Chapter 3
RDF Schema

RDF has a very simple data model

RDF Schema (RDFS) enriches the data model, adding vocabulary and associated semantics for:
- Classes and subclasses
- Properties and sub-properties
- Typing of properties

Support for describing simple ontologies

Adds an object-oriented flavor

But with a logic-oriented approach and using "open world" semantics

RDFS is a simple KB Language

Several widely used Knowledge-Base tools can import and export in RDFS, including Stanford’s Protégé KB editor

RDFS Vocabulary

RDFS introduces the following terms, giving each a meaning w.r.t. the rdf data model:

- Terms for classes
  - rdfs:Class
  - rdfs:subClassOf
- Terms for properties
  - rdfs:domain
  - rdfs:range
  - rdfs:subPropertyOf
- Special classes
  - rdfs:Resource
  - rdfs:Literal
  - rdfs:Datatype

- Terms for collections
  - rdfs:member
  - rdfs:Container
  - rdfs:ContainerMembershipProperty
- Special properties
  - rdfs:comment
  - rdfs:seeAlso
  - rdfs:isDefinedBy
  - rdfs:label
Modeling the semantics in logic

- We could represent any triple with a binary predicate, e.g.
  - type(john, human)
  - age(john, 32)
  - subclass(human, animal)
- But traditionally we model a class as a unary predicate
  - human(john)
  - age(john, 32)
  - subclass(human, animal)

Classes and Instances

- We must distinguish between
  - Concrete “things” (individual objects) in the domain: *Discrete Math, Richard Chang*, etc.
  - Sets of individuals sharing properties called classes: lecturers, students, courses etc.
- Individual objects that belong to a class are referred to as instances of that class
- The relationship between instances and classes in RDF is through \texttt{rdf:type}

Classes are Useful

Classes let us impose restrictions on what can be stated in an RDF document using the schema

- As in programming languages
  - E.g. \texttt{A+1}, where \texttt{A} is an array
  - Disallow nonsense from being stated

Preventing nonsensical Statements

- *Discrete Math* is taught by *Calculus*
  - We want courses to be taught by lecturers only
  - Restriction on values of the property “is taught by” (range restriction)
- *Room ITE228* is taught by *Richard Chang*
  - Only courses can be taught
  - This imposes a restriction on the objects to which the property can be applied (domain restriction)
Class Hierarchies

- Classes can be organized in hierarchies
  - A is a **subclass** of B if every instance of A is also an instance of B
  - We also say that B is a **superclass** of A
- A subclass graph needn’t be a tree
  - A class may have multiple superclasses
- In logic:
  - subclass(p, q) ↔ p(x) → q(x)
  - subclass(p, q) ∧ p(x) → q(x)

Domain and Range

- The domain and range properties let us associate classes with a property’s subject and object, e.g.
- Only a course can be taught
  - domain(isTaughtBy, course)
- Only an academic staff member can teach
  - range(isTaughtBy, academicStaffMember)
- Semantics in logic:
  - domain(pred, aclass) ∧ pred(subj, obj) → aclass(subj)
  - range(pred, aclass) ∧ pred(subj, obj) → aclass(obj)

Property Hierarchies

- Hierarchical relationships for properties
  - E.g., “is taught by” is a subproperty of “involves”
  - If a course C is taught by an academic staff member A, then C also involves A
- The converse is not necessarily true
  - E.g., A may be the teacher of the course C, or a TA who grades student homework but doesn’t teach
- Semantics in logic
  - subproperty(p, q) ∧ p(subj, obj) → q(sub, obj)
  - e.g, subproperty(mother, parent), mother(p1, p2) → parent(p1, p2)

RDF Layer vs RDF Schema Layer

- Discrete Math is taught by Richard Chang
- The schema is itself written in a formal language, RDF Schema, that can express its ingredients:
  - subClassOf, Class, Property, subPropertyOf, Resource, etc.
RDF Schema in RDF

- RDFS’s modeling primitives are defined using resources and properties (RDF itself is used!)
- To declare that “lecturer” is a subclass of “academic staff member”
  - Define resources lecturer, academicStaffMember, and subClassOf
  - define property subClassOf
  - Write triple (subClassOf, lecturer, academicStaffMember)
- We use the XML-based syntax of RDF

