Intelligent Internet Technologies

Lectures 15-16.

RDF Schema

Alexandra V. Vitko

KNURE, AI Department, alexandra_vitko@yahoo.com

Why RDFS ?

- Resource description communities require the ability to say certain things about certain kinds of resources.
- The declaration of these properties (attributes) and their corresponding semantics are defined in the context of RDF as an RDF schema.
- A schema defines not only the properties of the resource but may also define the kinds of resources being described.
- RDF schema lets developers define a particular vocabulary for RDF data and specify the kinds of object to which these attributes can be applied.

What is RDFS ?

- The RDF Schema is a collection of RDF resources that can be used to describe properties of other RDF resources.
- The core schema vocabulary is defined in a namespace informally called 'rdfs', and identified by the URI reference: http://www.w3.org/2000/01/rdf-schema#.
- Specification also uses the prefix 'rdf' to refer to the core RDF namespace: http://www.w3.org/1999/02/22-rdf-syntax-ns#.

What is RDFS ?

- RDF Schema
  - Defines vocabulary for RDF
  - Organizes this vocabulary in a typed hierarchy (Class, subClassOf, type, Property, subPropertyOf)
- W3C **Recommendation 10 February 2004**
- Rich, web-based publication format for declaring semantics (XML for exchange)
- Capability to explicitly declare semantic relations between vocabulary terms

RDF Schemas

- Semantic network on the Web
- Nodes are identified by URIs
- rdfs:Class
- rdf:Property
- rdfs:subClassOf
- rdf:type

RDF Classes

- Are groups of Web resources
- Have URIs to identify them
- The special class “rdfs:Literal” consists of all possible RDF string values

Property-centric classes

- In typical OO classes, each class specifies completely what properties it has and what their types are
- In RDF classes, each class specifies completely what properties it has and each property specifies what classes of subjects and objects it relates
- Therefore, new properties can be added to a class without modifying the class

Specifying classes

- To specify a class, create an RDF resource of type rdf:Class, for example:

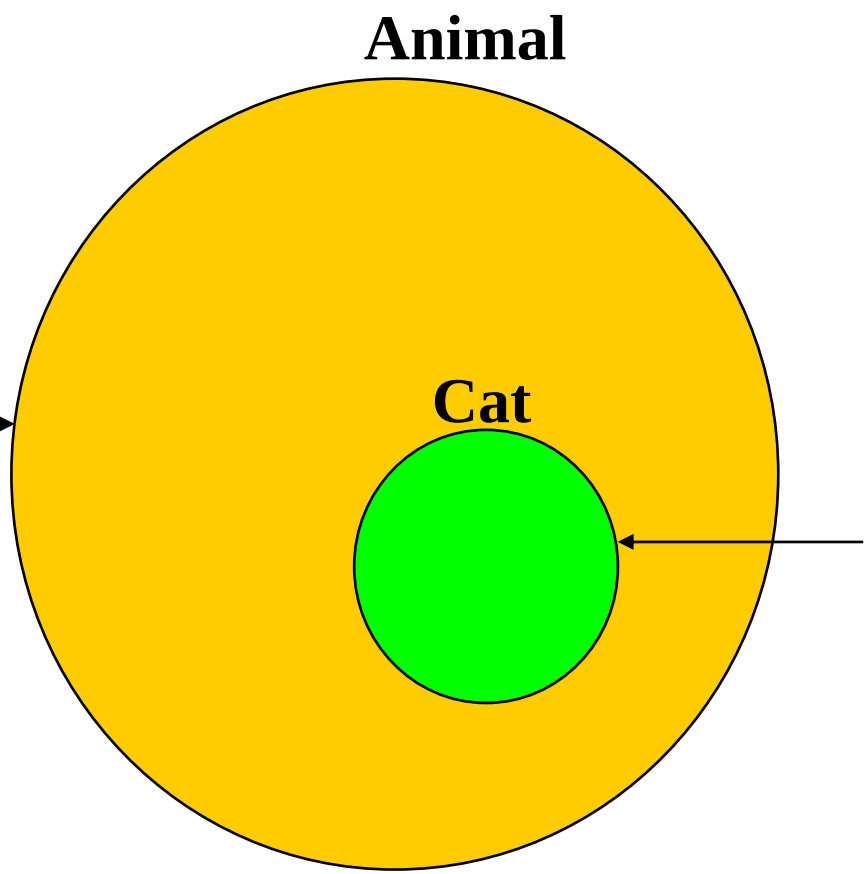
convention

```
<rdf:Description rdf:ID="Animal">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

or in abbreviated syntax

