Chapter 3
RDF and RDFS Semantics

Introduction
- RDF has a very simple data model
- But it is quite liberal in what you can say
- Semantics can be given using axiomatically
  - relating it to another representation, e.g., first order logic, for which a semantic model exists
  - May result in an executable semantics
- Semantics can be given by RDF Model Theory (MT)

RDF/RDFS “Liberality”
- No distinction between classes and instances (individuals)
  <Species, type, Class>
  <Lion, type, Species>
  <Leo, type, Lion>
- Properties can themselves have properties
  <hasDaughter, subPropertyOf, hasChild>
  <hasDaughter, type, familyProperty>
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
  <type, range, Class>
  <Property, type, Class>
  <type, subPropertyOf, subClassOf>

Semantics and model theories
- Ontology/KR languages aim to model (part of) world
- Terms in language correspond to entities in world
- MT defines relationship between syntax and interpretations
  - Can be many interpretations (models) of one piece of syntax
  - Models supposed to be analogue of (part of) world
    - e.g., elements of model correspond to objects in world
  - Formal relationship between syntax and models
    - structure of models reflect relationships specified in syntax
  - Inference (e.g., subsumption) defined in terms of MT
    - e.g., \( T \models A \Rightarrow B \) iff in every model of \( T \), \( \text{ext}(A) \subseteq \text{ext}(B) \)
Set Based Model Theory

- Many logics (including standard FOL) use a model theory based on Zermelo-Frankel set theory.
- The domain of discourse (i.e., the part of the world being modelled) is represented as a set (often referred as $\Delta$).
- Objects in the world are interpreted as elements of $\Delta$.
  - Classes/concepts (unary predicates) are subsets of $\Delta$.
  - Properties/roles (binary predicates) are subsets of $\Delta \times \Delta$ (i.e., $\Delta^2$).
  - Ternary predicates are subsets of $\Delta^3$, etc.
- The sub-class relationship between classes can be interpreted as set inclusion.
- Doesn’t work for RDF, because in RDF a class (set) can be a member (element) of another class (set).
  - In Z-F set theory, elements of classes are atomic (no structure).

Set Based Model Theory Example

- Formally, the vocabulary is the set of names we use in our model of (part of) the world: $\{\text{Daisy, Cow, Animal, Mary, Person, Z123ABC, Car, drives, …}\}$.
- An interpretation $I$ is a tuple $h \Delta, \phi^I i$:
  - $\Delta$ is the domain (a set).
  - $\phi^I$ is a mapping that maps:
    - Names of objects to elements of $\Delta$.
    - Names of unary predicates (classes/concepts) to subsets of $\Delta$.
    - Names of binary predicates (properties/roles) to subsets of $\Delta \times \Delta$.
    - And so on for higher arity predicates (if any).

RDF Semantics

- RDF has “non-standard” semantics to deal with this.
- Semantics given by RDF Model Theory (MT).
- In RDF MT, an interpretation $I$ of a vocabulary $V$ is:
  - $IR$, a non-empty set of resources (corresponds to $\Delta$).
  - $IS$, a mapping from $V$ into $IR$ (corresponds to $\phi^I$).
  - $IP$, a distinguished subset of $IR$ (the properties).
  - $IEXT$, a mapping from $IP$ into the powerset of $IR \times IR$.
  - $IL$, a mapping from typed literals into $IR$. 

---
Example RDF Simple Interpretation

- RDF imposes semantic conditions on interpretations, e.g.:
  - \( x \) is in \( IP \) iff \( <x, IS(\text{rdf:Property})> \) is in \( IEXT(I(\text{rdf:type})) \)

- All RDF interpretations must satisfy certain axiomatic triples, e.g.:
  - \( \text{rdf:type} \text{ rdf:type rdfs:Property} \)
  - \( \text{rdf:subject} \text{ rdf:type rdfs:Property} \)
  - \( \text{rdf:predicates} \text{ rdf:type rdfs:Property} \)
  - \( \text{rdf:object} \text{ rdf:type rdfs:Property} \)
  - \( \text{rdf:first} \text{ rdf:type rdfs:Property} \)
  - \( \text{rdf:rest} \text{ rdf:type rdfs:Property} \)
  - \( \text{rdf:value} \text{ rdf:type rdfs:Property} \)
  - …

