ontology inference and interchange
Ontology

- provide a “shared understanding of a domain for communication across people and systems”
How to put ontologies on the Web?

W3C’s vision: The Semantic Web

Declarative Languages

Logic layer

Schema layer

Data layer

RDF Schema

HTML

XHTML

SMIL

RDF

XML

DC

11/20/2002
Is XML sufficient for semantic annotation?

- XML: user definable and domain specific markup
- XML document is a labeled tree
- constraints on structure via DTD or XML Schema

```xml
<course date="...">
  <title>...</title>
  <teacher>...</teacher>
  <name>...</name>
  <http>...</http>
  <students>...</students>
</course>
```
XML: limitations for semantic markup

XML makes no commitment on:

1. Domain specific ontological vocabulary
2. Ontological modelling primitives

⇒ requires pre-arranged agreement on 1 & 2

Only feasible for closed collaboration
– agents in a small & stable community
– pages on a small & stable intranet

not for sharable Web-resources

11/20/2002
Is RDF(S) sufficient for semantic annotation?

• RDF provides metadata about Web resources

• Object -- Attribute --> Value triples

  pers05  Author-of  ISBN...

• RDF Schema
  • Defines vocabulary for RDF
  • Organized this vocabulary in a typed hierarchy
    • Class, subClassOf, type
    • Property, subPropertyOf
    • Domain, range
Conclusions about RDF(S)

Next step up from plain XML:
– (small) ontological commitment to modeling primitives
– possible to define vocabulary

However:
– no precisely described meaning
– no inference model
Requirements for an Ontology Language

- **Well designed**
  - Intuitive to human users
  - Adequate expressive power

- **Well defined**
  - Clearly specified *syntax* (obviously)
  - Formal *semantics* (equally important)
  - Adequate expressive power

- **Compatible** with existing (web) standards
The three roots of OIL

Frame-based systems
- Epistemological Modeling Primitives

Web languages
- XML- and RDF-based syntax

Description Logics

Formal Semantics & Reasoning Support
## OIL (explained by example)

<table>
<thead>
<tr>
<th>Class Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>class-def animal</code></td>
<td>% animals are a class</td>
</tr>
<tr>
<td><code>class-def plant</code></td>
<td>% plants are a class</td>
</tr>
<tr>
<td><code>subclass-of NOT animal</code></td>
<td>% that is disjoint from animals</td>
</tr>
<tr>
<td><code>class-def tree</code></td>
<td>% trees are a type of plants</td>
</tr>
<tr>
<td><code>subclass-of plant</code></td>
<td>% trees are a type of plants</td>
</tr>
<tr>
<td><code>class-def branch</code></td>
<td>% branches are parts of some tree</td>
</tr>
<tr>
<td><code>slot-constraint is-part-of has-value tree max-cardinality 1</code></td>
<td>% branches are parts of some tree</td>
</tr>
<tr>
<td><code>class-def defined carnivore</code></td>
<td>% carnivores are animals</td>
</tr>
<tr>
<td><code>subclass-of animal</code></td>
<td>% that eat any other animals</td>
</tr>
<tr>
<td><code>slot-constraint eats value-type animal</code></td>
<td>% that eat any other animals</td>
</tr>
<tr>
<td><code>class-def defined herbivore</code></td>
<td>% herbivores are animals</td>
</tr>
<tr>
<td><code>subclass-of animal, NOT carnivore</code></td>
<td>% that are not carnivores, and</td>
</tr>
<tr>
<td><code>slot-constraint eats value-type plant OR (slot-constraint is-part-of has-value plant)</code></td>
<td>% they eat plants or parts of plants</td>
</tr>
</tbody>
</table>
OIL has a Formal semantics

- Defined by mapping to very expressive DL
  - slot-constraint eats has-value meat, fish
    =
    \exists \text{eats:meat} \cap \exists \text{eats:fish}

- Mapping is used to provide reasoning support from a DL system (e.g., FaCT)
### Semantics via translation to $SHIQ$ DL:

