectProperty
  - Globally reflexive
  - Everything is part of itself
- IrreflexiveObjectProperty
  - Nothing can be a proper part of itself
- AsymmetricObjectProperty
  - If x is proper part of y, then the opposite does not hold
Disjoint properties

• E.g: you can’t be both the *parent of* and *child of* the same person

• DisjointObjectProperties
  – Deals with object properties
  – Pairwise disjointness can be asserted
  – E.g., connectedTo and contiguousWith

• DisjointDataProperties
  – Deals with data properties
  – Pairwise disjointness can be asserted
  – E.g., startTime and endTime of a surgery
Property chain inclusion

- Properties can be defined as a composition of other properties
- The brother of your parent is your uncle
  
  :uncle owl:propertyChainAxiom (:parent :brother)
Keys

• Individuals can be identified uniquely
• Identification can be done using
  – A data property
  – An object property or
  – A set of properties
• Example
  foaf:Person owl:hasKey (foaf:mbox);
  owl:hasKey (:homePhone :foaf:name).
Extended datatypes

• Extra datatypes
  – Examples: owl:real, owl:rational, xsd:pattern

• Datatype restrictions
  – Range of datatypes
  – For example, adult has an age >= 18
    – DatatypeRestriction(xsd:integer minInclusive 18)

• Datatype definitions
  – New datatypes
    – DatatypeDefinition( :adultAge DatatypeRestriction(xsd:integer minInclusive 18))
Extended datatypes

• Data range combinations
  – Intersection of
    • DataIntersectionOf( xsd:nonNegativeInteger xsd:nonPositiveInteger )
  – Union of
    • DataUnionOf( xsd:string xsd:integer )
  – Complement of data range
    • DataComplementOf( xsd:positiveInteger )
An example

:Teenager rdfs:subClassOf _:x .

_:x rdf:type owl:Restriction ;
  owl:onProperty :hasAge ;
  owl:someValuesFrom _:y .

_:y rdf:type rdfs:Datatype ;
  owl:onDatatype xsd:integer ;
  owl:withRestrictions ( _:z1 _:z2 ) .

_:z1 xsd:minInclusive "13"^^xsd:integer .

_:z2 xsd:maxInclusive "19"^^xsd:integer .
Punning

• An *OWL 1 DL* thing can’t be both a class and an instance
  — E.g., :SnowLeopard can’t be both a subclass of :Feline and an instance of :EndangeredSpecies

• *OWL 2 DL* offers better support for meta-modeling via *punning*
  — A URI denoting an owl thing can have two distinct views, e.g., as a class and as an instance
  — The one intended is determined by its use
  — A *pun* is often defined as a joke that exploits the fact that a word has two different senses or meanings
Punning Restrictions

• Classes and object properties also can have the same name
  – For example, :mother can be both a property and a class of people

• But classes and datatype properties can not have the same name

• Also datatype properties and object properties can not have the same name
Punning Example

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

foaf:Person a owl:Class.
:Woman a owl:Class.
:Parent a owl:Class.

:mother a owl:ObjectProperty;
   rdfs:domain foaf:Person;
   rdfs:range foaf:Person .

:mother a owl:Class;
   owl:intersectionOf (:Woman :Parent).