```
<rdfs:Class rdf:ID="Animal" />
```

SubClasses



This represents the set of Animals, i.e., the set of instances of type Animal.

This represents the set of Cats, i.e., the set of instances of type Cat.

Example

```
<?xml version="1.0"?>
<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.ai.kture.kharkov.ua/iti/taxonomy/animals">

<rdf:Description rdf:ID="Animal">
   <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
   <rdfs:subClassOf rdf:resource=" http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>
<rdf:Description rdf:ID="Cat">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Animal"/>
</rdf:Description>
<rdf:Description rdf:ID="Dog">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Animal"/>
</rdf:Description>
</rdf:RDF>
```

Assigns a namespace to the taxonomy!

Since the **Animal** class is defined in the same document we can reference it using a fragment identifier.

Example in Abbreviated Syntax

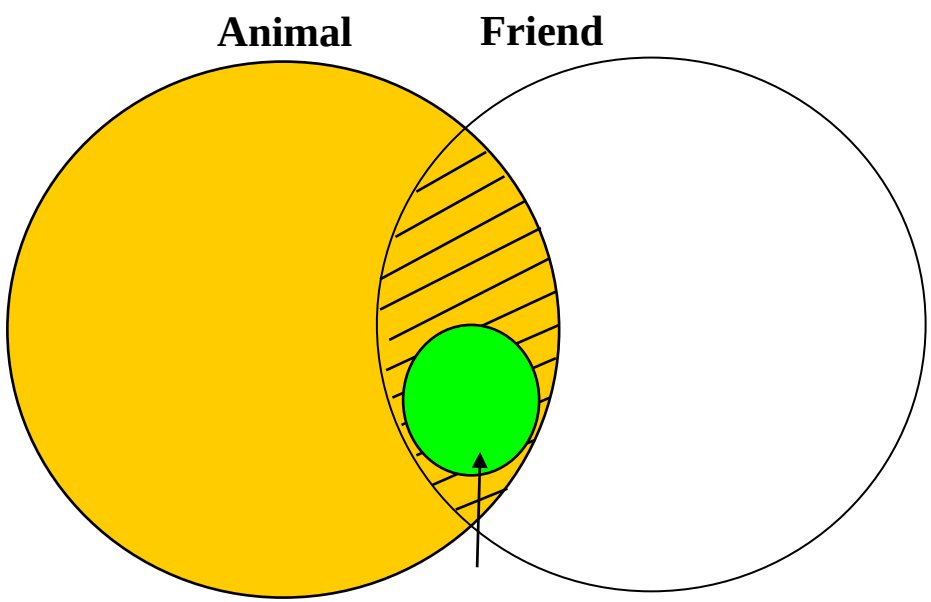
```
<?xml version="1.0"?>
<rdf:RDF    xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://ainet.kture/iti/classes/animals">

<rdfs:Class rdf:ID="Animal" />

<rdfs:Class rdf:ID="Cat">
<rdfs:subClassOf rdf:resource="#Animal"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Dog">
<rdfs:subClassOf rdf:resource="#Animal"/>
</rdfs:Class>

</rdf:RDF>
```

rdfs:subClassOf

- Use this property to indicate a subclass relationship between one class and another class.
- You may specify **zero, one, or multiple rdfs:subClassOf** properties.
  - **Zero**: if you define a class without specifying rdfs:subClassOf then you are implicitly stating that the class is a subClassOf **rdfs:Resource** (the root of all classes).
  - **One**: if you define a class by specifying one rdfs:subClassOf then you are indicating that the class is a subclass of that class.
  - **Multiple**: if you define a class by specifying multiple rdfs:subClassOf properties then you are indicating that the class is a subclass of each of the other classes (**AND**).
    - Example: consider the **Dog** class: suppose that it has two rdfs:subClassOf properties - one that specifies **Animal** and a second that specifies **Friend**. Thus, the two rdfs:subClassOf properties indicate that a Dog is an Animal *and* a Friend. *That is, each instance of Dog is both a n Animal and a Friend*.

Example of multiple rdfs:subClassOf properties

