Ontology Engineering
- The basics of OWL

Eva Blomqvist
eva.blomqvist@liu.se
Outline

• Recap from the ontology intro
• Ontology languages and logic
• Main focus: OWL
  – What does the OWL language contain?
  – Common misconceptions
Reminders from ontology intro
Components

• concepts
  - represent a set or class of entities in a domain
    
    *immune response*
  - organized in taxonomies
    (hierarchies based on e.g. *is-a* or *is-part-of*)
    
    *immune response is-a defense response*

• instances
  - often not represented in an ontology
    (instantiated ontology)
Components

• relations

\[ R: C_1 \times C_2 \times \ldots \times C_n \]

*Protein hasName ProteinName*

*Chromosome hasSubcellularLocation Nucleus*
Components

• axioms
  ‘facts that are always true’

*The origin of a protein is always of the type ‘gene coding origin type’*

*Each protein has at least one source.*

*A helix can never be a sheet and vice versa.*
Example

We have a lot of data and want to be able to ask for all research articles
Example

We have a lot of data and want to be able to ask for all research articles.
Ontologies as vocabularies for RDF data

- Concepts (classes) that represent the types of instances in the data
- Properties that can be used as predicates in our RDF triples

Then we can add more... IF we need to

In this course we focus on task-oriented ontologies, i.e. ontologies that serves a specific purpose, fulfils a certain set of requirements
Ontology languages for the Semantic Web
RDF(S): RDF Schema

• RDF gives a data representation format and ways to serialize, but it does not give any special meaning to vocabulary such as “subClassOf” or “range”

• Triple interpretation is an arbitrary binary relation

• RDF Schema extends RDF with a schema vocabulary
  – Classes as types for individuals: rdfs:Class, rdfs:Literal, rdfs:Datatype, rdf:type and rdfs:subClassOf, etc.
  – Property relations: rdf:Property, rdfs:subPropertyOf, rdfs:range, rdfs:domain, etc.
  – Annotations: rdfs:label, rdfs:comment, etc.
RDF/RDF(S) “Liberality”

- No distinction between classes and instances (individuals)
- Properties can themselves have properties
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
What does RDF(S) give us?

• Ability to use simple schema/vocabularies when describing our resources

• Consistent vocabulary use and sharing

• Simple inference, e.g. inheritance in a taxonomy

• But...
  – In some cases too weak to describe resources in sufficient detail
  – Not formally based on any logic
What are Description Logics?

• A family of logic based Knowledge Representation formalisms
  – Descendants of Semantic Networks, Minsky’s frames, and KL-ONE
  – Describe domain in terms of concepts (classes), roles (relationships) and individuals

• Distinguished by
  – Formal semantics (model theoretic)
    • Decidable fragments of FOL
    • Closely related to Propositional Modal & Dynamic Logics
  – Provision of inference services
    • Sound and complete decision procedures for key problems
    • Implemented systems (highly optimized)
DL Semantics

• Model theoretic semantics. An interpretation consists of
  – A domain of discourse (a collection of objects)
  – Functions mapping
    • classes to set of objects
    • properties to sets of pairs of objects
  – Rules describe how to interpret the constructors and tell us when an interpretation is a model.

• In DL, a class description is thus a characterization of the individuals that are members of that class.
OWL

- DL-based language
- OWL2 is the latest version of the standard (https://www.w3.org/TR/owl2-primer/)
- Different language profiles
  - OWL EL
    - Intended for large ontologies with many classes, mainly used for classification tasks
    - Example use: biomedical ontologies
    - Not allowed: negation, disjunction, inverse properties, universal quantification on properties
  - OWL QL
    - Covers most features of UML and ER-models, so is suitable for use with relational data
    - Example use: ontologies used to access relational data
  - OWL RL
    - Reasoning can be implemented as rules
    - Does not allow expressions that assume an anonymous individual
OWL syntaxes

• Abstract syntax
  – Used in the definition of the language
• Manchester syntax
• OWL in RDF
  – RDF/XML presentation
  – Turtle
• ...

## OWL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Example, Turtle syntax</th>
<th>Example, Manchester syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Classes&gt;</code></td>
<td>:Human rdf:type owl:Class</td>
<td>Class: Human</td>
</tr>
<tr>
<td>intersectionOf</td>
<td>owl:intersectionOf ( :Human :Male )</td>
<td>Human and Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>owl:unionOf ( :Male :Female )</td>
<td>Female or Male</td>
</tr>
<tr>
<td>complementOf</td>
<td>owl:complementOf ( :Male )</td>
<td>not Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>owl:oneOf ( :John :Mary )</td>
<td>{John, Mary}</td>
</tr>
</tbody>
</table>
### OWL Individual Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Example, Turtle syntax</th>
<th>Example, Manchester syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Individual&gt;</code></td>
<td>:Mary rdf:type :Human</td>
<td>Individual: Mary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Types: Human</td>
</tr>
<tr>
<td><code>&lt;Fact&gt;</code></td>
<td>:Mary :worksWith :John</td>
<td>Individual: John</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Facts: worksWith Mary</td>
</tr>
<tr>
<td>differentFrom</td>
<td>:Mary owl:differentFrom :John</td>
<td>Individual: Mary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DifferentFrom: John</td>
</tr>
<tr>
<td>sameAs</td>
<td>:Mary owl:sameAs :May</td>
<td>Individual: Mary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SameAs: May</td>
</tr>
</tbody>
</table>
## OWL Class Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Example, Turtle syntax</th>
<th>Example, Manchester syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>:Woman rdfs:subClassOf :Human</td>
<td>Class: Woman SubClassOf: Human</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>:Person owl:equivalentClass :Human</td>
<td>Class: Person EquivalentTo: Human</td>
</tr>
<tr>
<td>disjointClass</td>
<td>[] rdf:type owl:AllDisjointClasses ; owl:members ( :Woman :Man ) .</td>
<td>DisjointClasses: Woman, Man</td>
</tr>
</tbody>
</table>
## OWL Class Constructors (cont.)

