Semantic Web Ontologies (continued)
Expressing, Querying, Building

CS 431 - April 6, 2005
Carl Lagoze - Cornell University
Acknowledgements for various slides and ideas

- Ian Horrocks (Manchester U.K.)
- Eric Miller (W3C)
- Dieter Fensel (Berlin)
- Volker Haarslev (Montreal)
- Alun Preece (Aberdeen)
Putting the building blocks together

• XML
  - Syntax for document markup
• URIs
  - Universal naming syntax
• Namespaces
  - Concept space naming
• RDF
  - Sentence construction
• RDFs
  - Primitive Vocabulary building
• Description Logic
  - Formal basis for ontology models
• OWL
  - Expression of logic via RDF sentences
Web Ontology Language (OWL)

- W3C Web Ontology Working Group (WebOnt)
- Follow on to DAML, OIL efforts
- W3C Recommendation
- Vocabulary extension of RDF
Species of OWL

• **OWL Lite**
  - Good for classification hierarchies with simple constraints (e.g., thesauri)
  - Reasoning is computational simple and efficient
• **OWL DL**
  - Computationally complete and decidable (computation in finite time)
  - Correspondence to *description logics* (decidable fragment of first-order logic)
• **OWL Full**
  - Maximum expressiveness
  - No computational guarantees (probably never will be)

• Each language is extension of simpler predecessor
Namespaces and OWL

```
<rdf:RDF
    xmlns="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
    xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
    xml:base  ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
    xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2001/XMLSchema#"/>
```
OWL Class Definition

```xml
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>

<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  ...
</owl:Class>
```
Why owl:class vs. rdfs:class

- Rdfs:class is “class of all classes”
- In DL class can not be treated as individuals (undecidable)
- Thus owl:class, which is expressed as rdfs:subclass of rdfs:class
  - No problem for standard rdf processors since an owl:class “is a” rdfs:class

- Note: there are other times you want to treat class of individuals
  - Class drinkable liquids has instances wine, beer, ....
  - Class wine has instances merlot, chardonnay, zinfandel, ...
OWL class building operations

- **disjointWith**
  - No vegetarians are carnivores
- **sameClassAs** (equivalence)
- **Enumerations** (on instances)
  - The Ivy League is Cornell, Harvard, Yale, ....
- **Boolean set semantics** (on classes)
  - Union (logical disjunction)
    - Class `parent` is union of `mother`, `father`
  - Intersection (logical conjunction of class with properties)
    - Class `WhiteWine` is conjunction of things of class `wine` and have property `white`
  - complimentOf (logical negation)
    - Class `vegetarian` is disjunct of class `carnivore`
OWL Properties

Two types
- ObjectProperty - relations between instances of classes
- DatatypeProperty - relates an instance to an rdfs:Literal or XML Schema datatype
(Both rdfs:subClassOf rdf:Property)

<owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema:string" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="activity">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="ActivityArea" />
</owl:ObjectProperty>
OWL property building operations & restrictions

- **Transitive Property**
  - $P(x, y)$ and $P(y, z) \rightarrow P(x, z)$

- **SymmetricProperty**
  - $P(x, y)$ iff $P(y, x)$

- **Functional Property**
  - $P(x, y)$ and $P(x, z) \rightarrow y = z$

- **inverseOf**
  - $P1(x, y)$ iff $P2(y, x)$

- **InverseFunctional Property**
  - $P(y, x)$ and $P(z, x) \rightarrow y = z$

- **Cardinality**
  - Only 0 or 1 in lite and full
OWL DataTypes

• Full use of XML schema data type definitions
• Examples
  - Define a type age that must be a non-negative integer
  - Define a type clothing size that is an enumeration “small” “medium” “large”
OWL Instance Creation

• Create individual objects filling in slot/attribute/property definitions

<Person ref:ID="William Arms">
  <rdfs:label>Bill</rdfs:label>
  <age><xsd:integer rdf:value="57"></age>
  <shoesize><xsd:decimal rdf:value="10.5"></shoesize>
</Person>
## OWL Lite Summary