```
<rdfs:Class rdf:ID="Dog">
    <rdfs:subClassOf rdf:resource="#Animal"/>
    <rdfs:subClassOf rdf:resource="http://ainet.kture/iti/taxonomy#Friend"/>
</rdfs:Class>
```



Dog - a Dog is both an Animal and a Friend.

The conjunction (AND) of two subClassOf statements is a subset of the intersection of the classes.

RDF properties

- To specify a property, create an RDF resource of type rdf:Property, for example:

convention

URI rdf

```
<rdf:Description rdf:ID="name">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Description>
```

URI rdfs

or in abbreviated syntax

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

Careful: Class and Property are in different namespaces

- Class is in the rdfs namespace.
- Property is in the rdf namespace.

Literal value

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

A literal type is a simple, untyped string defined in RDFS

Domain and Range of a Property

- "rdfs:domain" specifies the domain of a property (the classes of its subjects); if unknown, anything can be a subject
- "rdfs:range" specifies the range of a property (the single class of its objects); if unknown, anything can be an object

rdfs:domain

- Use this property to indicate the classes that a property will be used with.
- You may specify **zero, one, or multiple rdfs:domain** properties.
  - **Zero**: if you define a property without specifying rdfs:domain then you are providing no information about the class that the property will be used with, i.e., the property can be used with any class.
  - **One**: if you define a property by specifying one rdfs:domain then you are indicating that the property will be used with the class specified by rdfs:domain.
  - **Multiple**: if you define a property by specifying multiple rdfs:domain properties then you are indicating that the property will be used with a class which belongs to every class defined by the rdfs:domain properties (**AND**).

rdfs:range

- Use this property to indicate the type of values that a property will contain.
- You may specify **zero, one, or multiple rdfs:range** properties.
  - **Zero**: if you define a property without specifying rdfs:range then you are providing no information about the type of value that the property will contain.
  - **One**: if you define a property by specifying one rdfs:range then you are indicating that the property will contain a value whose type is that specified by rdfs:range.
  - **Multiple**: if you define a property by specifying multiple rdfs:range properties then you are indicating that the property will contain a value which belongs to every class defined by the rdfs:range properties (**AND**).

subProperties

URI rdfs !!!

The rdfs:subPropertyOf property may be used to state that one property is a subproperty of another.

If a property P is a subproperty of property P', then all pairs of resources which are related by P are also related by P'.

Example rdfs:subPropertyOf

If the **property fatherOf** is a **subproperty** of the broader **property parentOf**, and if Fred is the **father** of John, then it is implied that Fred is also the **parent** of John.

