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>
```

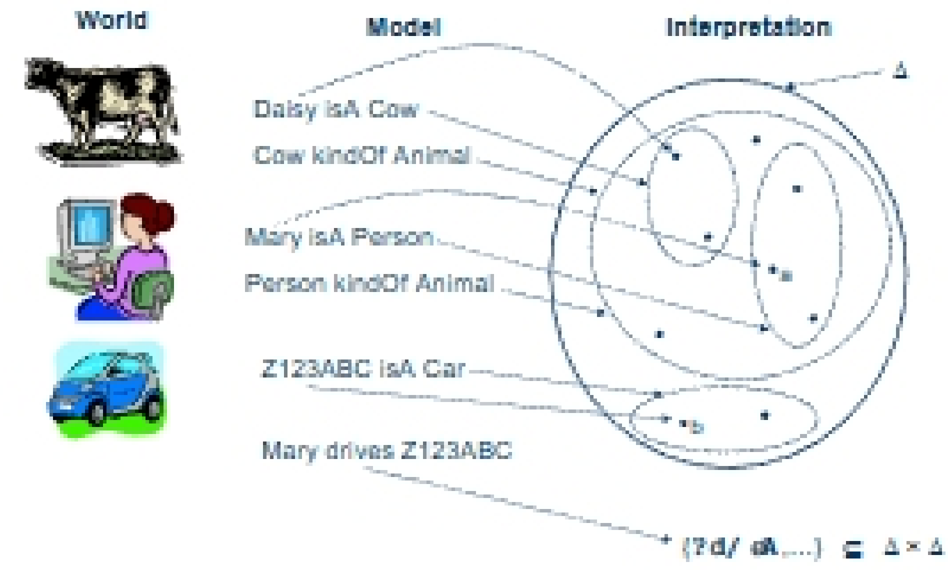
Semantics and model theories

- 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., W ¶ A y B iff in every model of W, ext(A) ⊆ 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 Δ)
- Objects in the world are interpreted as elements of Δ
  - Classes/concepts (unary predicates) are subsets of Δ
  - Properties/roles (binary predicates) are subsets of Δ Δ (i.e., Δ^2)
  - Ternary predicates are subsets of Δ^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



Set Based Model Theory Example

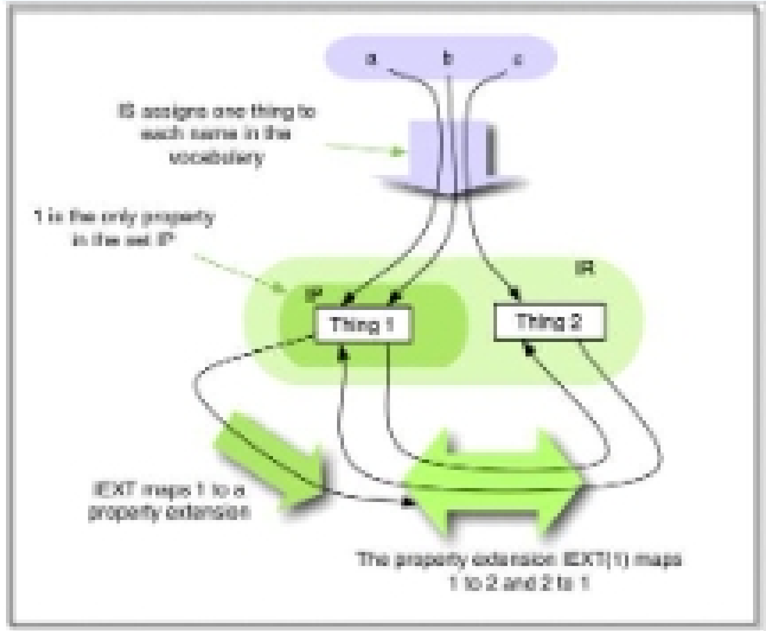
- Formally, the vocabulary is the set of names we use in our model of (part of) the world

  {Daisy, Cow, Animal, Mary, Person, Z123ABC, Car, drives, …}
- An interpretation L is a tuple kΔ, ⨍ 1
  - Δ is the domain (a set)
  - ⨍ is a mapping that maps
    - Names of objects to elements of Δ
    - Names of unary predicates (classes/concepts) to subsets of Δ
    - Names of binary predicates (properties/roles) to subsets of Δ × Δ
    - And so on for higher arity predicates (if any)

RDF Semantics

- RDF has "Non-standard" semantics in order to deal with this
- Semantics given by RDF Model Theory (MT)
- In RDF MT, an interpretation L of a vocabulary V consists of:
  - IR, a non-empty set of resources (corresponds to Δ)
  - IS, a mapping from V into IR (corresponds to ⨍)
  - IP, a distinguished subset of IR (the properties)
