Chapter 3

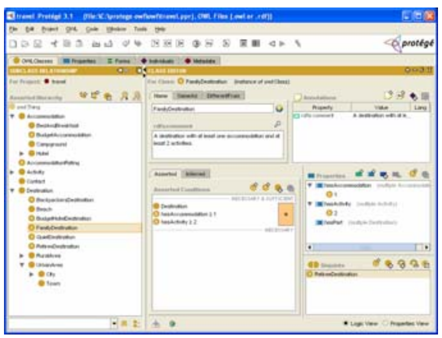
RDF Schema

Introduction

- 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

RDF Schema (RDFS)

- RDFS adds taxonomies for classes & properties
  - subClass and subProperty
- and some metadata.
  - domain and range constraints on properties
- Several widely used KB tools can import and export in RDFS



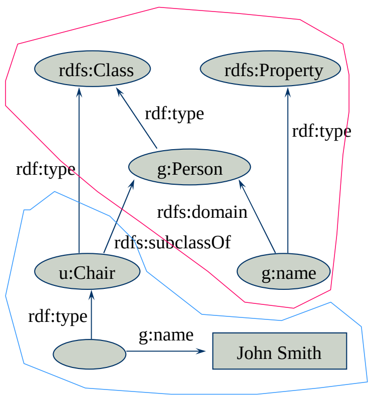
Stanford Protégé KB editor
- Java, open sourced
- extensible, lots of plug-ins
- provides reasoning & server capabilities

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

RDF and RDF Schema




```
<rdfs:Property rdf:ID=“name”>
   <rdfs:domain rdf:resource=“Person”>
</rdfs:Property>

<rdfs:Class rdf:ID=“Chair”>
   <rdfs:subclassOf  rdf:resource=
      “http://schema.org/gen#Person”>
</rdfs:Class>
```

```
<rdf:RDF
     xmlns:g=“http://schema.org/gen”
     xmlns:u=“http://schema.org/univ”>
  <u:Chair rdf:ID=“john”>
    <g:name>John Smith</g:name>
  </u:Chair>
</rdf:RDF>
```

RDFS supports simple inferences

New and Improved! 100% Better than XML!!

- An RDF ontology plus some RDF statements may imply additional RDF statements.
- This is not true of XML.
- Note that this is **part of the data model** and not of the accessing or processing code.

```
@prefix rdfs: <http://www.....>.
@prefix : <genesis.n3>.
     parent rdfs:domain person;
            rdfs:range person.
     mother rdfs:subProperty parent;
            rdfs:domain woman;
            rdfs:range person.
     eve mother cain.
```

```
parent a property.
person a class.
woman subClass person.
mother a property.
eve a person;
    a woman;
    parent cain.
cain a person.
```

N3 example

```
@prefix rdf: <ht                                              #>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix : <#> .
<> rdfs:comment “This is an N3 example”.
:Person a rdfs:Class.
:Woman a rdfs:Class; rdfs:subCl
:eve a :Woman; :age “100”
:sister a rdf:Property;  rdfs:domain
        rdfs:range :Woman.
:eve :sister [a :Woman; :age 98].
:eve :believe {:eve :age “100”}.
[is :spouse of [is :sister of :eve]] :age 9
:eve.:sister.:spouse :age 99.
```

Here’s how you declare a namespace.

This defi   em as referring to “this docu

<> Is an alias for the URI of this document.

“person is a class”. The “a” syntax is sugar for rdf:type

“Woman is a class and a subclass of person”. Note the ; syntax.

“eve is a woman whose age is 100.”

“sister is a property from person to woman”

“eve has woman” an

“eve believes that her age is 100”. The braces introduce a re

“the spouse of the sister of eve is 99”.

“the spouse of the sister of eve is 99”.

Ex: University Lecturers – Prefix

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#

>
```

Ex: University Lecturers -- Classes

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

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

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

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

Ex: University Lecturers -- Properties

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

Ex: University Lecturers -- Instances

```
<uni:lecturer rdf:ID="949318"
    uni:name="David Billington"
    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>
```

RDFS vs. OO Models

- In OO models, an object class defines the properties that apply to it
  - Adding a new property means to modify the class
- In RDF, properties are defined globally and aren't encapsulated as attributes in the class definition
  - One can define new properties without changing the class
  - Properties can have properties
    - :mother rdfs:subPropertyOf :parent; rdf:type :FamilyRelation.
  - You can't narrow the domain and range of properties in a subclass

Example

@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.
bio:Human rdfs:subClassOf bio:Animal.
bio:Dog rdfs:subClassOf 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: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.
:john bio:child :mary.
:fido bio:puppy :rover.

> 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 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:pet a rdfs:Property;
  rdfs:domain bio:Human;
  rdfs:range bio:Dog;
  rdfs:range bio:Cat.

No disjunctions or union types

We have to define the Class explicitly.

bio:Cat rdfs:subClassOf bio:Animal;
  rdfs:subClassOf bio:Pet.
bio:Dog rdfs:subClassOf bio:Pet.
bio:Pet rdfs:subClassOf bio:Animal.

bio:pet a rdfs:Property;
  rdfs:domain bio:Pet;
  rdfs:range bio:Pet;

There's redundancy here. It may or may not be what we want to say Only dogs and cats can be pets?. Are all cats pets? What about feral cats?

Classes and individuals are not disjoint

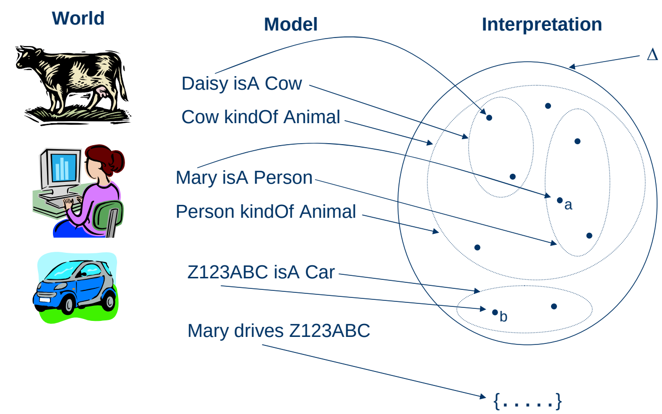
- In OO systems a thing is either a class or object
  - Many KR systems are like this: you are either an instance or a class, not both.
- Not so in RDFS

  bio:Species rdf:type rdfs:Class.
  bio:Dog rdf:type rdfs:Species; rdfs:subClassOf bio:Animal.
  :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 try 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).