```
<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:Description rdf:ID="parentOf">
      <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
   </rdf:Description>
   <rdf:Description rdf:ID="fatherOf">
      <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
      <rdfs:subPropertyOf rdf:resource="#parentOf"/>
   </rdf:Description>
</rdf:RDF>
or
...
<rdf:Property rdf:ID="parentOf"/>
 <rdf:Property rdf:ID="fatherOf">
      <rdfs:subPropertyOf rdf:resource="#parentOf"/>
 </rdf:Property> …
```

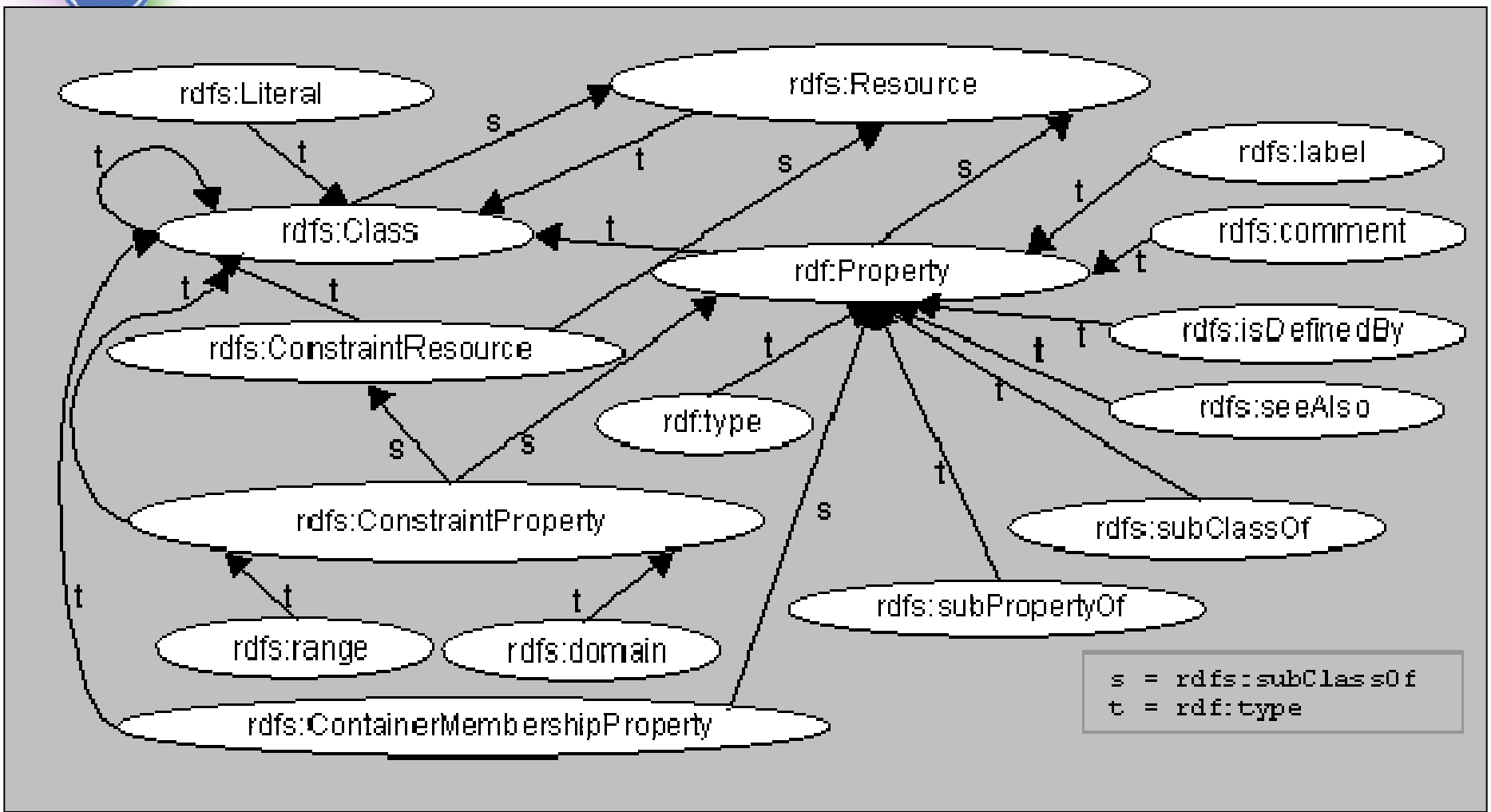
Note that properties are defined separately from classes

- With most Object-Oriented languages when a class is defined the properties (attributes) are simultaneously defined.
  - For example, "I hereby define a Rectangle class, and its attributes are length and width."
- With RDF Schema things are different. You define a class (and indicate its relationships to other classes). Separately, you define properties and then associate them with a class!
