Semantic Web Languages

Lin Zuoquan

Information Science Department

Peking University

lz@is.pku.edu.cn

http://www.is.pku.edu.cn/~lz/teaching/stm/saswws.html

Courtesy some graphic slides from online

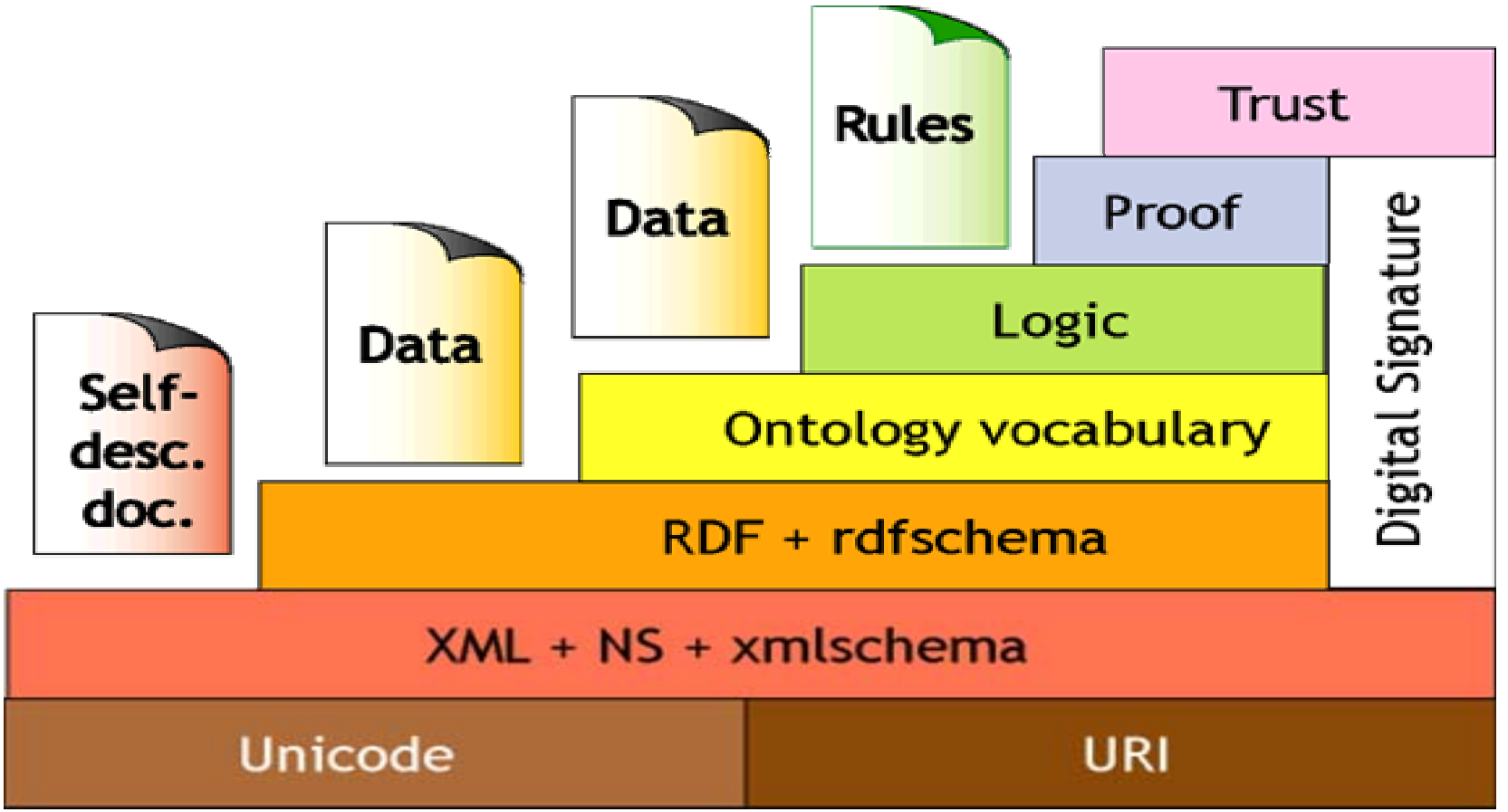
Outline

- RDF
- RDFS
- DAML+OIL
- OWL

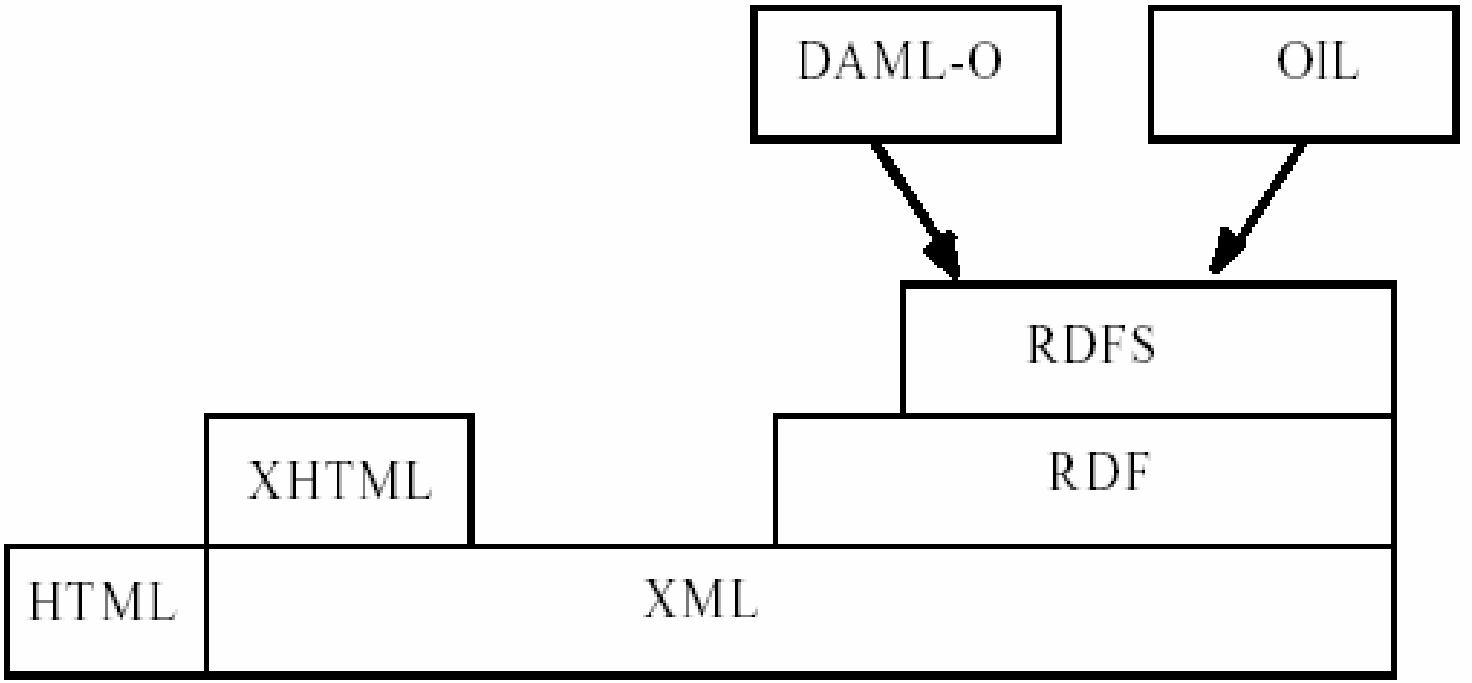
Review

- Metadata
- Ontology

SW Layer


Rules
Trust
Data
Proof
Data
Logic
Self-
desc.
doc.
Ontology vocabulary
Digital Signature
RDF + rdfschema
XML + NS + xmlschema
Unicode
URI

SWL Layer



RDF

- Model
- Representation
- Syntax
- Formal Model and Syntax

RDF

- The Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web.
- RDF defines a simple model for describing relationships among resources in terms of named properties and values.