Core Classes

- rdfs:Resource: class of all resources
- rdfs:Class: class of all classes
- rdfs:Literal: class of all literals (strings)
- rdf:Property: class of all properties
- rdf:Statement: class of all reified statements

Core Properties

- rdf:type: relates a resource to its class
  The resource is declared to be an instance of that class
- rdfs:subClassOf: relates a class to one of its superclasses
  All instances of a class are instances of its superclass
- rdfs:subPropertyOf: relates a property to one of its superproperties

Core Properties

- rdfs:domain: specifies the domain of a property P
  - The class of those resources that may appear as subjects in a triple with predicate P
  - If the domain is not specified, then any resource can be the subject
- rdfs:range: specifies the range of a property P
  - The class of those resources that may appear as values in a triple with predicate P
Examples

```xml
<rdfs:Class rdf:about="#lecturer">
  <rdfs:subClassOf rdf:resource="#staffMember"/>
</rdfs:Class>

<rdfs:Property rdf:ID="phone">
  <rdfs:domain rdf:resource="#staffMember"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdfs:Property>
```

Relationships: Core Classes & Properties

- `rdfs:subClassOf` and `rdfs:subPropertyOf` are transitive, by definition
- `rdfs:Class` is a subclass of `rdfs:Resource`
  - Because every class is a resource
- `rdfs:Resource` is an instance of `rdfs:Class`
  - `rdfs:Resource` is the class of all resources, so it is a class
- Every class is an instance of `rdfs:Class`
  - For the same reason

Subclass Hierarchy of RDFS Primitives

![Diagram showing subclass hierarchy of RDFS primitives]

Instance Relationships of RDFS Primitives

![Diagram showing instance relationships of RDFS primitives]

arrows represent the `rdfs:subClassOf` relation

arrows represent the `rdf:type` relation
**RDF and RDFS Property Instances**

- `rdf:Property`
- `rdfs:domain`
- `rdf:range`
- `rdf:type`
- `rdfs:subClassOf`
- `rdfs:subPropertyOf`

Arrows represent the `rdf:type` relation.

**Reification and Containers**

- `rdf:subject`: relates a reified statement to its subject
- `rdf:predicate`: relates a reified statement to its predicate
- `rdf:object`: relates a reified statement to its object
- `rdf:Bag`: the class of bags
- `rdf:Seq`: the class of sequences
- `rdf:Alt`: the class of alternatives
- `rdfs:Container`: a superclass of all container classes, including the three above

**Utility Properties**

- `rdfs:seeAlso`: relates a resource to another resource that explains it
- `rdfs:isDefinedBy`: a subproperty of `rdfs:seeAlso` that relates a resource to the place where its definition, typically an RDF schema, is found
- `rdfs:comment`: Comments, typically longer text, can be associated with a resource
- `rdfs:label`: A human-friendly label (name) is associated with a resource

**Ex: University Lecturers – Prefix**

```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#">
  ...
</rdf:RDF>
```
Ex: University Lecturers -- Classes

<ex:staffMember rdf:ID="staffMember">  
<ex:comment>The class of staff members</ex:comment>  
</ex:Class>

<ex:academicStaffMember rdf:ID="academicStaffMember">  
<ex:comment>The class of academic staff members</ex:comment>  
<ex:subClassOf rdf:resource="#staffMember"/>  
</ex:Class>

<ex:lecturer rdf:ID="lecturer">  
<ex:comment>The class of lecturers. All lecturers are academic staff members.</ex:comment>  
<ex:subClassOf rdf:resource="#academicStaffMember"/>  
</ex:Class>

<ex:course rdf:ID="course">  
<ex:comment>The class of courses</ex:comment>  
</ex:Class>

Ex: University Lecturers -- Properties

<ex:isTaughtBy rdf:ID="isTaughtBy">  
<ex:comment>Assigns lecturers to courses.</ex:comment>  
<ex:domain rdf:resource="#course"/>  
<ex:range rdf:resource="#lecturer"/>  
</ex:Property>

<ex:teaches rdf:ID="teaches">  
<ex:comment>Assigns courses to lecturers.</ex:comment>  
<ex:domain rdf:resource="#lecturer"/>  
<ex:range rdf:resource="#course"/>  
</ex:Property>

Ex: University Lecturers -- Instances

<uni:lecturer rdf:ID="949318"  
uni:name="Richard Chang"  
uni:title="Associate Professor">  
<uni:teaches rdf:resource="#CIT1111"/>  
<uni:teaches rdf:resource="#CIT3112"/>  
</uni:lecturer>

