Semantics - 2

Matthew J. Graham
CACR

Methods of Computational Science
Caltech, 2009 April 7

- **Expressivity** is the ability to describe certain aspects of the world
- Concept schemes can be arranged in terms of expressivity, with more expressive ones capable of expressing a wider variety of statements:

ontologies

- An ontology is a formal specification of a conceptualization - it is a data model that represents a set of concepts within a domain and the relationships between those concepts.

- Ontologies generally describe:
  - Individuals: the basic or "ground level" objects
  - Classes: sets, collections, or types of objects
  - Attributes: properties, features, characteristics, or parameters that objects can have and share
  - Relations: ways that objects can be related to one another
  - Events: the changing of attributes or relations

ontology structure

Ontologies typically have two distinct components:

- Names for important concepts in the domain:
  - **Elephant** is a concept whose members are a kind of animal
  - **Herbivore** is a concept whose members are exactly those animals who eat plants or parts of plants
  - **Adult_Elephant** is a concept whose members are exactly those elephants whose age is greater than 20 years

- Background knowledge/constraints on the domain:
  - Adult_Elephants weigh at least 2000kg
  - All Elephants are either African_Elephants or Indian_Elephants
  - No individual can be both a Herbivore and a Carnivore

rdf schema

- W3C standard for describing RDF vocabularies: RDF Schema is the RDF Vocabulary Description Language
- A semantic extension to RDF that provides mechanisms for describing classes of resources and the properties that will be used with them
- Gives special meaning to certain RDF properties and resources
- Provides the means to describe application specific RDF vocabularies

rdfs constructs

- Describing classes:

  -- **rdfs:Class** and **rdfs:subClassOf**

- Describing properties:

  -- **rdfs:domain** and **rdfs:range**

- Others:

  -- **rdfs:subPropertyOf, rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy**

rdfs example

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://www.example.org/biology#">
    <rdfs:Class rdf:ID="organism">
       <rdfs:comment>Class to describe a living organism</rdfs:comment>
       <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource" />
    </rdfs:Class>
    <rdfs:Class rdf:ID="animal">
       <rdfs:comment>Class to describe an animal</rdfs:comment>
       <rdfs:subClassOf rdf:resource="#organism" />
    </rdfs:Class>
    <rdfs:Class rdf:ID="horse">
       <rdfs:subClassOf rdf:resource="#animal"/>
       <rdfs:comment>Class to describe a horse</rdfs:label>
    </rdf:Description>
    <rdf:Property rdf:ID="scientificName">
       <rdfs:comment>The scientific name of an organism</rdfs:comment>
       <rdfs:domain rdf:resource="#organism"/>
       <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
    </rdf:Property> </rdf:RDF>
```

rdf example

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.example.org/biology#">
  <horse rdf:ID="Seabiscuit">
    <scientificName>Equus ferus caballus</scientificName>
  </horse>
</rdf:RDF>
```

rdfs limitations

- RDF and RDFS provide basic capabilities for describing vocabularies that describe resources
- Other capabilities are desirable, for example:
  - Cardinality constraints (e.g. exactly one)
  - Specifying that properties are transitive (e.g. if A is B and B is C then A is C)
  - Specifying inverse properties
  - Specifying the 'local' range and/or cardinality for a property when used with a given class
  - Describing new classes by combining existing classes (using intersections and unions)
  - Negation (using 'not')

web ontology language (owl)

- W3C standard for authoring ontologies
- Based on RDF (OWL semantically extends RDFS)
- Regarded as one of the fundamental technologies underpinning the Semantic Web
- OWL allows descriptions of:

-- relations between classes (e.g. disjointness)
-- cardinality (e.g. "exactly one")
-- characteristics of properties (e.g. symmetry)
-- enumerated classes

- Data is interpreted as:
  - -- a set of **individuals**
  - -- a set of **property assertions** relating the individuals to each other
  - -- a set of **axioms** placing constraints on sets of individuals (**classes**) and the types of relationships allowed between them

- For example, the family ontology:
  - -- "hasMother" is only present between two individuals when "hasParent" is also present
  - -- members of "HasTypeOBlood" are never related via "hasParent" to members of "HasTypeABBlood"
  - -- If Ada "hasMother" Anne and Ada is "HasTypeOBlood" then Anne is not "HasTypeABBlood"

types of owl

- OWL Lite
  - Simplest type meant to support taxonomies with simple constraints, e.g. cardinality is 0 or 1
- OWL-DL
  - Designed for maximum expressiveness with completeness, decidability and practical reasoning algorithms
  - Corresponds to a description logic
  - Certain restrictions on how/where language constructs can be used in order to guarantee decidability (e.g. transitive properties cannot have number restrictions)
- OWL Full
  - No restrictions on how/where language constructs can be used
  - Compatible with RDFS
  - Not decidable and complete reasoning is probably insupportable
- OWL Lite constructs as legal and valid as OWL-DL constructs and OWL-DL constructs are legal and valid as OWL Full constructs