Model

- The basic RDF data model consists of three object types:
  - Resources
  - Properties
  - Statements

Resourses

- All things being described by RDF expressions are called *resources*.
  - A resource may be an entire Web page; e.g., the HTML document "http://www.w3.org/Overview.html".
  - A resource may be a part of a Web page; e.g. a specific HTML or XML element within the document source.
  - A resource may be a whole collection of pages; e.g. an entire Web site.
  - A resource may be an object that is not directly accessible via the Web; e.g. a printed book.
- Resources are always named by URIs (*resource identifier*) plus optional anchor IDs.
  - Anything can have a URI;
  - The extensibility of URIs allows the introduction of identifiers for any entity imaginable.

Properties

- A *property* is a specific aspect, characteristic, attribute, or relation used to describe a resource.
- Each property has a specific meaning, defines its permitted values, the types of resources it can describe, and its relationship with other properties.
- RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs.
- RDF properties also represent relationships between resources. As such, the RDF data model can therefore resemble anentity-relationship diagram.

Statements

- A specific resource together with a named property plus the value of that property for that resource is an RDF *statement*.
- These three individual parts of a statement are called, respectively, the *subject*, the *predicate*, and the *object*.
- The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e., a resource (specified by a URI) or a simple string or other primitive datatype defined by XML.
  - In RDF terms, a *literal* may have content that is XML markup but is not further evaluated by the RDF processor. There are some syntactic restrictions on how markup in literals may be expressed.

Triple

- Example 1

Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.

This sentence has the following parts:

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

Triple Notes

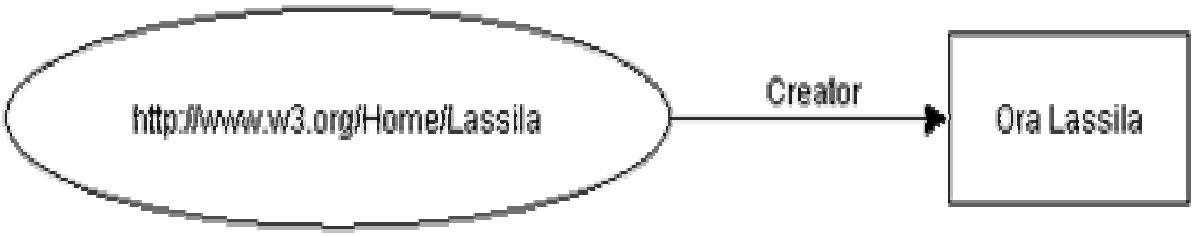
- RDF can be encoded as a set of triples.
- RDF triples have one of two forms:
  - <URI> <URI> <URI>
  - <URI> <URI> <quoted string>
- Triples are also easily mapped into logic
  - <subject> <predicate> <object>
  - <predicate>(<subject>,<object>)
  - With type(<S>,<O>) becoming <O>(<S>)
  - Example:
    - subclass(man,person)
    - sex(man,male)
    - domain(sex,animal)
    - man(adam)
    - age(adam,100)
- Triples are easily stored and managed in a DBMS

Representation

- There are three basic RDF representations of the triples <*subject*> <*predicate*> <*object*> :
  - Graph
  - N3
  - XML

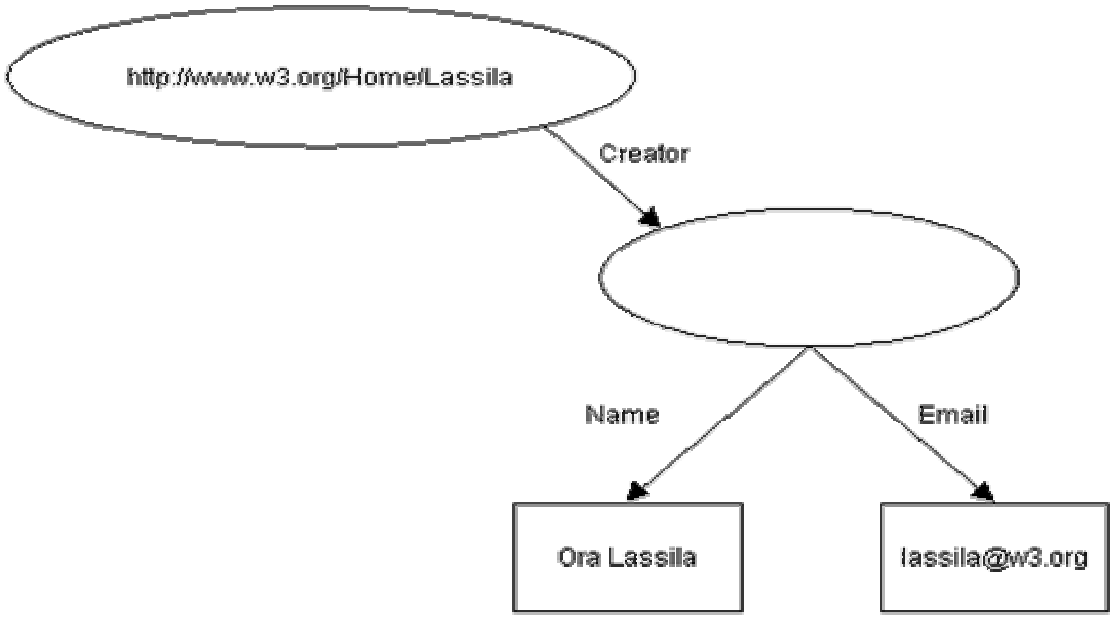
Pictorial Representation

- Directed labeled graphs
  - "nodes and arcs diagrams": the nodes (ovals) represent resources, arcs represent named properties and nodes (rectangles) represent string literals.
- "*http://www.w3.org/Home/Lassila has creator Ora Lassila*", or in general "<*subject*> *HAS* <*predicate*> <*object*>".



Property with Structured Value 1

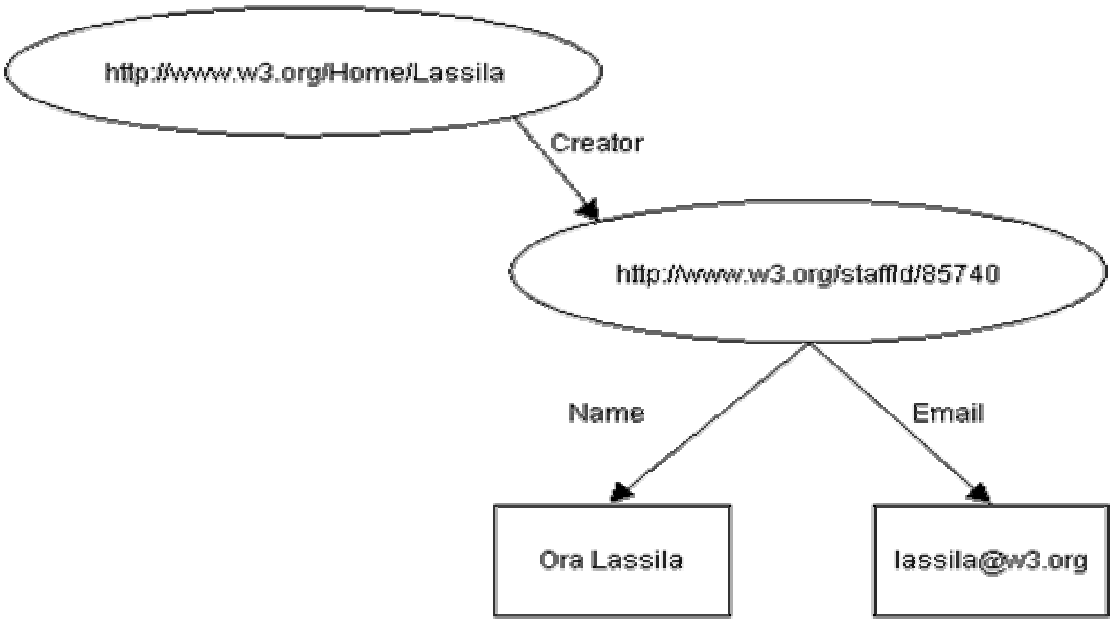
- *The individual whose name is Ora Lassila, email <lassila@w3.org>, is the creator of http://www.w3.org/Home/Lassila.*



Property with Structured Value 2

- *The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource*

  http://www.w3.org/Home/Lassila was created by this individual.



N3 (Notation3)

- N3 is a compact notation for triples which is easier for people to read and edit.