  - For the above example you would define the Rectangle class (and indicate that it is a subclass of GeometricObject). Separately, you then define a length property, indicate its range of value, and then indicate that length may be used with the Rectangle class. *(Thus, if you have an untyped Resource with a length property you can infer the Resource is a Rectangle.)* Likewise for the width property.

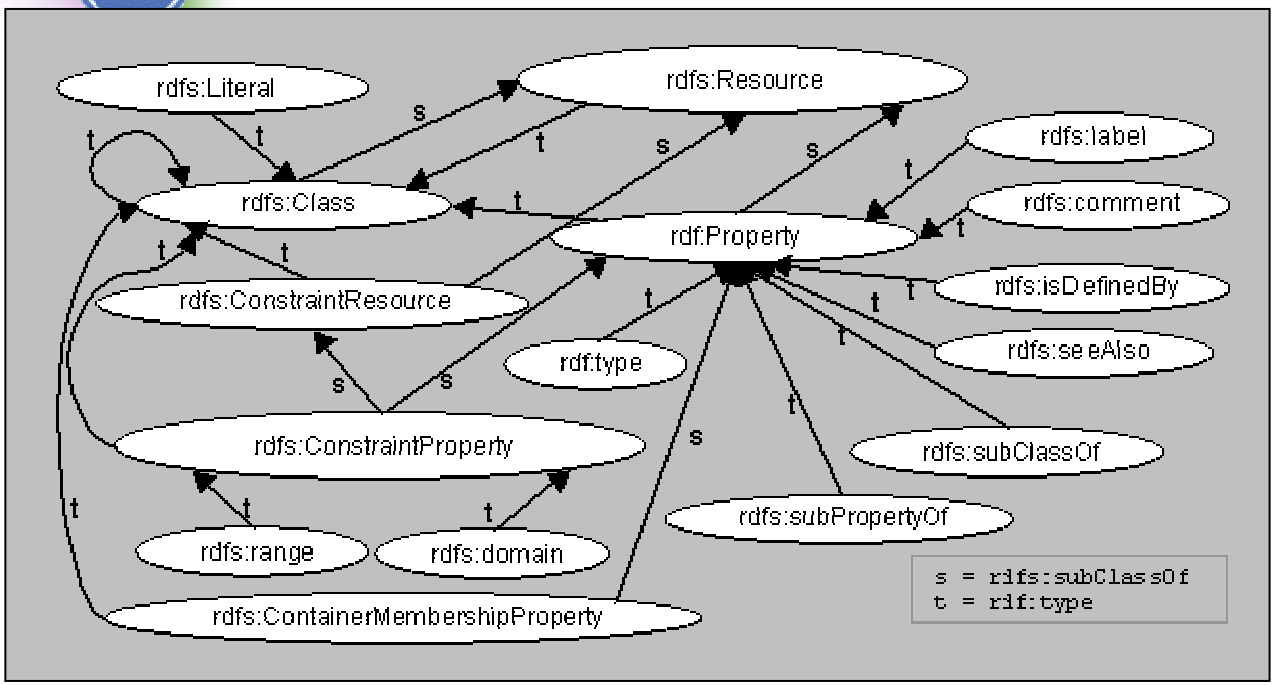
Advantage of separately defining classes and properties

- As we have seen, the RDF Schema approach is to define a class, and then separately define properties and state that they are to be used with the class.
- The advantage of this approach is that anyone, anywhere, anytime can create a property and state that it is usable with the class!

Class Hierarchy of the RDFS (1)



Class Hierarchy of the RDFS (2)



Class hierarchy is shown using a "nodes and arcs" graph representation of the RDF data model. If one class is a subset of another, then there is an rdfs:subClassOf arc from the node representing the first class to the node representing the second.