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Example, Turtle syntax</th>
<th>Example, Manchester syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>someValuesFrom</td>
<td>owl:onProperty :hasChild ; owl:someValuesFrom :Male</td>
<td>hasChild some Male</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>owl:onProperty :hasChild ; owl:allValuesFrom :Female</td>
<td>hasChild only Female</td>
</tr>
<tr>
<td>minCardinality</td>
<td>owl:minQualifiedCardinality &quot;2&quot;^^xsd:nonNegativeInteger ; owl:onProperty :hasChild</td>
<td>hasChild min 2</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>owl:maxQualifiedCardinality &quot;2&quot;^^xsd:nonNegativeInteger ; owl:onProperty :hasChild</td>
<td>hasChild max 2</td>
</tr>
</tbody>
</table>
## OWL Property Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Example, Turtle syntax</th>
<th>Example, Manchester syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>subPropertyOf</code></td>
<td>:hasSon rdfs:subPropertyOf :hasChild</td>
<td>ObjectProperty: hasSon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SubPropertyOf: hasChild</td>
</tr>
<tr>
<td><code>domain</code></td>
<td>:hasChild rdfs:domain :Parent</td>
<td>ObjectProperty: hasChild</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Domain: Parent</td>
</tr>
<tr>
<td><code>range</code></td>
<td>:hasSon rdfs:range :Man</td>
<td>ObjectProperty: hasSon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Range: Man</td>
</tr>
<tr>
<td><code>symmetric</code></td>
<td>:worksWith rdf:type owl:SymmetricProperty</td>
<td>ObjectProperty: worksWith</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Characteristics: Symmetric</td>
</tr>
<tr>
<td><code>transitive</code></td>
<td>:hasAncestor rdf:type owl:TransitiveProperty</td>
<td>ObjectProperty: hasAncestor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Characteristics: Transitive</td>
</tr>
<tr>
<td><code>inverseOf</code></td>
<td>:hasParent owl:inverseOf :hasChild</td>
<td>ObjectProperty: hasParent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>InverseOf: hasChild</td>
</tr>
</tbody>
</table>
Other useful OWL constructs

• XML namespaces and prefixes
  – Turtle: @prefix : <http://example.com/owl/families/> .
    @prefix owl: <http://www.w3.org/2002/07/owl#> .
  – Manchester: Prefix: : <http://example.com/owl/families/>
    Prefix: owl: <http://www.w3.org/2002/07/owl#>

• Datatype properties and XML schema datatypes
  – Turtle: :John :hasAge 33
  – Manchester: Individual: John
    Facts: hasAge "33"^^xsd:integer

• Property chains and keys
• owl:imports
• owl:Ontology
• Annotation properties
  – rdfs:label, rdfs:comment, ...
Common misconceptions

- Disjointness of primitives
- Properties do not "belong" to classes
- Interpreting domain and range
- And and or
- Quantification
- Closed and open worlds
Disjointness

• By default, primitive classes are not disjoint.
• Unless we explicitly say so, the description (Animal and Vegetable) is not an unsatisfiable class
• Similarly with individuals – the so-called Unique Name Assumption does not hold, and individuals are not considered to be distinct unless explicitly asserted to be so.
Properties

• Unlike frame-based languages, UML and many other common modelling languages in OWL properties do not "belong" to any specific class

• To "connect" a property to a class we can
  – Add domain and range axioms of the property
  – Add restrictions on the class

• But neither is necessary for it to be a valid OWL ontology!
Domain and Range

• Note domain and range are **NOT** interpreted as a constraint as you might expect
• Domain and range assertions allow us to make inferences about individuals
• Example
  :hasChild rdfs:domain :Parent
  :Mary :hasChild :Bob
  – If we haven’t said anything else about Mary or Bob, this is not an error. But we can now **infer** that Mary is a Parent
And/Or and quantification

- The logical connectives *and* and *or* often cause confusion
  - Milk and sugar? Tea or coffee? – think carefully of the meaning when modeling
- Quantification can be contrary to our intuition.
  - Universal quantification over an empty set is true
  - :John may belong to the class :OnlyDaughterParent if he has no child at all and we describe that class as:
    - :OnlyDaughterParent rdf:type owl:Class ;
    - owl:equivalentClass [ rdf:type owl:Restriction ;
    - owl:onProperty :hasChild ;
    - owl:allValuesFrom :Female ] .
- Existential quantification may imply the existence of an individual that we don’t know the name of
Closed and open world assumptions

• The standard semantics of OWL makes an Open World Assumption (OWA)
  – We cannot assume that all information is known about all the individuals in a domain
  – Negation only through contradiction
    • Anything might be true unless it can be proven false

• Closed World Assumption (CWA)
  – Named individuals are the only individuals in the domain
  – Negation as failure
    • If we don't know that x is of type C, then we assume that x is NOT of type C