  Example:

<http://www.w3.org/Home/Lassila><has creator> <"Ora Lassila“>.

To use literal values, simply enclose the value in double quote marks, thus:

<http://www.w3.org/Home/Lassila><has creator> "Ora Lassila".

N3 Usages

- anonymous nodes: *"there is someone called... but without giving them a URI.*

_:a1 <http://xyz.org/#name> "Sean" .

- *@prefix xyz: <http://xyz.org/#> .*

For example, instead of writing:

<http://xyz.org/#a>

<http://xyz.org/#b>

<http://xyz.org/#c> .

We can instead do:

@prefix xyz: <http://xyz.org/#> .

:a :b :c .

N-Triples

- A syntax was devised to be an even simpler subset of N3, called *N-Triples*
- Say, it doesn't use prefixes.

XML RDF

Ora Lassila is the creator of the resource
http://www.w3.org/Home/Lassila.

is represented in RDF/XML as:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

XML → RDF (1)

- Example 2

  an XML document that specifies data about China's Yangtze river:

```
<?xml version="1.0"?>
<River id="Yangtze"
        xmlns="http://www.geodesy.org/river">
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</River>
```

"Here is data about the Yangtze River. It has a length of 6300 kilometers. Its startingLocation is western China's Qinghai-Tibet Plateau. Its endingLocation is the East China Sea."

XML → RDF (2)

Modify the following XML document so that it is also a valid RDF document:

XML

```
<?xml version="1.0"?>
<River id="Yangtze"
        xmlns="http://www.geodesy.org/river">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

Yangtze.xml

"convert to"

RDF

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

Yangtze.rdf

XML → RDF (2)

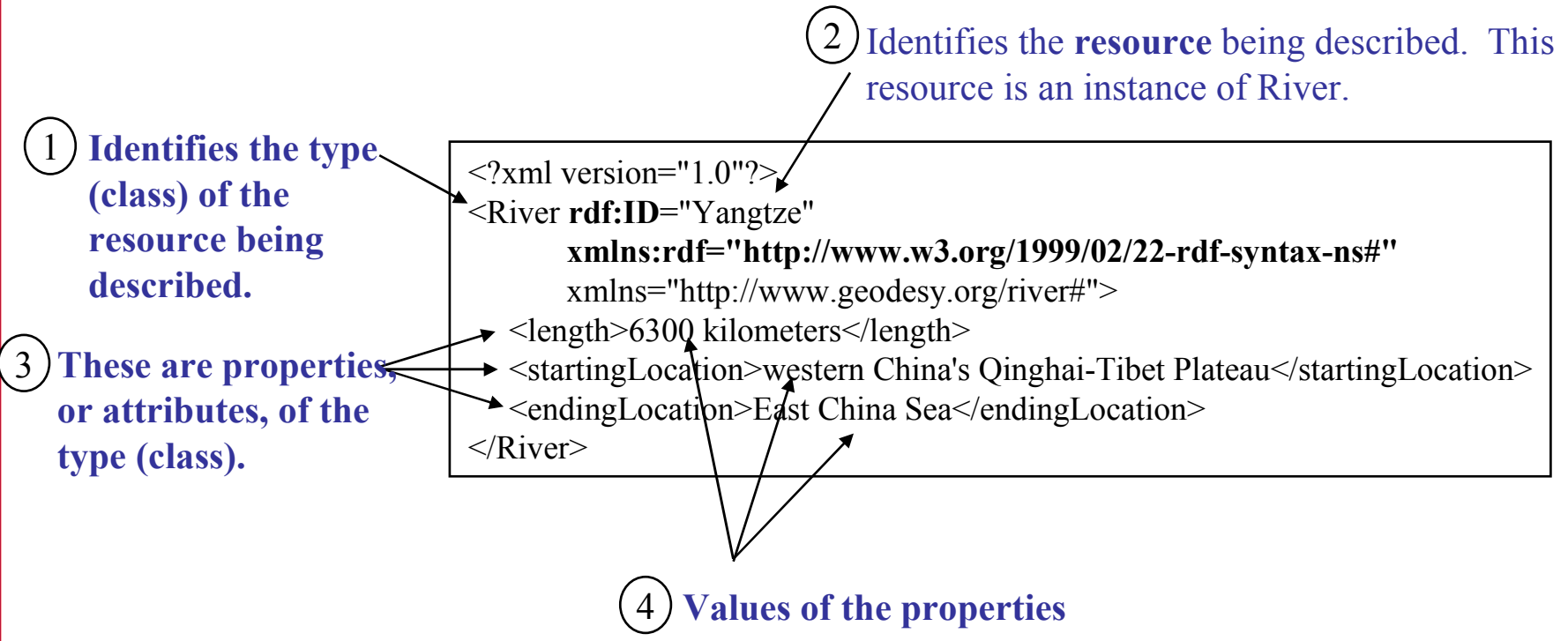
1 RDF provides an ID attribute for identifying the **resource** being described.

2 The ID attribute is in the RDF namespace.

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

3 Add the "fragment identifier symbol" to the namespace.

XML → RDF (4)


2 Identifies the **resource** being described. This resource is an instance of River.
1 Identifies the type (class) of the resource being described.
3 These are properties, or attributes, of the type (class).
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.geodesy.org/river#">
<length>6300 kilometers</length>
<startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
<endingLocation>East China Sea</endingLocation>
</River>
4 Values of the properties

RDF Format

```
<?xml version="1.0"?>
<Class rdf:ID="Resource"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="uri">
   <property>value</property>
   <property>value</property>
   ...
</Class>
```

Advantage of RDF Format

- You may ask: "Why should I bother designing my XML to be in the RDF format? Answer: there are numerous benefits:
  - The RDF format, if widely used, will help to make XML more interoperable:
    - Tools can instantly characterize the structure, "this element is a type (class), and here are its properties".
    - RDF promotes the use of standardized vocabularies ... standardized types (classes) and standardized properties.
  - The RDF format gives you a structured approach to designing your XML documents. The RDF format is a regular, recurring pattern.
  - It enables you to quickly identify weaknesses and inconsistencies of non-RDF-compliant XML designs. It helps you to better understand your data.
  - You reap the benefits of both worlds:
    - You can use standard XML editors and validators to create, edit, and validate your XML.
    - You can use the RDF tools to apply inferencing to the data.
  - It positions your data for the Semantic Web!

Disadvantage of RDF Format

- Constrained: the RDF format constrains you on how you design your XML (i.e., you can't design your XML in any arbitrary fashion).
- RDF uses namespaces to uniquely identify types (classes), properties, and resources. Thus, you must have a solid understanding of namespaces.
- Another XML vocabulary to learn: to use the RDF format you must learn the RDF vocabulary.

URI (Uniquely Identify the Resource)

- RDF is very concerned about uniquely identifying the type (class) and the properties. RDF is also very concerned about uniquely identifying the resource, e.g.,

This is the **resource** being described. We want to uniquely identify this resource.

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</River>
```

RDF Parser

- There is a nice RDF parser at the W3 Web site:

http://www.w3.org/RDF/Validator/

This RDF parser will tell you if your XML is in the proper RDF format.

Syntax

- Serialization Syntax
- Abbreviations

Basic Serialization Syntax

[1] RDF ::= ['<*rdf*:RDF>'] description* ['</*rdf*:RDF>']
[2] description ::= '<*rdf*:Description' idAboutAttr? '>' propertyElt* '</*rdf*:Description>'
[3] idAboutAttr ::= idAttr | aboutAttr
[4] aboutAttr ::= 'about="' URI-reference '"'
[5] idAttr ::= 'ID="' IDsymbol '"'
[6] propertyElt ::= '<' propName '>' value '</' propName '>'
| '<' propName resourceAttr '/>'
[7] propName ::= Qname
[8] value ::= description | string
[9] resourceAttr ::= 'resource="' URI-reference '"'
[10] Qname ::= [NSprefix ':'] name
[11] URI-reference ::= string, interpreted per [URI]
[12] IDsymbol ::= (any legal XML name symbol)
[13] name ::= (any legal XML name symbol)
[14] NSprefix ::= (any legal XML namespace prefix)
[15] string ::= (any XML text, with "<", ">", and "&" escaped)

Example 3

Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.

is represented in RDF/XML as:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Example 3 (contd.)