If a resource is an instance of a class, then there is an rdf:type arc from the resource to the node representing the class.

Useful classes (1)

- "rdfs:Resource" is the class of all resources
- "rdfs:Literal" is the class of all strings
- "rdfs:Class" is the class of all classes
- "rdf:Property" is the class of all properties
- "rdf:Statement" is the class of all asserted RDF statements

Useful classes (2)

- "rdfs:Container" is the superclass of all container classes
- "rdf:Bag", "rdf:Seq", "rdf:Alt" are the classes of Bags, Seqs, and Alts
- (Any other class that is a subclass of "rdfs:Container" can be used in RDF syntax in place of a standard container)
- *Others- see RDF Specification*

Useful properties (1)

- “rdf:type” relates any resource to its class
- “rdfs:subClassOf” relates a subclass to its superclass
- “rdfs:subPropertyOf” relates a subproperty to its superproperty
- “rdfs:seeAlso” relates a resource to another resource explaining it
- “rdfs:isDefinedBy” the definition of the subject resource
- “rdfs:seeAlso” and “rdfs:isDefinedBy” relates a resource to its definition, typically an RDF schema

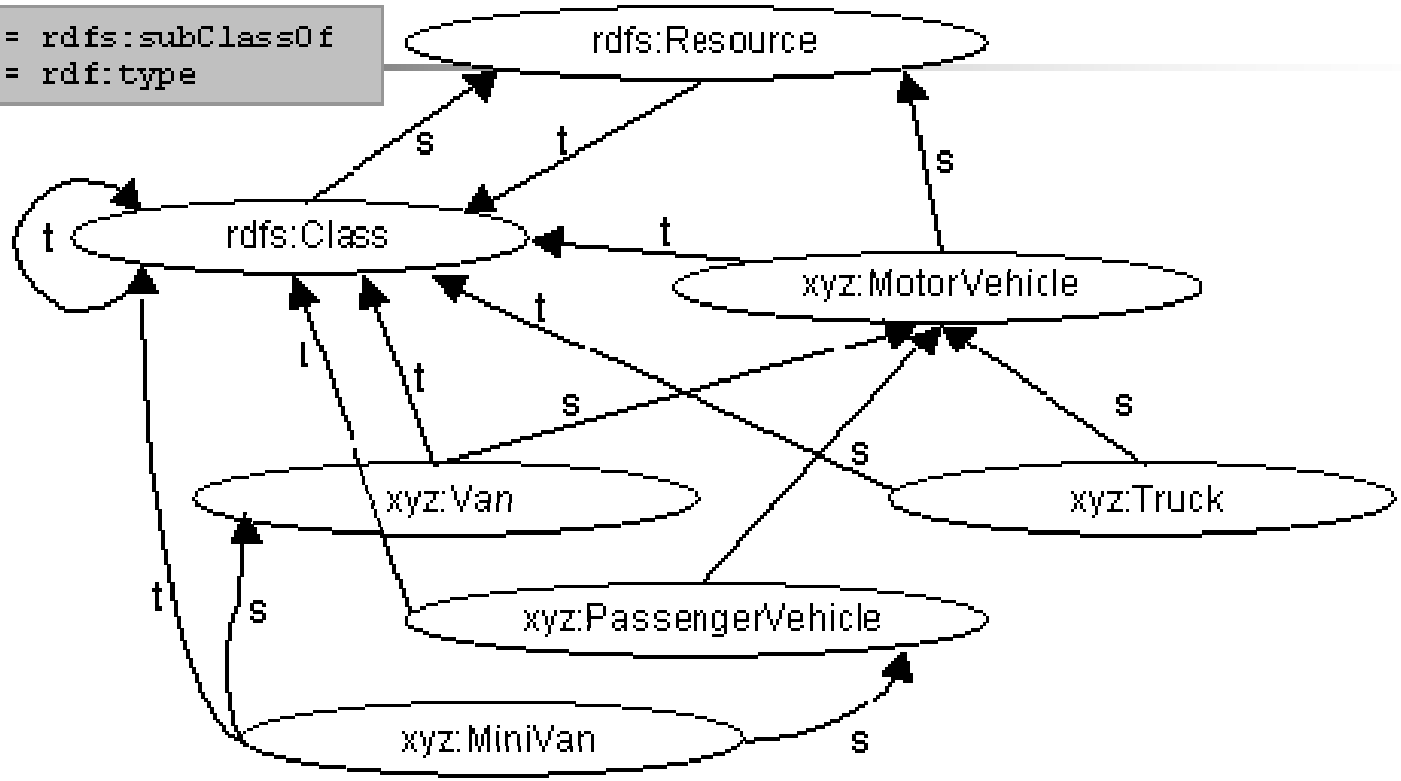
Useful properties (2)

- "rdf:subject" is the property relating a reified statement to its subject (resource)
- "rdf:predicate" is the property relating a reified statement to its predicate (property)
- "rdf:object" is the property relating a reified statement to its object (value)

Useful properties (3)

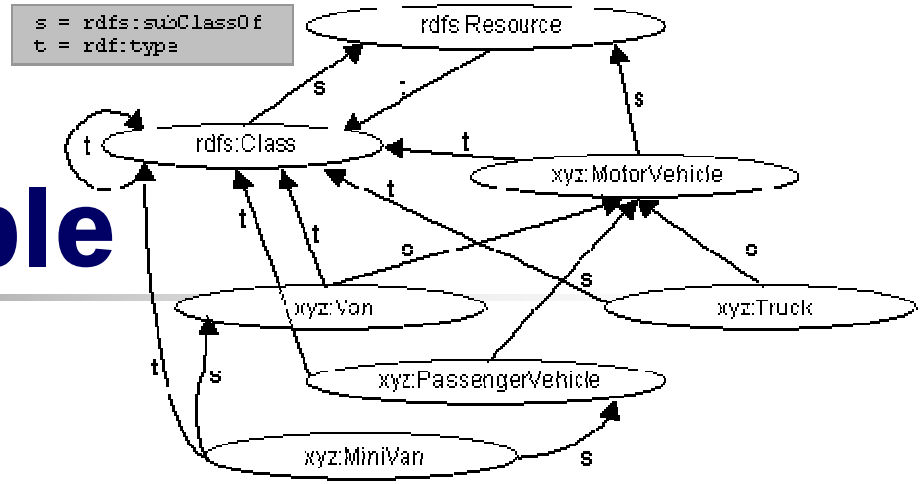
- "rdfs:label" specifies a human-readable name for this Class, Property, or whatever
- "rdfs:comment" specifies human-readable documentation
- Multiple values are useful for specifying multiple languages

RDFS Example



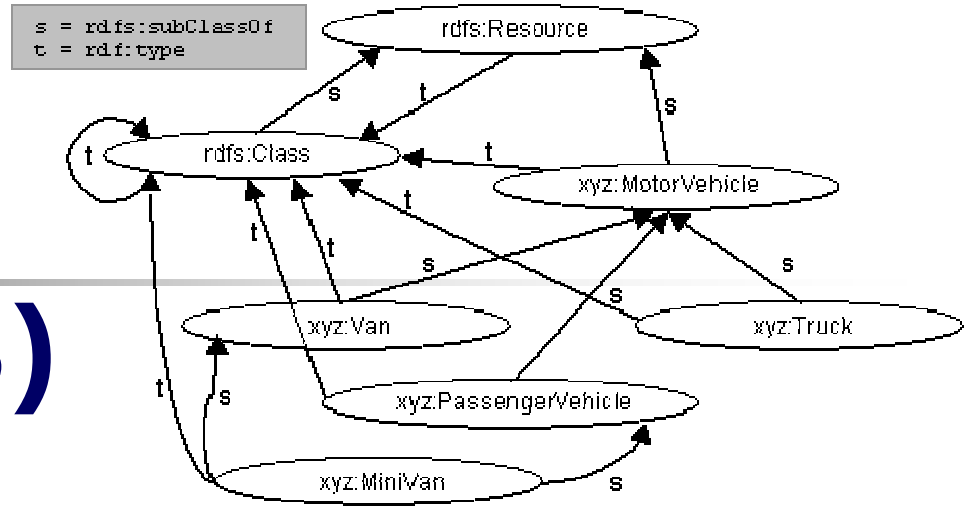
Example expresses the following class hierarchy. We first define a **class** **MotorVehicle**. We then define three **subclasses** of **MotorVehicle**, namely **PassengerVehicle**, **Truck** and **Van**. We then define a **class** **Minivan** which is a **subclass** of both **Van** and **PassengerVehicle**.

RDFS Example (2)



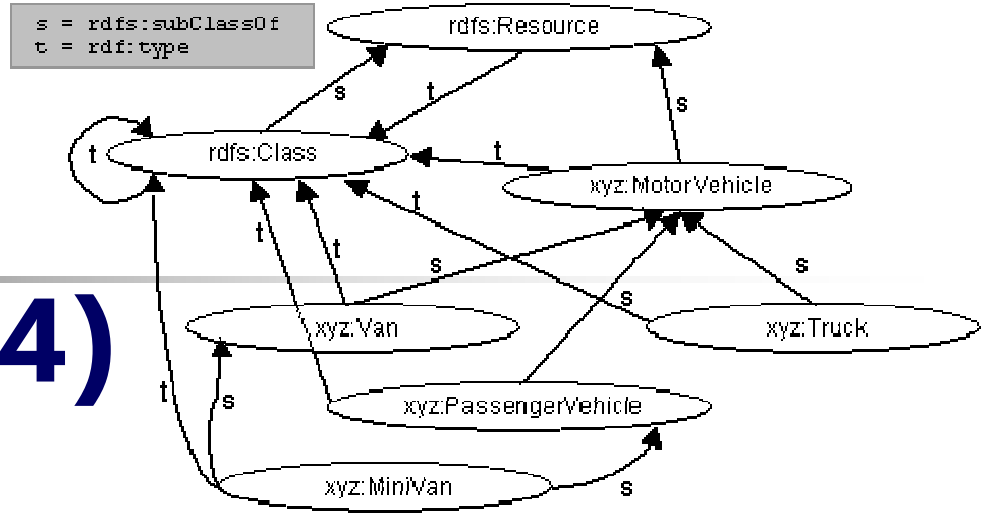

