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;
 owlmembers (: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 ;
    owl:targetIndividual  dbp:Kenya] .

[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:Re-striction, namely *{min|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:propertyChainAxion (: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/](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

```
:Man rdfs:subClassOf :Person .
_:x  rdf:type       owl:Axiom ;
    owl:subject    :Man ;
    owl:predicate  rdfs:subClassOf ;
    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:

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

- 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 (basics)

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

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$

$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$

Saturation-based Technique (basics)

Example:

$$\begin{aligned}
\mathsf{OrganTransplant} &\equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ} \\
\mathsf{HeartTransplant} &\equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart} \\
\mathsf{Heart} &\sqsubseteq \mathsf{Organ}
\end{aligned}$$

$$\begin{aligned}
\mathsf{OrganTransplant} &\sqsubseteq \mathsf{Transplant} \\
\mathsf{OrganTransplant} &\sqsubseteq \exists \mathsf{site}.\mathsf{Organ} \\
\exists \mathsf{site}.\mathsf{Organ} &\sqsubseteq \mathsf{SO} \\
\mathsf{Transplant} \sqcap \mathsf{SO} &\sqsubseteq \mathsf{OrganTransplant} \\
\mathsf{HeartTransplant} &\sqsubseteq \mathsf{Transplant} \\
\mathsf{HeartTransplant} &\sqsubseteq \exists \mathsf{site}.\mathsf{Heart} \\
\exists \mathsf{site}.\mathsf{Heart} &\sqsubseteq \mathsf{SH} \\
\mathsf{Transplant} \sqcap \mathsf{SH} &\sqsubseteq \mathsf{HeartTransplant}
\end{aligned}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

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

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$

$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$

$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$

$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

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

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$

$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

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

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$

$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$

$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$

$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$

$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$

$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$

$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$$

Saturation-based Technique (basics)

Example:

$$\mathsf{OrganTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Organ}$$
$$\mathsf{HeartTransplant} \equiv \mathsf{Transplant} \sqcap \exists \mathsf{site}.\mathsf{Heart}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

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

$$\mathsf{OrganTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{OrganTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Organ}$$
$$\exists \mathsf{site}.\mathsf{Organ} \sqsubseteq \mathsf{SO}$$
$$\mathsf{Transplant} \sqcap \mathsf{SO} \sqsubseteq \mathsf{OrganTransplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{Transplant}$$
$$\mathsf{HeartTransplant} \sqsubseteq \exists \mathsf{site}.\mathsf{Heart}$$
$$\exists \mathsf{site}.\mathsf{Heart} \sqsubseteq \mathsf{SH}$$
$$\mathsf{Transplant} \sqcap \mathsf{SH} \sqsubseteq \mathsf{HeartTransplant}$$
$$\mathsf{Heart} \sqsubseteq \mathsf{Organ}$$

$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{SO}$$
$$\mathsf{HeartTransplant} \sqsubseteq \mathsf{OrganTransplant}$$

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

- 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:

  ans(Q, O, A) = 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:

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

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

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

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

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

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

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

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

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

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

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

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

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

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

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

Query Rewriting Technique (basics)

- Example:

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

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

$$Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$
$$Q(x) \leftarrow \text{Doctor}(x) \wedge \text{Patient}(f(x))$$
$$Q(x) \leftarrow \text{treats}(x, f(x)) \wedge \text{Doctor}(x)$$
$$Q(x) \leftarrow \text{Doctor}(x)$$
$$Q(x) \leftarrow \text{Consultant}(x)$$

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

Query Rewriting Technique (basics)

- Example:

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

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

$$Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$
$$\cancel{Q(x) \leftarrow \text{Doctor}(x) \wedge \text{Patient}(f(x))}$$
$$\cancel{Q(x) \leftarrow \text{treats}(x, f(x)) \wedge \text{Doctor}(x)}$$
$$Q(x) \leftarrow \text{Doctor}(x)$$
$$Q(x) \leftarrow \text{Consultant}(x)$$

Query Rewriting Technique (basics)

- Example:

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

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

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

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

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

- 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

- Expressivenss required by the application
- Priority given to reasoning on classes or data
- Size of the datasets

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.

Key OWL 2 Documents

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.

http://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?