- The namespace prefix 's‘:
  - xmlns:s="http://description.org/schema/"
- This namespace declaration would typically be included as an XML attribute on the rdf:RDF element but may also be included with a particular Description element or even an individual propertyElt expression.
- The namespace name URI in the namespace declaration is a globally unique identifier for the particular schema this metadata author is using to define the use of the Creator property.
- Other schemas may also define aproperty named Creator and the two properties will be distinguished via their schema identifiers.
- Note also that a schema usually defines several properties; a single namespace declaration will suffice to make a large vocabulary of properties available for use.

Example 3 (contd.)

The complete XML document containing the description above would be:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:s="http://description.org/schema/">
   <rdf:Descriptionabout="http://www.w3.org/Home/Lassila">
     <s:Creator>OraLassila</s:Creator>
   </rdf:Description>
</rdf:RDF>
```

Basic Abbreviated Syntax

Three forms of abbreviation are defined for the basic serialization syntax:

1. The first is usable for properties that are not repeated within a Description and where the values of those properties are literals. In this case, the properties may be written as XML attributes of the Description element.
2. The second RDF abbreviation form works on nested Description elements. This abbreviation form can be employed for specific statements when the object of the statement is another resource and the values of any properties given in-line for this second resource are strings. In this case, a similar transformation of XML element names into XML attributes is used: the properties of the resource in the nested Description may be written as XML attributes of the propertyElt element in which that Description was contained.
3. The third basic abbreviation applies to the common case of a Description element containing a type property. In this case, the resource type defined in the schema corresponding to the value of the type property can be used directly as an element name.

Example 4
Abbreviation 1 – Element -> Attribute

- Example 1 becomes:

```
<rdf:RDF>
<rdf:Description about="http://www.w3.org/Home/Lassila"
                 s:Creator="Ora Lassila" />
</rdf:RDF>
```

- Since the Description element has no other content once the Creator property is written in XML attribute form, the XML empty element syntax is employed to elide the Description end-tag.

Example 5
Abbreviation 2 – Element -> Attribute

"The individual referred to by employee id 85740 is named OraLassilaand has the email address lassila@w3.org. The resource http://www.w3.org/Home/Lassila was created by this individual."

is written in RDF/XML using explicit serialization form as

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
  </rdf:Description>

  <rdf:Description about="http://www.w3.org/staffId/85740">
    <v:Name>OraLassila</v:Name>
    <v:Email>lassila@w3.org</v:Email>
  </rdf:Description>
</rdf:RDF>
```

Example 5'

- This form makes it clear to a reader that two separate resources are being described but it is less clear that the second resource is used within the first description. This same expression could be written in the following way to make this relationship more obvious to the human reader. Note that to the machine, there is no difference:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <rdf:Description about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
      </rdf:Description>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Example 5”

- Using the second basic abbreviation syntax, the inner Description element and its contained property expressions can be written as attributes of the Creator element:

```
<rdf:RDF>
 <rdf:Description about="http://www.w3.org/Home/Lassila">
     <s:Creator rdf:resource="http://www.w3.org/staffId/85740"
         v:Name="Ora Lassila"
         v:Email="lassila@w3.org" />
     </rdf:Description>
</rdf:RDF>
```

Example 6
Abbreviation 3 – Type -> Attribute

- Using the previous RDF fragment if we wanted to add the fact that the resource http://www.w3.org/staffId/85740 represents an instance of a Person, we would write this in full serialization syntax as:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#“
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <rdf:Description about="http://www.w3.org/staffId/85740">
        <rdf:type resource="http://description.org/schema/Person"/>
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
      </rdf:Description>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Example 6'

- And using this third abbreviated form as:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <s:Person about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
      </s:Person>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>
```

rdf:ID

- The value of rdf:ID is a "relative URI".
- The "complete URI" is obtained by concatenating the URL of the XML document with "#" and then the value of rdf:ID, e.g.,

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</River>
```

Yangtze.rdf

Suppose that this RDF/XML document is located at this URL: http://www.china.org/geography/rivers. Thus, the complete URI for this resource is:

http://www.china.org/geography/rivers#Yangtze

xml:base

- On the previous slide we showed how the URL of the document provided the base URI.
- Depending on the location of the document is brittle: it will break if the document is moved, or is copied to another location.
- A more robust solution is to specify the base URI in the document, e.g.,

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#"
        xml:base="http://www.china.org/geography/rivers">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

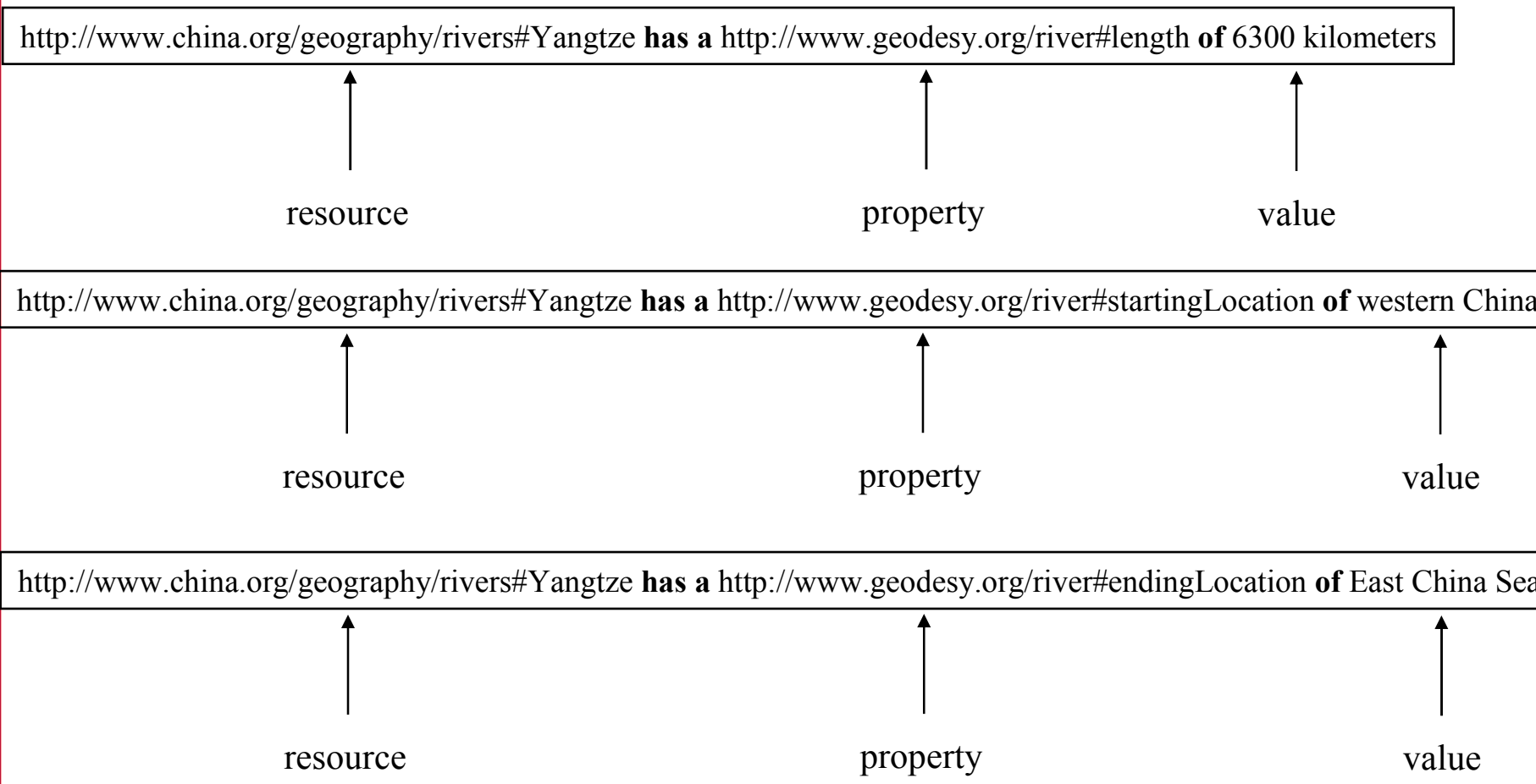
Resource URI = concatenation(xml:base, '#', rdf:ID)
= concatenation(http://www.china.org/geography/rivers, **'#',** "Yangtze")
= http://www.china.org/geography/rivers#Yangtze