validate via http://owl.cs.manchester.ac.uk/validator/
Annotations

• In OWL *annotations* comprise information that carries no official meaning
• Some properties in OWL 1 are considered as annotation properties, e.g., owl:comment, rdf:label and rdf:seeAlso
• OWL 1 allowed RDF reification as a way to say things about triples, again w/o official meaning

```[a rdf:Statement;
   rdf:subject :Barack_Obama;
   rdf:predicate dbpo:born_in;
   rdf:object :Kenya;
   :certainty “0.01” ].```
Annotations

• OWL 2 has native support for annotations, including
  – Annotations on owl axioms (i.e., triples)
  – Annotations on entities (e.g., a Class)
  – Annotations on annotations

• The mechanism is again reification
Annotations

:_x rdf:type owl:Axiom ;
owl:subject :Man ;
owl:object :Person ;
:probability "0.99"^^xsd:integer;
rdfs:label "Every man is a person." .
Inverse object properties

— some object property can be inverse of another property
— For example, partOf and hasPart
— ObjectInverseOf( :partOf ): this expression represents the inverse property of :part of
— This makes writing ontologies easier by avoiding the need to name an inverse
OWL Sub-languages

• OWL 1 had sub-languages: OWL FULL, OWL DL and OWL Lite
• OWL FULL is undecidable
• OWL DL is worst case highly intractable
• Even OWL Lite turned out to be not very tractable (EXPTIME-complete)
• OWL 2 introduced three sub-languages, called Profiles, designed for different use cases
OWL 2 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 Profiles

• Profiles considered
  – Useful computational properties, e.g., reasoning complexity
  – Implementation possibilities, e.g., using RDBs

• There are three profiles
  – OWL 2 EL
  – OWL 2 QL
  – OWL 2 RL
OWL 2 EL

• A (near maximal) fragment of OWL 2 such that
  – Satisfiability checking is in PTime (PTime-Complete)
  – Data complexity of query answering is PTime-Complete

• Based on EL family of description logics
  – Existential (someValuesFrom) + conjunction

• It does not allow disjunction and universal restrictions

• Saturation is an efficient reasoning technique

• It can capture the expressive power used by many large-scale ontologies, e.g., SNOMED CT
Basic Saturation-based Technique

• 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:
  \[
  \frac{A \subseteq B \quad B \subseteq C}{A \subseteq C} \quad \frac{A \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D}{A \subseteq D}
  \]
  \[
  \frac{A \subseteq \exists R.B \quad B \subseteq C \quad \exists R.C \subseteq D}{A \subseteq D}
  \]

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

Example: infer that a heart transplant is a kind of organ transplant

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

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

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

Example:

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

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

Heart ⊆ Organ
Saturation-based Technique (basics)

Example:

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

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

Example:

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

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

Example:

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

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

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

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

\[ \text{OrganTransplant} \subseteq \exists \text{site.Organ} \]

\[ \exists \text{site.Organ} \subseteq \text{SO} \]

\[ \text{Transplant} \sqcap \text{SO} \subseteq \text{OrganTransplant} \]
Saturation-based Technique (basics)

Example:

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

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

Example:

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

Example:

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

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

Example:

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

Example:

$$\text{OrganTransplant} \equiv \text{Transplant} \cap \exists \text{site.Organ}$$

$$\text{HeartTransplant} \equiv \text{Transplant} \cap \exists \text{site.Heart}$$

$$\text{Heart} \sqsubseteq \text{Organ}$$

$$\begin{align*}
A & \sqsubseteq \exists R.B \\
B & \sqsubseteq C \\
\exists R.C & \sqsubseteq D \\
\hline \\
A & \sqsubseteq D
\end{align*}$$

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

Example:

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

\[
\begin{align*}
A & \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D \\
\hline
A & \subseteq D \\
\end{align*}
\]

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

\[
\begin{align*}
\text{HeartTransplant} & \subseteq \text{SO} \\
\end{align*}
\]
Saturation-based Technique (basics)

Example:

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

\[
A \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D \\
A \subseteq D
\]

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

\[
\begin{align*}
\text{HeartTransplant} & \subseteq \text{SO} \\
\text{HeartTransplant} & \subseteq \text{OrganTransplant}
\end{align*}
\]
## Saturation-based Technique

Performance with large bio-medical ontologies

<table>
<thead>
<tr>
<th>Concepts:</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>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

• The QL acronym reflects its relation to the standard relational Query Language
• It does not allow existential and universal restrictions to a class expression or a data range
• These restrictions
  — enable a tight integration with RDBMSs,
  — reasoners can be implemented on top of standard relational databases
• Can answer complex queries (in particular, unions of conjunctive queries) over the instance level (ABox) of the DL knowledge base
OWL 2 QL

We 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 $O$ and query $Q$, use $O$ to rewrite $Q$ as $Q^0$ such that, for any set of ground facts $A$:
  \[ \text{ans}(Q, O, A) = \text{ans}(Q^0, ;, A) \]

- Resolution based query rewriting
  - **Clausify** ontology axioms
  - **Saturate** (clausified) ontology and query using resolution
  - **Prune** redundant query clauses
Query Rewriting Technique (basics)

• Example:

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

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

• Example:

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

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

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

• Example:

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

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

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

• Example:

Doctor \sqsubseteq \exists \text{treats.Patient}
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))
\end{align*}
Query Rewriting Technique (basics)

• Example:

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

• Example:

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

- Example:

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

• Example:

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

• Example:

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

• Example:

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

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

- Example:

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

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

• Example:

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

• Example:

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

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

• For DL-Lite, result is a union of conjunctive queries (UCQ)
Query Rewriting Technique (basics)

• Data can be stored/kept in **RDBMS**

• Relationship between ontology and DB defined by **mappings**, e.g.:

  ```plaintext
  Doctor  ⟷ SELECT Name FROM Doctor
  Patient ⟷ SELECT Name FROM Patient
  treats  ⟷ SELECT DName, PName FROM Treats
  ```

• UCQ translated into **SQL query**:

  ```sql
  SELECT Name FROM Doctor UNION
  SELECT DName FROM Treats, Patient WHERE PName=Name
  ```
OWL 2 RL

• The RL acronym reflects its relation to Rule Languages

• OWL 2 RL is designed to accommodate
  – OWL 2 applications that can trade the full expressivity of the language for efficiency
  – RDF(S) applications that need some added expressivity from OWL 2

• Not allowed: existential quantification to a class, union and disjoint union to class expressions

• These restrictions allow OWL 2 RL to be implemented using rule-based technologies such as rule extended DBMSs, Jess, Prolog, etc.
Profiles

Profile selection depends on

– Expressiveness required by the application
– Priority given to reasoning on classes or data
– Size of the datasets
# Key OWL 2 Documents

<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>

[hOp://w3.org/TR/2009/WD-owl2-overview-20090421/]
Conclusion

• Most of the new features of OWL 2 in comparing with the initial version of OWL have been discussed
• Rationale behind the inclusion of the new features have also been discussed
• Three profiles – OWL 2 EL, OWL 2 QL and OWL 2 RL, and their necessity have been presented
Thank you!
Questions?