```
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <!-- Note: this RDF schema would typically be used in RDF instance data by
       referencing it with an XML namespace declaration, for example
       xmlns:xyz="http://www.w3.org/2000/03/example/vehicles#".  This allows us to
       use abbreviations such as xyz:MotorVehicle to refer unambiguously to the RDF
       class 'MotorVehicle'. -->

   <rdf:Description ID="MotorVehicle">
      <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf
                  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
   </rdf:Description>
...
```

RDFS Example (3)




```
…

    <rdf:Description ID="PassengerVehicle">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
    </rdf:Description>

    <rdf:Description ID="Truck">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
    </rdf:Description>

…
```

RDFS Example (4)




```
…

    <rdf:Description ID="Van">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
    </rdf:Description>

    <rdf:Description ID="MiniVan">
        <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Van"/>
        <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
    </rdf:Description>

</rdf:RDF>
```

Dublin Core (1)

- A set of fifteen basic properties for describing generalised Web resources
- The “obvious” mapping of Dublin Core properties into RDF properties has not yet been approved by the Dublin Core initiative, but is generally a good example

Dublin Core (2)

- “dc:title”: the name given to the resource
- “dc:creator”: the person or organisation primarily responsible for the resource
- “dc:subject”: what the resource is about
- “dc:description”: a description of the content

Dublin Core (3)

- "dc:publisher": the person or organisation responsible for making the resource available
- "dc:contributor": someone who has provided content to the resource other than the creator
- "dc:date": date of creation or publication

Dublin Core (4)

- "dc:type": type of resource, such as home page, technical report, novel, photograph...
- "dc:format": data format of the resource
- "dc:identifier": URL, ISBN number, ...
- "dc:source": another resource that this resource is derived from

Dublin Core (5)

- "dc:language": the language of the content
- "dc:relation": another resource and its relationship to this one
- "dc:coverage": the portion of time or space described by this resource (atlases, histories, etc.)
- "dc:rights": the intellectual property rights adhering to this resource, or a pointer to them

Dublin Core Example

```
<?xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:dc="http://purl.org/dc/elements/1.0/">
   <rdf:Description rdf:about="http://www.ukoln.ac.uk/metadata/resources/dc/
                                datamodel/WD-dc-rdf/">
     <dc:title> Guidance on expressing the Dublin Core within the Resource
                Description Framework (RDF) </dc:title>
     <dc:creator> Eric Miller </dc:creator>
     <dc:creator> Paul Miller </dc:creator>
     <dc:creator> Dan Brickley </dc:creator>
     <dc:subject> Dublin Core; Resource Description Framework; RDF; eXtensible
                  Markup Language; XML </dc:subject>
     <dc:publisher> Dublin Core Metadata Initiative </dc:publisher>
     <dc:contributor> Dublin Core Data Model Working Group </dc:contributor>
     <dc:date> 1999-07-01 </dc:date>
     <dc:format> text/html </dc:format>
     <dc:language> en </dc:language>
   </rdf:Description>
</rdf:RDF>
```

Read More in

World Wide Web Consortium

http://www.w3.org/TR/rdf-schema/