<table>
<thead>
<tr>
<th>Schema constructs</th>
<th>Equality constructs</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class (i.e. owl:Class)</td>
<td>equivalentClass</td>
<td>imports</td>
</tr>
<tr>
<td>rdf:Property</td>
<td>equivalentProperty</td>
<td>priorVersion</td>
</tr>
<tr>
<td>rdfs:subClassOf</td>
<td>sameIndividualAs</td>
<td>backwardCompatibleWith</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>differentFrom</td>
<td>incompatibleWith</td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>allDifferent</td>
<td></td>
</tr>
<tr>
<td>rdfs:range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Individual</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Property characteristics

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>Property type restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverseOf</td>
<td>allValuesFrom</td>
</tr>
<tr>
<td>TransitiveProperty</td>
<td>someValuesFrom</td>
</tr>
<tr>
<td>FunctionalProperty</td>
<td>Cardinality (0 or 1)</td>
</tr>
<tr>
<td>InverseFunctionalProperty</td>
<td></td>
</tr>
<tr>
<td>SymmetricProperty</td>
<td></td>
</tr>
<tr>
<td>Class intersection</td>
<td>RDF datatyping</td>
</tr>
<tr>
<td>intersectionOf</td>
<td></td>
</tr>
</tbody>
</table>
OWL DL and Full Summary

Class axioms
oneOf
disjointWith

Class expressions
evervalentClass
rdfs:subClassOf
unionOf
intersectionOf
complementOf

Property fillers
hasValue

Arbitrary cardinality
minCardinality
maxCardinality
Cardinality
OWL DL vs. OWL-Full

- Same vocabulary
- OWL DL restrictions
  - Type separation
    - Class can not also be an individual or property
    - Property can not also be an individual or class
  - Separation of ObjectProperties and DatatypeProperties
## Language Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>DTD</th>
<th>XSD</th>
<th>RDF(S)</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounded lists (&quot;X is known to have exactly 5 children&quot;)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Cardinality constraints (Kleene operators)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Class expressions (unionOf, complementOf)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Data types</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Enumerations</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Equivalence (properties, classes, instances)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Formal semantics (model-theoretic &amp; axiomatic)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Inheritance</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Inference (transitivity, inverse)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Qualified constraints (&quot;all children are of type person&quot;)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Reification</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Protégé and RACER – tools for building, manipulating and reasoning over ontologies

• Protégé - [http://protege.stanford.edu/](http://protege.stanford.edu/)
  - Use the 3.x version
  - Multiple plug-ins are available

• Protégé OWL plug-in
  - [http://protege.stanford.edu/plugins/owl/](http://protege.stanford.edu/plugins/owl/)

• Other semantic web related plug-ins
  - [http://protege.cim3.net/cgi-bin/wiki.pl?ProtegePluginsLibraryByTopic#nid349](http://protege.cim3.net/cgi-bin/wiki.pl?ProtegePluginsLibraryByTopic#nid349)

• Racer
  - Description Logic based reasoning engine
  - Server-based
  - Integrates with Protégé-OWL

• Pizza Ontology
Protégé and OWL Concepts

• Classes and sub-classes
  - Disjoint classes (remember the open-world assumption)
  - Multiple inheritance

• Properties
  - Functional, inverse functional, transitive, symmetric
  - Domains and ranges

• Property restrictions
  - Quantifier
    • Existential
    • Universal
      - Closure axioms
        » Remember the open-world assumption
    • Cardinality
  - Covering axioms
    • Remember the open-world assumption
Storing and querying RDF-based models

- **Persistent storage implementations**
    - Relational databases (mysql, postgres, oracle)
  - Kowari - [http://www.kowari.org](http://www.kowari.org)
    - Mapped files
    - Relational databases (mysql, postgres, oracle)

- **Query languages**
  - RDQL (Kowari, Jena)
  - SPARQL
    - W3C working draft
    - [http://www.w3.org/TR/rdf-sparql-query/](http://www.w3.org/TR/rdf-sparql-query/)
RDQL-by-example

• RDF source

• Queries
  - http://webpub.cs.cornell.edu/courses/cs431/2005sp/examples/RDQL/vc-q1