Example RDF Interpretation

- RDF S 59: simply adds semantic conditions and axiomatic triples that give meaning to schema vocabulary
- Class interpretation \( IEXT \) simply induced by \( \text{rdf:type} \), i.e.:
  - \( x \) is in \( IEXT(y) \) if and only if \( <x,y> \) is in \( IEXT(IS(\text{rdf:type})) \)

- Other semantic conditions include:
  - If \( <x,y> \) is in \( IEXT(IS(\text{rdfs:domain})) \) and \( <u,v> \) is in \( IEXT(x) \) then \( u \) is in \( IEXT(y) \)
  - If \( <x,y> \) is in \( IEXT(IS(\text{rdfs:subClassOf})) \) then \( x \) and \( y \) are in \( IC \) and \( IEXT(x) \) is a subset of \( IEXT(y) \)
  - \( IEXT(IS(\text{rdfs:subClassOf})) \) is transitive and reflexive on \( IC \)

- Axiomatic triples include:
  - \( \text{rdf:type} \text{ rdfs:domain rdfs:Resource} \)
  - \( \text{rdfs:domain} \text{ rdfs:domain rdf:Property} \)

RDFS Semantics
RDFS Interpretation Example

- If RDFS graph includes triples
  <Species, type, Class>
  <Lion, type, Species>
  <Leo, type, Lion>
  <Lion, subClassOf, Mammal>
  <Mammal, subClassOf, Animal>

- Interpretation conditions imply existence of triples
  <Lion, subClassOf, Animal>
  <Leo, type, Mammal>
  <Leo, type, Animal>
  ...

RDFS Axioms

- Another way to define the semantics of RDF and RDFS is to give axioms that relate it to a well-understood representation, such as FOL, that has a formal semantics.
- A benefit of this approach is that the axioms may provide the basis of an “executable semantics”
- For a list of FOL axioms (in N3) defining RDFS vocabulary, see
  [http://www.csee.umbc.edu/691s/n3/rdfs-rules.n3](http://www.csee.umbc.edu/691s/n3/rdfs-rules.n3)

RDFS Inference Rules

- ![RDFS Inference Rules](http://example.com/inference-rules)

RDFS Classes

- ![RDFS Classes](http://example.com/classes)

- rdf:Alt rdfs:subClassOf rdfs:Container.
- rdf:Bag rdfs:subClassOf rdfs:Container.
- rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property.
- rdfs:Datatype rdfs:subClassOf rdfs:Class.
- rdf:Seq rdfs:subClassOf rdfs:Container.
- rdf:XMLLiteral rdfs:subClassOf rdfs:Literal; a rdfs:Datatype.
RDFS Properties

rdfs:seeAlso rdfs:domain rdfs:Resource; rdfs:range rdfs:Resource.
rdfs:isDefinedBy rdfs:domain rdfs:Resource; rdfs:range rdfs:Resource;
  rdfs:subPropertyOf rdfs:seeAlso.
rdfs:domain rdfs:domain rdf:Property; rdfs:range rdfs:Class.
rdfs:range rdfs:domain rdf:Property; rdfs:range rdfs:Class.
rdfs:member rdfs:domain rdf:Container; rdfs:range rdfs:Resource.
rdfs:subClassOf rdfs:domain rdfs:Class; rdfs:range rdfs:Class.
rdfs:subPropertyOf rdfs:domain rdf:Property; rdfs:range rdf:Property.
rdf:type rdfs:domain rdfs:Resource; rdfs:range rdfs:Class.

RDFS individuals

rdfs:first a owl:FunctionalProperty.
rdfs:rest a owl:FunctionalProperty
rdf:nil a rdf:List.

Problems with RDFS

- **RDFS too weak** to describe resources in sufficient detail
  - No **localised range and domain** constraints
  - Can't say that the range of hasChild is person when applied to
    persons and elephant when applied to elephants
  - No **existence/cardinality** constraints
  - Can't say that all **instances** of person have a mother that is also
    a person, or that persons have exactly 2 parents
  - No **transitive, inverse or symmetrical** properties
  - Can't say that isPartOf is a transitive property, that hasPart is
    the inverse of isPartOf or that touches is symmetrical
  - ...
- Difficult to provide **reasoning support**
  - No “native” reasoners for non-standard semantics
  - Possible to reason via FO axiomatisation

Conclusions

- **RDF has a very simple data model**
- But it is quite liberal in what you can say
- Semantics can be given using axiomatically
  - relating it to another representation, e.g., first order
    logic, for which a semantic model exists
  - May result in an executable semantics
- Semantics can be given by RDF Model Theory (MT)