    - A vocabulary element v ∈ V is a property iff IS(v) ∈ IP
  - IEXT, a mapping from IP into the powerset of IR × IR
    - I.e., property elements mapped to subsets of IR × IR
  - IL, a mapping from typed literals into IR

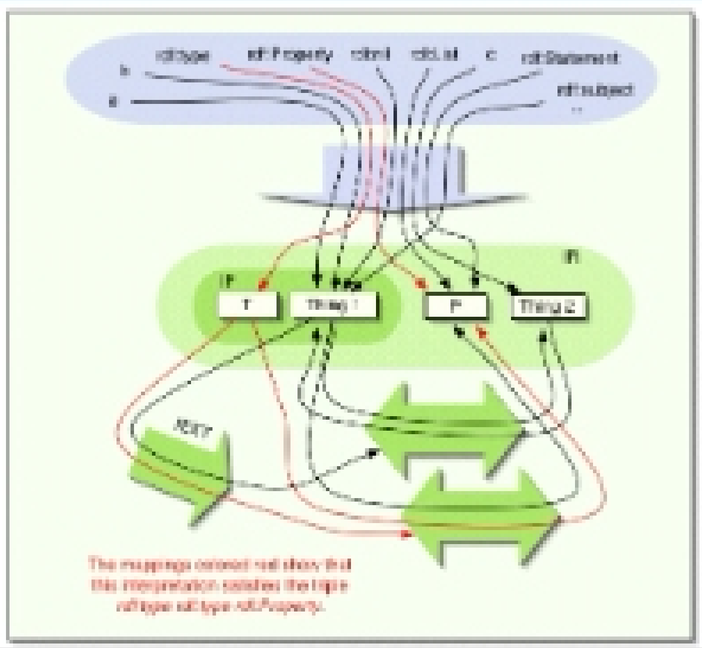
Example RDF Simple Interpretation



RDF Semantic Conditions

- RDF Imposes semantic conditions on interpretations, e.g.:
  - x is in IP if and only if <x, IS(rdf:Property)> is in IEXT(I(rdf:type))
- All RDF interpretations must satisfy certain axiomatic triples, e.g.:
  - rdf:type rdf:type rdf:Property
  - rdf:subject rdf:type rdf:Property
  - rdf:predicate rdf:type rdf:Property
  - rdf:object rdf:type rdf:Property
  - rdf:first rdf:type rdf:Property
  - rdf:rest rdf:type rdf:Property
  - rdf:value rdf:type rdf:Property
  - …

Example RDF Interpretation



RDFS Semantics

- RDFS simply adds semantic conditions and axiomatic triples that give meaning to schema vocabulary
- Class interpretation ICEXT simply induced by rdf:type, i.e.:
  - x is in ICEXT(y) if and only if <x,y> is in IEXT(IS(rdf:type))
- Other semantic conditions include:
  - If <x,y> is in IEXT(IS(rdfs:domain)) and <u,v> is in IEXT(x) then u is in ICEXT(y)
  - If <x,y> is in IEXT(IS(rdfs:subClassOf)) then x and y are in IC and ICEXT(x) is a subset of ICEXT(y)
  - IEXT(IS(rdfs:subClassOf)) is transitive and reflexive on IC
- Axiomatic triples include:
  - rdf:type rdfs:domain rdfs:Resource
  - rdfs:domain rdfs:domain rdf:Property