OWL supports six main ways of describing classes:

- **Named** class: Professor
- **Intersection** class: Human $\sqcap$ Female
- **Union** class: JavaProgrammer $\sqcup$ CProgrammer
- **Complement** class: $\neg$ Professor $\sqcap$ Woman
- **Restriction** class
  - -- **Existential**: $\exists$ hasColleague Lecturer
  - -- **Universal**: $\forall$ hasColleague Professor
  - -- **Cardinality**: hasParent = 2
  - -- **Has Value**: hasColleague $\ni$ Matthew
- **Enumerated** class: {George Matthew Ashish Ciro Roy}

owl properties

- OWL has two main categories of properties:
  - **Object** properties: link individuals to individuals
  - **Datatype** properties: link individuals to datatype values
- Object properties can have an inverse, e.g. worksFor and employs
- Properties can have a specified **domain** and **range**
- Certain property characteristics can be specified:
  - **Functional**: for a given individual, the property takes only value, e.g. husband
  - **Inverse functional**: the inverse of the property is functional (c.f. rdb keys)
  - **Symmetric**: if A links to B then it can be inferred that B links to A
  - **Transitive**: if A links to B and B links to C then it can be inferred that A links to C

owl example: sheep

```
<owl:Class rdf:about="http://www.example.org/biology#sheep">
  <rdfs:label>sheep</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://www.example.org/biology#animal"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
    <owl:onProperty rdf:resource="http://www.example.org/biology#eats"/>
      <owl:allValuesFrom>
        <owl:Class rdf:about="http://www.example.org/biology#grass"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl example: grass and plants

```
<owl:Class rdf:about="http://www.example.org/biology#grass">
   <rdfs:label>grass</rdfs:label>
   <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.example.org/biology#plant"/>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class>
   <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="http://www.example.org/biology#plant"/>
      <owl:Restriction>
         <owl:onProperty rdf:resource="http://www.example.org/biology#part_of"/>
         <owl:someValuesFrom>
            <owl:Class rdf:about="http://www.example.org/biology#plant"/>
         </owl:someValuesFrom>
      </owl:Restriction>
```

```
    </owl:unionOf>
    <owl:disjointWith>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.example.org/biology#part_of"/>
            <owl:someValuesFrom>
              <owl:Class rdf:about="http://www.example.org/biology#animal"/>
            </owl:someValuesFrom>
          </owl:Restriction>
          <owl:Class rdf:about="http://www.example.org/biology#animal"/>
        </owl:unionOf>
      </owl:Class>
    </owl:disjointWith>
</owl:Class>
```

owl example: vegetarian

```
<owl:Class rdf:about="http://www.example.org/biology#vegetarian">
  <rdfs:label>vegetarian</rdfs:label>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://www.example.org/biology#animal"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://www.example.org/biology#eats"/>
          <owl:allValuesFrom>
            <owl:Class>
              <owl:complementOf>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="http://www.example.org/biology#part_of"/>
                  <owl:someValuesFrom>
                    <owl:Class rdf:about="http://www.example.org/biology#animal"/>
                  </owl:someValuesFrom>
```

```
                    </owl:Restriction>
                 </owl:complementOf>
              </owl:Class>
           </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
           <owl:onProperty rdf:resource="http://www.example.org/biology#eats"/>
              <owl:allValuesFrom>
                 <owl:Class>
                    <owl:complementOf>
                       <owl:Class rdf:about="http://www.example.org/biology#animal"/>
                    </owl:complementOf>
                 </owl:Class>
              </owl:allValuesFrom>
          </owl:Restriction>
        </owl:intersectionOf>
     </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

inference and reasoning

- Computers provide reasoning services over a knowledge domain where the domain and the knowledge have been formally and rigorously specified and reasoning algorithms have been implemented in a way which that computer can apply.

- Reasoning with OWL-DL: can infer information that is not explicitly represented in an ontology

  -- subsumption testing
  -- equivalent testing
  -- consistency testing
  -- instantiation testing

inference example

- Sheep only eat grass
- Grass is a plant
- Plants and parts of plants are disjoint from animals and parts of animals
- Vegetarians only eat things which are not animals or parts of animals

=> sheep are vegetarians!

- Protege (protege.stanford.edu)
- Jena (jena.sourceforge.net)