rdf:about

- Instead of identifying a resource with a relative URI (which then requires a base URI to be prepended), we can give the complete identity of a resource. However, we use rdf:about, rather than rdf:ID, e.g.,

```
<?xml version="1.0"?>
<River rdf:about="http://www.china.org/geography/rivers#Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

Triple -> resource/property/value

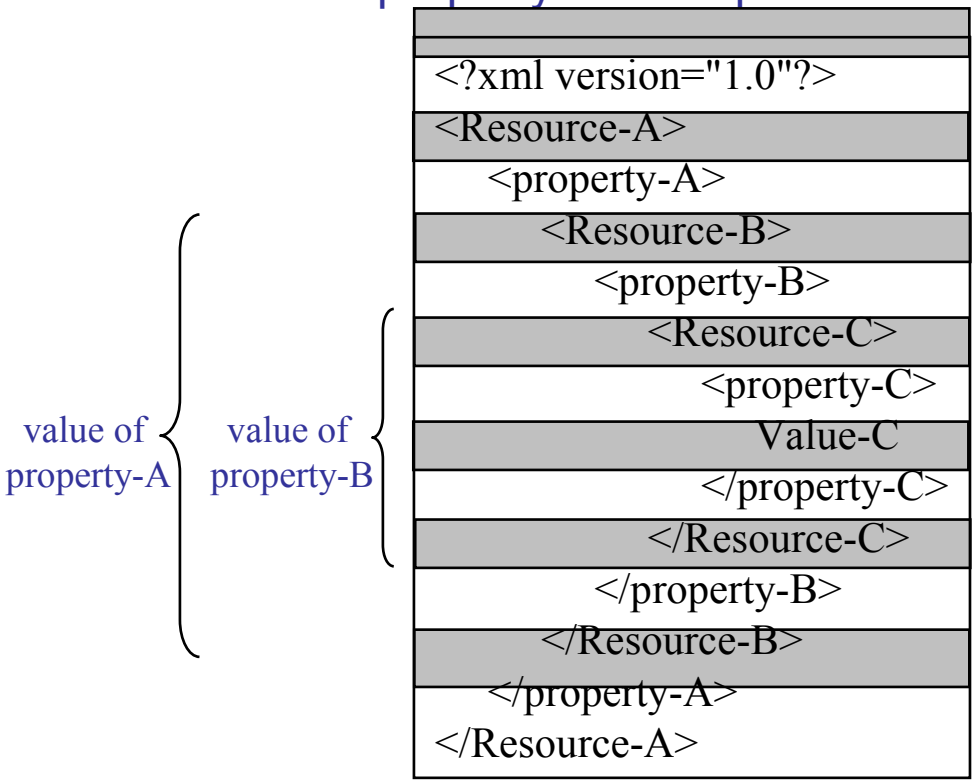


Graph

Legend:
Ellipse indicates "Resource"
Rectangle indicates "literal string value"

RDF Format = triples!

- The fundamental design pattern of RDF is to structure your XML data as resource/property/value triples.



Notice that the RDF design pattern is an alternating sequence of resource-property.

This pattern is known as "**striping**".

The value of a property can be a literal (e.g., length has a value of 6300 kilometers). Also, the value of a property can be a resource, as shown above (e.g., property-A has a value of Resource-B, property-B has a value of Resource-C). We will see examples of properties having a resource value in a little bit.

Naming Convention

- The convention is to use a capital letter to start a type (class) name, and use a lowercase letter to start a property name.
  - This helps the eye quickly discern the striping pattern.

uppercase

lowercase

```
<?xml version="1.0"?>
<River rdf:about="http://www.china.org/geography/rivers#Yangtze"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.geodesy.org/river#">
  <length>6300 kilometers</length>
  <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
  <endingLocation>East China Sea</endingLocation>
</River>
```

Example 7 rdf:Description + rdf:type

- There is still another way of representing the XML. This way makes it very clear that you are describing something, and it makes it very clear what the type (class) is of the thing you are describing:

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://www.china.org/geography/rivers#Yangtze"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.geodesy.org/river#">
   <rdf:type rdf:resource="http://www.geodesy.org/river#River"/>
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</rdf:Description>
```

This is read as: "This is a Description about the resource http://www.china.org/geography/rivers#Yangtze This resource is an instance of the River type (class). The http://www.china.org/geography/rivers#Yangtz resource has a length of 6300 kilometers, a startingLocation of western China's Qinghai-Tibet Plateau, and an endingLocation of the East China Sea."

ote: this form of describing a resource is called the "long form". The form we have seen previously is an bbreviation of this long form.

Example 7’

- Alternatively, we can use rdf:ID rather than rdf:about, as shown here:

```
<?xml version="1.0"?>
<rdf:Description rdf:ID="Yangtze"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.geodesy.org/river#"
                 xml:base="http://www.china.org/geography/rivers">
   <rdf:type rdf:resource="http://www.geodesy.org/river#River"/>
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</rdf:Description>
```

Example 7''- Equivalent Representations

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#"
        xml:base="http://www.china.org/geography/rivers">
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</River>
```

```
<?xml version="1.0"?>
<River rdf:about="http://www.china.org/geography/rivers#Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</River>
```

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://www.china.org/geography/rivers#Yangtze"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.geodesy.org/river#">
   <rdf:type rdf:resource="http://www.geodesy.org/river#River"/>
   <length>6300 kilometers</length>
   <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
   <endingLocation>East China Sea</endingLocation>
</rdf:Description>
```

← Note: In the RDF literature the examples are typically shown in this form.

Basic Abbreviated Syntax

- The EBNF for the basic abbreviated syntax replaces productions [2] and [6] of the grammar for the basic serialization syntax in the following manner:

[2a] description ::= '<*rdf*:Description' idAboutAttr? propAttr* '/>'
| '<*rdf*:Description' idAboutAttr? propAttr* '>'
propertyElt* '</*rdf*:Description>' | typedNode
[6a] propertyElt ::= '<' propName '>' value '</' propName '>'
| '<' propName resourceAttr? propAttr* '/>'
[16] propAttr ::= propName '="' string '"'
(with embedded quotes escaped)
[17] typedNode ::= '<' typeName idAboutAttr? propAttr* '/>'
| '<' typeName idAboutAttr? propAttr* '>'
property* '</' typeName '>'

Schemas and Namespaces

- Meaning in RDF is expressed through reference to a *schema*.
  - A schema defines the terms that will be used in RDF statements and gives specific meanings to them.
  - A variety of schema forms can be used with RDF, including a specific form defined in RDF Schema that has some specific characteristics to help with automating tasks using RDF.
- RDF uses the XML namespace facility.
  - Namespaces are simply a way to tie a specific use of a word in context to the dictionary (schema) where the intended definition is to be found.
  - In RDF, each predicate used in a statement must be identified with exactly one namespace, or schema. However, a Description element may contain statements with predicates from many schemas.

Qualified Property Values

- Often the value of a property is something that has additional contextual information that is considered "part of" that value. In other words, there is a need to qualify property values.
- In the RDF model a qualified property value is simply another instance of a structured value. The object of the original statement is this structured value and the qualifiers are further properties of this common resource. The principal value being qualified is given as the value of the *value* property of this common resource.

