DAML: Ontology, Services and Rules

Stuart Aitken
Artificial Intelligence Applications Institute
W3C + DAML

- W3C standards
  - XML
  - RDF - Resource Description Framework
  - RDFS - RDF Schema
- Ontology languages
  - DAML-O - ontology
  - DAML-S - services
  - DAML-R - rules
W3C initiatives

• XML
• RDF
  • Extends XML
  • Represents semantics as triples
• RDF Schema
  • Encodes the type hierarchy
RDF

• Identify ‘things’ through URIs, and
describe them in terms of simple
properties and property values
• Triples: subject predicate object
  • http://www.example.org/index.html
  • http://purl.org/dc/elements/1.1/creator
  • http://www.example.org/staffid/85740
• Subjects and objects are viewed as nodes,
predicates as links in a graph
• Predicates are defined - ontology
• rdf:type - objects can have types
  • a defined predicate
RDFS

• RDF Properties: represent relationships between resources
• No way to describe these properties, or relationships between these properties and other resources
• RDFS: specify Classes and the domain and range of properties:
  • author - domain:Document
    - range: Person
RDFS

- rdfs:Resource the class of everything
- rdfs:Class the class of classes
- rdfs:Literal the class of literal values e.g. string and integer
- rdf:Property instance of rdfs:Class
- rdfs:domain instance of rdf:Property
- rdfs:range instance of rdf:Property
- rdfs:subClassOf
- rdfs:subPropertyOf
DAML-O

• A DAML+OIL knowledge-base is a collection of RDF triples
• DAML+OIL prescribes the meaning of triples that use DAML+OIL vocabulary
• Adds 12 classes and 26 properties to RDFS (axiomatised)
DAML-O

- `daml:Class` a class element - refers to a class name (URI) may contain:
  - `rdfs:subClassOf`
  - `daml:disjointWith`
  - boolean combination of class expressions
  - enumeration elements

- **Class expression**
  - class name (URI)
  - enumeration of classes
  - property restriction
  - boolean combination of the above
Property restrictions: qualify a defined class, A, by stating (quant.) property C.
  e.g. RedWine := Wine /\ hasColour.RED

  • `daml:toClass` for all x, if `property(x,y)` holds of an element y, y is in C
  • `daml:hasClass` for some x, `property(x,y)` holds of an element y of C
• **Cardinality constraints**
  • N values of property
  • Max values
  • Min values
  • E.g. Wine has exactly one colour

• **Description Logic reasoners exist for DAML-O**

• **DL is good for defining concepts, computing the subsumption relation, but**

• **Expressivity is intentionally limited.**
### Description Logic: Syntax and Semantics

<table>
<thead>
<tr>
<th>atomic construct</th>
<th>A</th>
<th>A is a subset of the Universal set</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic role</td>
<td>R</td>
<td>R subset U * U</td>
</tr>
<tr>
<td>conjunction</td>
<td>C \ D</td>
<td>intersection of C and D</td>
</tr>
<tr>
<td>disjunction</td>
<td>C ∨ D</td>
<td>set union of C and D</td>
</tr>
<tr>
<td>negation</td>
<td>-C</td>
<td>complement of C (U\C)</td>
</tr>
<tr>
<td>exists restriction</td>
<td>Some R.C</td>
<td>{x</td>
</tr>
<tr>
<td>value restriction</td>
<td>All R.C</td>
<td>{x</td>
</tr>
<tr>
<td>role hierarchy</td>
<td>R [ S]</td>
<td>R subset S</td>
</tr>
</tbody>
</table>
Description Logic: Subsumption

\[ C := \text{Person} \land \text{All eats.Meat} \]
\[ O := \text{Person} \land \text{Some eats.Meat} \]
\[ V := \text{Person} \land \text{All eats.-Meat} \]

Q1. Is \( O \) a subclass of \( C \) ?
Q2. Are \( C \) and \( V \) disjoint ?
Q3. Are \( O \) and \( V \) disjoint ?
Description Logic: Subsumption

Person

Carnivore  Omnivore  Vegetarian

subclass

disjoint
DAML-S

• Semantic mark-up for web services
• Agents should be able to
  • discover,
  • invoke,
  • compose, and
  • monitor web resources.
• Ontology - expressed in DAML-O
  • A Service
    • presents a Service Profile (what is on offer)
    • described by a Service Model (how it is achieved)
    • supports a Service Grounding (implementation details)
DAML-S

- **Service Profile**
  - serviceName; textDescription; contactInformation

- **Actor**
  - name; title; phone, fax….

- ‘Functional’ characteristics of Service Profiles and Service Models
  - input/output (Parameter Description)
  - precondition/effect (Parameter Description)
DAML-S

- Service Model: Process Ontology
  - Atomic, Simple, Composite Process
- Control Construct
  - Sequence, Split, Choice, If-Then-Else
- Data flow/Parameter Bindings
  - There are no variables in the language to allow instances to be equated
  - E.g. item1 is input; item1 is output, but we can only specify the type as input/output
  - Annotation is used:
    sameValues(Process, [ (valueOf Class,Parameter),…])
DAML-S

- Formalisation of the Process Ontology is weak
  - Classes
  - No/few axioms
- Alternative formalisations of the execution semantics exist
  - Narayanan & McIlraith: situation calculus + petri nets
  - Ankolekar, Huch & Sycara: pi calculus/ functional programming
- Declarative Semantics for a CycL Process Ontology may be relevant
- Uses: Verification, Simulation, Composition
At the ‘implementation level’ DAML-S specs will map to WSDL, SOAP…
Introduce Rules to solve the instance identification (variable) problem - this is a general problem with DL

- RuleML
- Grosof & Horrocks ‘Logic Programming + DL’
- Others…