<table>
<thead>
<tr>
<th>OIL</th>
<th>Equivalent $SHIQ$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>slot-def</strong> part-of</td>
<td></td>
</tr>
<tr>
<td><strong>subslot-of</strong> structural-relation</td>
<td>% part-of $\subseteq$ structural-relation</td>
</tr>
<tr>
<td><strong>inverse</strong> has-part</td>
<td>% has-part $\not=!$ part-of$\sim$</td>
</tr>
<tr>
<td><strong>properties</strong> transitive</td>
<td>% part-of $\in R_4$</td>
</tr>
<tr>
<td><strong>class-def</strong> defined herbivore</td>
<td></td>
</tr>
<tr>
<td><strong>subclass-of</strong> animal</td>
<td>% herbivore $\equiv$</td>
</tr>
<tr>
<td><strong>slot-constraint</strong> eats</td>
<td>% animal $\sqcap$</td>
</tr>
<tr>
<td><strong>value-type</strong> plant OR</td>
<td></td>
</tr>
<tr>
<td><strong>slot-constraint</strong> part-of</td>
<td>% $\forall$ eats.(plant $\sqcup$</td>
</tr>
<tr>
<td><strong>has-value</strong> plant</td>
<td>% $\exists$ part-of.plant) $\sqcap$</td>
</tr>
<tr>
<td><strong>min-cardinality</strong> 2 vegetable</td>
<td>% $\geq 2$.eats.vegetable</td>
</tr>
<tr>
<td><strong>disjoint</strong> herbivore carnivore</td>
<td>% herbivore $\subseteq \neg$ carnivore</td>
</tr>
</tbody>
</table>
OIL: Restricts Frame Languages

- No defaults
- limited axioms/rules
- only definition of ontology (not individuals)

**Main reasons for this:**
- Reasoning support
- Semantics
OIL: Extends Frame Languages

- Classes can be primitive (nec. conditions)
  - elephant ⇒ animal that has-colour grey
- or defined (nec. and sufficient conditions)
  - vegetarian ⇔ person who eats meat nor fish
- Classes allowed in slot constraints
  - slot-constraint eats has-value meat
    (eats some meat)
  - slot-constraint eats value-type meat
    (eats only meat)
OIL: Extends Frame Languages

• Can use arbitrary class expressions instead of only class names
  – slot-constraint eats value-type NOT (OR meat fish)
• Cardinality constraints can include value-types
  – slot-constraint eats max-cardinality 1 plant
• Supports sub-slot relation
  – daughter-of sub-slot of child-of
• Slot properties
  – transitive (e.g., part-of )
  – symmetrical (e.g., connected-to)
Extending RDF Schema

Goal
- make RDFS useable as ontology language
  - give RDF(S) precise semantics
  - extend RDF(S) with additional modeling primitives
- to facilitate semantically grounded metadata

Procedure
- formulate ontology language as RDF Schema document
  - using existing primitives as much as possible
  - placing additional primitives in the hierarchy of RDFS primitives
OIL as extension to RDFS (1)

• part of the is-a hierarchy of RDFS extension
• ontology language is defined in RDFS
OIL as RDF(S) extension (1/2)

```xml
<rdfs:Class rdf:ID="herbivore">
  <rdf:type rdf:resource="http://www.ontoknowledge.org/#DefinedClass"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
  <rdfs:subClassOf>
    <oil:NOT>
      <oil:hasOperand rdf:resource="#carnivore"/>
    </oil:NOT>
  </rdfs:subClassOf>
</rdfs:Class>
```
OIL as RDF(S) extension (1/2)

```xml
< rdf:Class rdf:ID="herbivore">
  < rdf:type rdf:resource="http://www.ontoknowledge.org/#DefinedClass">
    < rdf:subClassOf rdf:resource="#animal"/>
    < rdf:subClassOf>
      < oil:NOT oil:hasOperand rdf:resource="#carnivore"/>
    </ rdf:subClassOf>
  </ rdf:Class>
</xml>
Using the extension: three levels

1. **OIL modeling primitives**
   - *slot-constraint, subclass-of, value-type,...*
   - RDF-S document which extends RDF-S

2. **a specific OIL ontology**
   - *animal, plant, herbivore, leaf*
   - RDF-S document (using 1)

3. **instances of this ontology**
   - “Mel the giraffe”, “Tux the penguin”
   - RDF expressions (uses 1 & 2)
   - explicit metadata