Containers

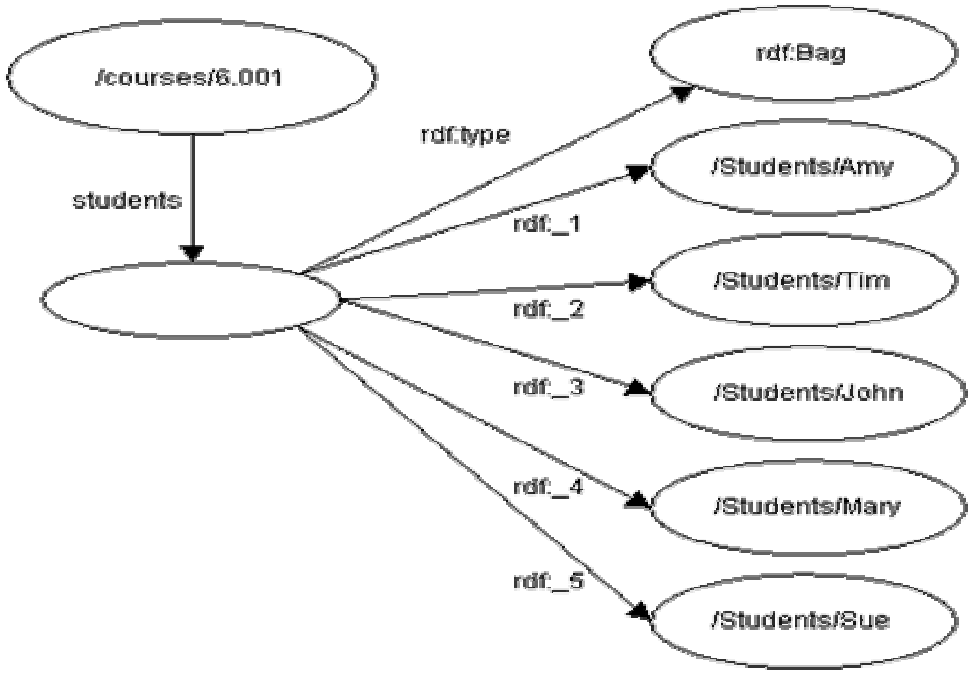
RDF defines three types of container objects:

- **Bag.** An unordered list of resources or literals. *Bag*s are used to declare that a property has multiple values and that there is no significance to the order in which the values are given. *Bag* might be used to give a list of part numbers where the order of processing the parts does not matter. Duplicate values are permitted.
- **Sequence.** An ordered list of resources or literals. *Sequence* is used to declare that a property has multiple values and that the order of the values is significant. *Sequence* might be used, for example, to preserve an alphabetical ordering of values. Duplicate values are permitted.
- **Alternative.** A list of resources or literals that represent alternatives for the (single) value of a property. *Alternative* might be used to provide alternative language translations for the title of a work, or to provide a list of Internet mirror sites at which a resource might be found. An application using a property whose value is an *Alternative* collection is aware that it can choose any one of the items in the list as appropriate.

Example 8 - Bag

"The students in course 6.001 are Amy, Tim, John, Mary, and Sue."

The RDF model is



Example 8' - Bag

- Written in RDF/XML as

```
<rdf:RDF>
  <rdf:Description about="http://mycollege.edu/courses/6.001">
    <s:students>
      <rdf:Bag>
        <rdf:li resource="http://mycollege.edu/students/Amy"/>
        <rdf:li resource="http://mycollege.edu/students/Tim"/>
        <rdf:li resource="http://mycollege.edu/students/John"/>
        <rdf:li resource="http://mycollege.edu/students/Mary"/>
        <rdf:li resource="http://mycollege.edu/students/Sue"/>
      </rdf:Bag>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

Example 9 – Seq

- *"The students in course 6.001 are ranked by Amy, Tim, John, Mary, and Sue."*

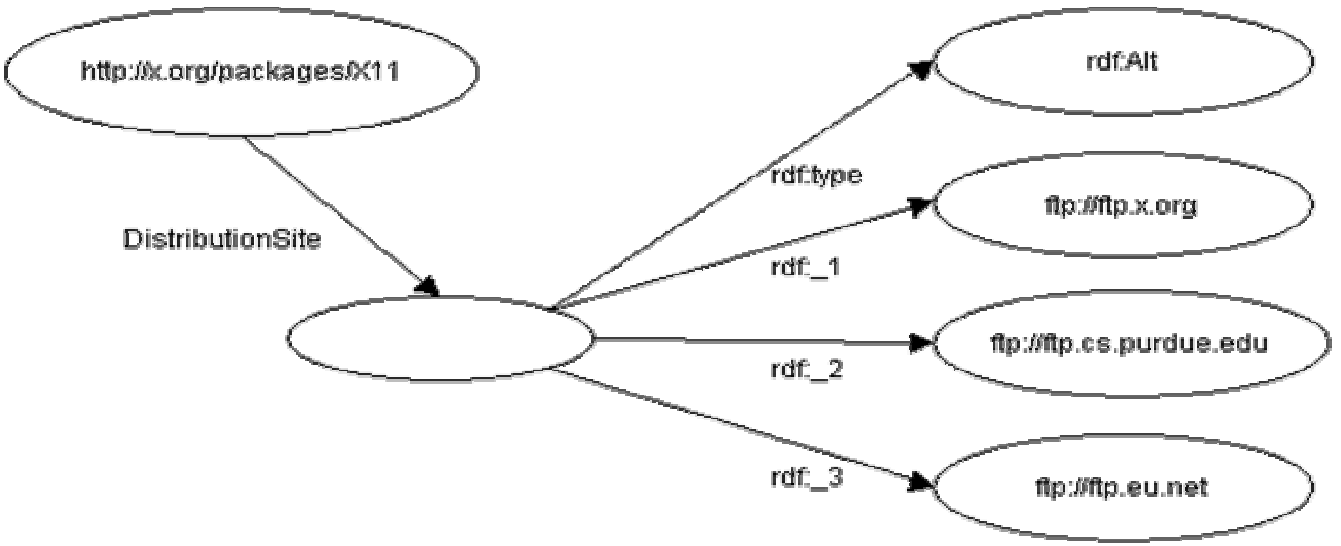
Written in RDF/XML as

```
<rdf:RDF>
  <rdf:Description about="http://mycollege.edu/courses/6.001">
    <s:students>
      <rdf:Seq>
        <rdf:li resource="http://mycollege.edu/students/Amy"/>
        <rdf:li resource="http://mycollege.edu/students/Tim"/>
        <rdf:li resource="http://mycollege.edu/students/John"/>
        <rdf:li resource="http://mycollege.edu/students/Mary"/>
        <rdf:li resource="http://mycollege.edu/students/Sue"/>
      </rdf:Seq>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

Example 10 - Alt

- *"The source code for X11 may be found at ftp.x.org, ftp.cs.purdue.edu, or ftp.eu.net."*

is modeled in RDF as



Example 10' - Alt

- Written in RDF/XML as

```
<rdf:RDF>
  <rdf:Description about="http://x.org/packages/X11">
    <s:DistributionSite>
      <rdf:Alt>
        <rdf:li resource="ftp://ftp.x.org"/>
        <rdf:li resource="ftp://ftp.cs.purdue.edu"/>
        <rdf:li resource="ftp://ftp.eu.net"/>
       </rdf:Alt>
    </s:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

Container Syntax

- RDF container syntax takes the form:

[18] container ::= sequence | bag | alternative
[19] sequence ::= '<*rdf*:Seq' idAttr? '>' member* '</*rdf*:Seq>'
[20] bag ::= '<*rdf*:Bag' idAttr? '>' member* '</*rdf*:Bag>'
[21] alternative ::= '<*rdf*:Alt' idAttr? '>' member+ '</*rdf*:Alt>'
[22] member ::= referencedItem | inlineItem
[23] referencedItem ::= '<*rdf*:li' resourceAttr '/>'
[24] inlineItem ::= '<*rdf*:li>' value '</*rdf*:li>'

- Containers may be used everywhere a Description is permitted:

[1a] RDF ::= '<*rdf*:RDF>' obj* '</*rdf*:RDF>'
[8a] value ::= obj | string
[25] obj ::= description | container

Example 11 - Distributive Referents: Statements about Members of a Container

```
<rdf:Bag ID="pages">
   <rdf:li resource="http://foo.org/foo.html" />
   <rdf:li resource="http://bar.org/bar.html" />
</rdf:Bag>

<rdf:Description about="#pages">
   <s:Creator>Ora Lassila</s:Creator>
</rdf:Description>
```

abouteach

[3a] idAboutAttr ::= idAttr | aboutAttr | aboutEachAttr

[26] aboutEachAttr ::= 'aboutEach="' URI-reference '"'

Example 12 - abouteach