<uni:lecturer rdf:ID="949352"  
uni:name="Grigoris Antoniou"  
uni:title="Professor">  
<uni:teaches rdf:resource="#CIT1112"/>  
<uni:teaches rdf:resource="#CIT1113"/>  
</uni:lecturer>

<uni:course rdf:ID="CIT1111"  
uni:courseName="Discrete Mathematics">  
<uni:isTaughtBy rdf:resource="#949318"/>  
</uni:course>

<uni:course rdf:ID="CIT1112"  
uni:courseName="Concrete Mathematics">  
<uni:isTaughtBy rdf:resource="#949352"/>  
</uni:course>

Example: A University

<uni:lecturer rdf:ID="lecturer">  
<uni:comment>The class of lecturers. All lecturers are academic staff members.</uni:comment>  
</uni:lecturer>

<uni:academicStaffMember rdf:ID="academicStaffMember">  
</uni:subClassOf rdf:resource="#lecturer"/>  
</uni:Class>
Example: A University

```xml
<rdfs:Class rdf:ID="course">
  <rdfs:comment>The class of courses</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="isTaughtBy">
  <rdfs:comment>
    Inherits its domain ("course") and range ("lecturer")
    from its superproperty "involves"
  </rdfs:comment>
  <rdfs:domain rdf:resource="#course"/>
  <rdfs:range rdf:resource="#lecturer"/>
</rdf:Property>
```

Example: A University

```xml
<rdf:Property rdf:ID="phone">
  <rdfs:comment>
    It is a property of staff members
    and takes literals as values.
  </rdfs:comment>
  <rdfs:domain rdf:resource="#staffMember"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

### RDF and RDFS Namespaces

- The RDF, RDFS and OWL namespaces specify some constraints on the ‘languages’
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#
  - http://www.w3.org/2000/01/rdf-schema#
  - http://www.w3.org/2002/07/owl#
- Strangely, each uses terms from all three to define its own terms
- Don’t be confused: the real semantics of the terms isn’t specified in the namespace files

### RDF Namespace

```xml
<rdf:RDF
xhtml:rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dc="http://purl.org/dc/elements/1.1#"
xmlns:owl:Ontology
rdfs:about="http://www.w3.org/2000/01/rdf-schema#"
dc:title="The RDF Schema vocabulary (RDFS)"/>
```

```xml
<rdf:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
  <rdfs:isDefinedBy rdf:resource="#Resource"/>
  <rdfs:label>Resource</rdfs:label>
  <rdfs:comment>The class resource, everything.</rdfs:comment>
</rdf:Class>
```
This example shows how RDFS terms are used to say something important about the RDF predicate property:

```xml
<rdf:Property
  rdf:ID="predicate"
  rdfs:comment="Identifies the property of a statement in reified form"/>
  <rdfs:domain rdf:resource="#Statement"/>
  <rdfs:range rdf:resource="#Property"/>
</rdf:Property>
```

Define rdf:Resource and rdf:Class as instances of rdfs:Class & rdf:Class as a subclass of rdf:Resource:

```xml
<rdfs:Class rdf:ID="Resource"
  rdfs:comment="The most general class"/>

<rdfs:Class rdf:ID="Class"
  rdfs:comment="The concept of classes. All classes are resources"/>
  <rdfs:subClassOf rdf:resource="#Resource"/>
</rdfs:Class>
```

Define rdf:Resource and rdf:Class as instances of rdfs:Class & rdf:Class as a subclass of rdf:Resource:

```xml
<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Class">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Class</rdfs:label>
  <rdfs:comment>The class of classes.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Resource"/>
</rdfs:Class>

<rdfs:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#subClassOf">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>subClassOf</rdfs:label>
  <rdfs:comment>The subject is a subclass of a class.</rdfs:comment>
  <rdfs:range rdf:resource="#Class"/>
  <rdfs:domain rdf:resource="#Class"/>
</rdfs:Property>
```
### Namespaces vs. Semantics

- Consider **rdfs:subClassOf**
  - The namespace specifies only that it applies to classes and has a class as a value
  - The meaning of being a subclass not specified
- The meaning cannot be expressed in RDF
  - If it could RDF Schema would be unnecessary
- External definition of semantics required
  - Respected by RDF/RDFS processing software

### RDFS vs. OO Models

- In OO models, an object class defines the properties that apply to it
  - Adding a new property means modifying the class
- In RDF, properties are defined globally and aren’t encapsulated as attributes in the class definition
  - One can define new properties w/o changing the class
  - Properties can have properties
    - You can’t narrow the domain and range of properties in a subclass

### Example