```
<rdf:Description aboutEach="#pages">
  <s:Creator>Ora Lassila</s:Creator>
</rdf:Description>

<rdf:Description about="http://foo.org/foo.html">
  <s:Creator>Ora Lassila</s:Creator>
</rdf:Description>

<rdf:Description about="http://bar.org/bar.html">
  <s:Creator>Ora Lassila</s:Creator>
</rdf:Description>
```

Containers Defined By A URI Pattern

- One very frequent use of metadata is to make statements about "all pages at my Web site". In many cases it is impractical or even undesirable to try to list each such resource explicitly and identify it as a member of a container. RDF therefore has a second distributive referent type. This second distributive referent type is a shorthand syntax that represents an instance of a Bag whose members are by definition all resources whose resource identifiers begin with a specified string:

[26a] aboutEachAttr ::= 'aboutEach="' URI-reference '"'
| 'aboutEachPrefix="' string '"'

- The aboutEachPrefix attribute declares that there is a Bag whose members are all the resources whose fully resolved resource identifiers begin with the character string given as the value of the attribute. The statements in a Description that has the aboutEachPrefix attribute apply individually to each of the members of this Bag.

Example 13 - abouteachprefix

If the two resources http://foo.org/doc/page1 and http://foo.org/doc/page2 exist then we can say that each of them has a copyright property by writing:

<*rdf*:Description aboutEachPrefix="http://foo.org/doc">
<*s*:Copyright>© 1998, The Foo Organization</*s*:Copyright>
</*rdf*:Description>

Containers Vs. Repeated Properties

- A resource may have multiple statements with the same predicate (i.e., using the same property). This is not the same as having a single statement whose object is a container containing multiple members. The choice of which to use in any particular circumstance is in part made by the person who designs the schema and in part made by the person who writes the specific RDF statements.

Statements about Statements

- Statements: about resources.
- Higher-order statements: making statements about other statements.
- In order to make a statement about another statement, we actually have to build a model of the original statement; this model is a new resource to which we can attach additional properties.

Example 14 – Two-order Statement

Consider the sentence:

- *Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.*

RDF would regard this sentence as a fact. Consider the sentence:

- *Ralph Swick says that Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.*

we have said nothing about the resource http://www.w3.org/Home/Lassila; instead, we have expressed a fact about a statement Ralph has made.

- In order to express this fact to RDF, we have to model the original statement as a resource with four properties.

Statement Model

- A model of a statement is the resource we need in order to be able to make new statements (higher-order statements) about the modeled statement.
- A model of a statement is called a *reified statement*.
- This process is formally called *reification.*

Reification

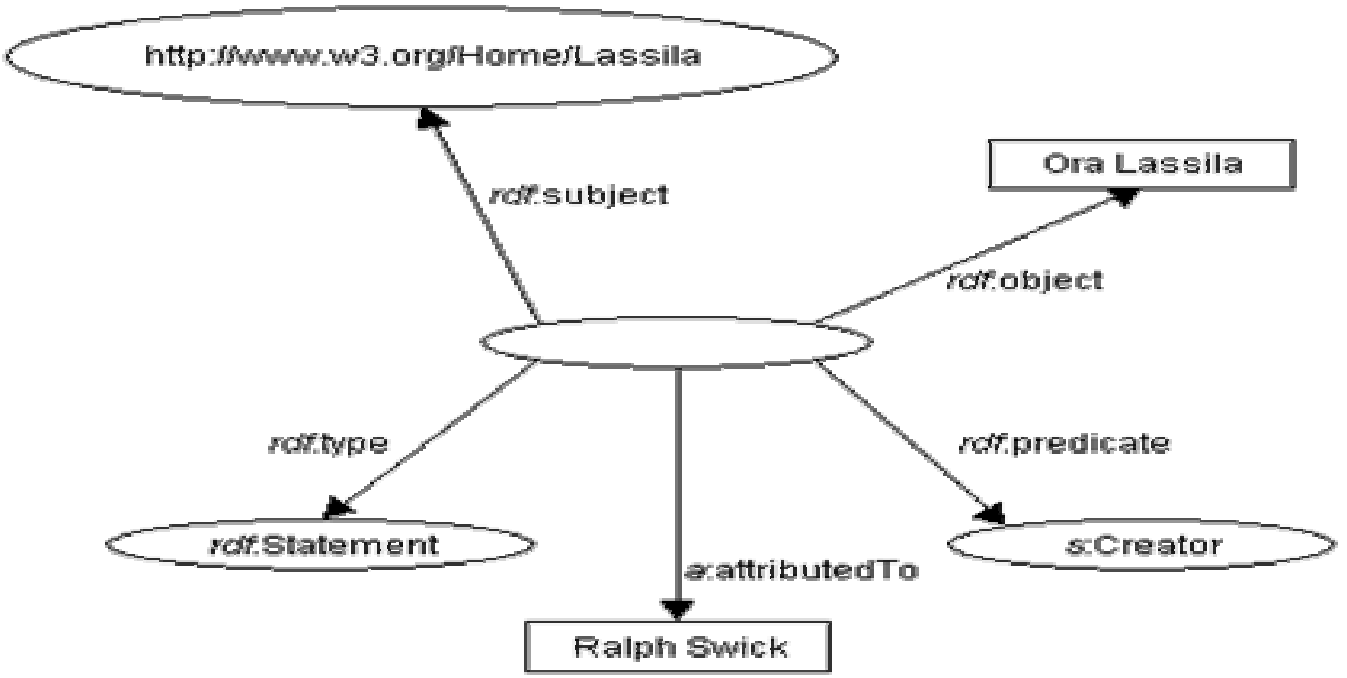
To model statements RDF defines the following properties:

- Subject. The *subject* property identifies the resource being described by the modeled statement; that is, the value of the *subject* property is the resource about which the original statement was made.
- Predicate. The *predicate* property identifies the original property in the modeled statement. The value of the *predicate* property is a resource representing the specific property in the original statement.
- Object. The *object* property identifies the property value in the modeled statement. The value of the *object* property is the object in the original statement.
- Type. The value of the *type* property describes the type of the new resource. All reified statements are instances of RDF:Statement; that is, they have a *type* property whose object is RDF:Statement. The *type* property is also used more generally to declare the type of any resource.

1. A new resource with the above four properties represents the original statement and can both be used as the object of other statements and have additional statements made about it.
2. The resource with these four properties is not a replacement for the original statement, it is a model of the statement. A statement and its corresponding reified statement exist independently in an RDF graph and either may be present without the other.

Example 14'

- Attaching another property to the reified statement (say, "attributedTo") with an appropriate value (in this case, "Ralph Swick").



Example 14”

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:a="http://description.org/schema/">
  <rdf:Description>
    <rdf:subject resource="http://www.w3.org/Home/Lassila" />
    <rdf:predicate resource="http://description.org/schema/Creator" />
    <rdf:object>Ora Lassila</rdf:object>
    <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#Statement" />
    <a:attributedTo>Ralph Swick</a:attributedTo>
  </rdf:Description>
</rdf:RDF>
```

ID & BagID

[2b] description ::= '<*rdf*:Description' idAboutAttr? bagIDAttr? propAttr* '/>' | '<*rdf*:Description' idAboutAttr? bagIDAttr? propAttr* '>' propertyElt* '</*rdf*:Description>'

[27] bagIDAttr ::= 'bagID="' IDsymbol '"'

Note: (1) ID specifies the identification of an in-line resource whose properties are further detailed in the Description.

(2) BagID specifies the identification of the container resource whose members are the reified statements about another resource.

(3) A Description may have both an ID attribute and a bagID attribute.

Syntactic Shorthand for Statements About Statements

- Since attaching a bagID to a Description results in including in the model a Bag of the reified statements of the Description, we can use this as a syntactic shorthand when making statements about statements.
- Example

  if we wanted to say that Ralph states that Ora is the creator of http://www.w3.org/Home/Lassila and that he also states that the title of that resource is "Ora's Home Page", we can simply add to the example above:

```
<rdf:Description aboutEach="#D_001">
  <a:attributedTo>Ralph Swick</a:attributedTo>
</rdf:Description>
```

Formal Model

- RDF specification shows three representations of the data model:
  - N3: 3-tuples (triples),
  - Graph: Directed label graph,
  - XML: RDF/XML.