```plaintext
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix bio: <http://example.com/biology#> .
bio:Animal a rdfs:Class.
bio:offspring a rdfs:Property;
  rdfs:domain bio:Animal;
  rdfs:range bio:Animal.
:fido a bio:Dog.
:john a bio:Human;
  bio:offspring :fido.
```

There is no way to say that the offspring of humans are humans and the offspring of dogs are dogs.

### Example

**Bio:**

<table>
<thead>
<tr>
<th>Child property</th>
<th>Offspring property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bio:child</td>
<td>Bio:offspring</td>
</tr>
</tbody>
</table>

```plaintext
Bio:child rdfs:subPropertyOf bio:offspring;
  rdfs:domain bio:Human;
  rdfs:range bio:Human.
Bio:puppy rdfs:subPropertyOf bio:offspring;
  rdfs:domain bio:Dog;
  rdfs:range bio:Dog.
```

What do we know after each of the last two triples are asserted?

Suppose we also assert:
- :john bio:puppy :rover
- :john bio:child :fido
Not like types in OO systems

- Classes differ from types in OO systems in how they are used.
  - They are not constraints on well-formedness
- The lack of negation and the open world assumption in RDF+RDFS make it impossible to detect contradictions
  - Can’t say that Dog and Human are disjoint classes
  - Not knowing that there are individuals who are both doesn’t mean it’s not true

No disjunctions or union types

What does this mean?

bio:Cat rdfs:subClassOf bio:Animal.
bio:hasPet a rdfs:Property;
  rdfs:domain bio:Human;
  rdfs:range bio:Dog;
  rdfs:range bio:Cat.

What do we want to say?

- There are many different possibilities
  - Only a dog or a cat can be the object of a hasPet property.
  - Dogs and cats and maybe other animals are possible as pets.
  - Dogs and cats and maybe other things, not necessarily animals, are possible as pets.
  - All dogs and all cats are pets.
  - It is possible for some dogs and for some cats to be pets.
- Not all of these can be said in RDF+RDFS
Classes and individuals are not disjoint

- In OO systems a thing is either a class or object
  - Many KR systems are like this: you are either an instance or a class, not both.
- Not so in RDFS
  bio:Species rdf:type rdfs:Class.
  :fido rdf:type bio:Dog.
- Adds richness to the language but causes problems, too
  - In OWL lite and OWL DL you can’t do this.
  - OWL has it’s own notion of a Class, owl:Class

Inheritance is simple

- No defaults, overriding, shadowing
- What you say about a class is necessarily true of all sub-classes
- A class’ properties are not inherited by its members.
  - Can’t say “Dog’s are normally friendly” or even “All dogs are friendly”
  - The meaning of the Dog class is a set of individuals

Set Based Model Theory Example

Is RDF(S) better than XML?

Q: For a specific application, should I use XML or RDF?
A: It depends…
- XML’s model is
  - a tree, i.e., a strong hierarchy
  - applications may rely on hierarchy position
  - relatively simple syntax and structure
  - not easy to combine trees
- RDF’s model is
  - a loose collections of relations
  - applications may do “database”-like search
  - not easy to recover hierarchy
  - easy to combine relations in one big collection
  - great for the integration of heterogeneous information
Problems with RDFS

- **RDFS too weak** to describe resources in sufficient detail, e.g.:
  - 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
- We need RDF terms providing these and other features.

Conclusions

- RDF is a simple data model based on a graph
  - Independent on any serialization (e.g., XML or N3)
- RDF has a formal semantics providing a dependable basis for reasoning about the meaning of RDF expressions
- RDF has an extensible URI-based vocabulary
- RDF has an XML serialization and can use values represented as XML schema datatypes
- Anyone can make statements about any resource (open world assumption)
- RDFS builds on RDF’s foundation by adding vocabulary with well defined semantics (e.g., Class, subClassOf, etc.)
- OWL addresses some of RDFS’s limitations adding richness (and complexity).