  These representations have equivalent meaning. The mapping between the representations is not intended to constrain in any way the internal representation used by implementations.
- RDF data model is defined formally as follows:

1. There is a set called *Resources*.
2. There is a set called *Literals*.
3. There is a subset of *Resources* called *Properties*.
4. There is a set called *Statements*, each element of which is a triple of the form

   {pred, sub, obj}

Where pred is a property (member of *Properties*), sub is a resource (member of *Resources*), and obj is either a resource or a literal (member of *Literals*).

Formal Model (contd.)

5. There is an element of *Properties* known as RDF:type.
6. Members of *Statements* of the form {RDF:type, sub, obj} must satisfy the following: sub and obj are members of *Resources.* [RDFSchema] places additional restrictions on the use of type.

7. There is an element of *Resources*, not contained in *Properties*, known as RDF:Statement.
8. There are three elements in *Properties* known as RDF:predicate, RDF:subject and RDF:object.
9. Reification of a triple {pred, sub, obj} of *Statements* is an element r of *Resources* representing the reified triple and the elements s1, s2, s3, and s4 of *Statements* such that

   s1: {RDF:predicate, r, pred}
   s2: {RDF:subject, r, subj}
   s3: {RDF:object, r, obj}
   s4: {RDF:type, r, [RDF:Statement]}

10. There are three elements of *Resources*, not contained in *Properties*, known as RDF:Seq, RDF:Bag, and RDF:Alt.
11. There is a subset of *Properties* corresponding to the ordinals (1, 2, 3, ...) called *Ord*. We refer to elements of *Ord* as RDF:_1, RDF:_2, RDF:_3, ...

Formal Grammar

[1] RDF ::= ['<*rdf*:RDF>'] obj* ['</*rdf*:RDF>']
[2] obj ::= description | container
[3] description ::= '<*rdf*:Description' idAboutAttr? bagIdAttr? propAttr* '/>' | '<*rdf*:Description' idAboutAttr? bagIdAttr? propAttr* '>' propertyElt* '</*rdf*:Description>' | typedNode
[4] container ::= sequence | bag | alternative
[5] idAboutAttr ::= idAttr | aboutAttr | aboutEachAttr
[6] idAttr ::= ' ID="' IDsymbol '"'
[7] aboutAttr ::= ' about="' URI-reference '"'
[8] aboutEachAttr ::= ' aboutEach="' URI-reference '"' | ' aboutEachPrefix="' string '"'
[9] bagIdAttr ::= ' bagID="' IDsymbol '"'
[10] propAttr ::= typeAttr | propName '="' string '"' (with embedded quotes escaped)
[11] typeAttr ::= ' type="' URI-reference '"'
[12] propertyElt ::= '<' propName idAttr? '>' value '</' propName '>' | '<' propName idAttr? parseLiteral '>' literal '</' propName '>' | '<' propName idAttr? parseResource '>' propertyElt* '</' propName '>' | '<' propName idRefAttr? bagIdAttr? propAttr* '/>'
[13] typedNode ::= '<' typeName idAboutAttr? bagIdAttr? propAttr* '/>' | '<' typeName idAboutAttr? bagIdAttr? propAttr* '>' propertyElt* '</' typeName '>'
[14] propName ::= Qname
[15] typeName ::= Qname [6.16] idRefAttr ::= idAttr | resourceAttr

Formal Grammar (contd.)

[17] value ::= obj | string
[18] resourceAttr ::= ' resource="' URI-reference '"'
[19] Qname ::= [NSprefix ':'] name
[20] URI-reference ::= string, interpreted per [URI]
[21] IDsymbol ::= (any legal XML name symbol)
[22] name ::= (any legal XML name symbol)
[23] NSprefix ::= (any legal XML namespace prefix)
[24] string ::= (any XML text, with "<", ">", and "&" escaped)
[25] sequence ::= '<*rdf*:Seq' idAttr? '>' member* '</*rdf*:Seq>' | '<*rdf*:Seq' idAttr? memberAttr* '/>'
[26] bag ::= '<*rdf*:Bag' idAttr? '>' member* '</*rdf*:Bag>' | '<*rdf*:Bag' idAttr? memberAttr* '/>'
[27] alternative ::= '<*rdf*:Alt' idAttr? '>' member+ '</*rdf*:Alt>' | '<*rdf*:Alt' idAttr? memberAttr? '/>'
[28] member ::= referencedItem | inlineItem
[29] referencedItem ::= '<*rdf*:li' resourceAttr '/>'
[30] inlineItem ::= '<*rdf*:li '>' value </*rdf*:li>' | '<*rdf*:li' parseLiteral '>' literal </*rdf*:li>' | '<*rdf*:li' parseResource '>' propertyElt* </*rdf*:li>'
[31] memberAttr ::= ' *rdf*:_*n*="' string '"' (where *n* is an integer)
[32] parseLiteral ::= ' parseType="Literal"'
[33] parseResource ::= ' parseType="Resource"'
[34] literal ::= (any well-formed XML)

Example 15 - Sharing Values

- A single resource can be the value of more than one property; that is, it can be the object of more than one statement and therefore pointed to by more than one arc.

Consider the case of specifying the collected works of an author, sorted once by publication date and sorted again alphabetically by subject:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <Seq ID="JSPapersByDate">
    <li resource="http://www.dogworld.com/Aug96.doc"/>
    <li resource="http://www.webnuts.net/Jan97.html"/>
    <li resource="http://www.carchat.com/Sept97.html"/>
  </Seq>
  <Seq ID="JSPapersBySubj">
    <li resource="http://www.carchat.com/Sept97.html"/>
    <li resource="http://www.dogworld.com/Aug96.doc"/>
    <li resource="http://www.webnuts.net/Jan97.html"/>
  </Seq>
</RDF>
```

Example 16 - Aggregates

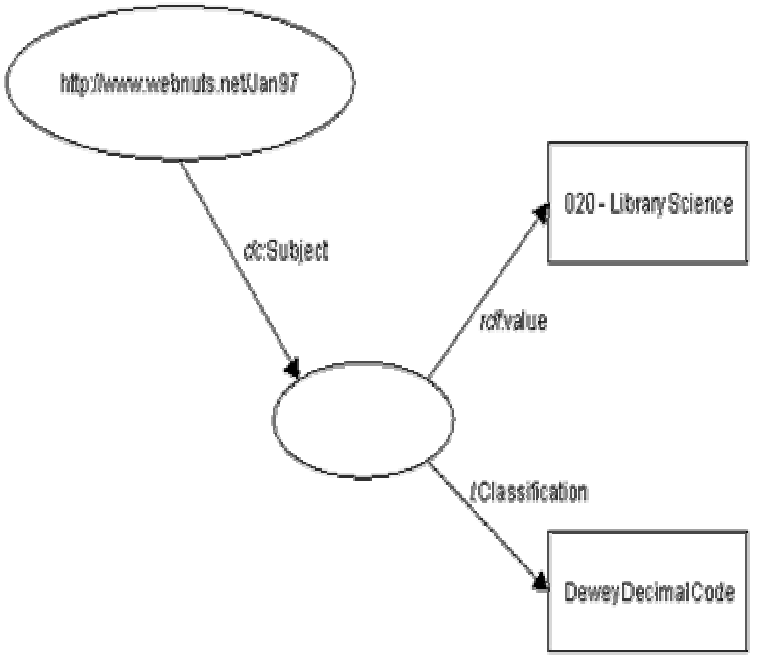
- Consider an example of a document with two authors specified alphabetically, a title specified in two different languages, and having two equivalent locations on the Web:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/metadata/dublin_core#">
  <rdf:Description about="http://www.foo.com/cool.html">
    <dc:Creator>
      <rdf:Seq ID="CreatorsAlphabeticalBySurname">
        <rdf:li>Mary Andrew</rdf:li>
        <rdf:li>Jacky Crystal
        </rdf:li> </rdf:Seq>
    </dc:Creator>

    <dc:Identifier>
      <rdf:Bag ID="MirroredSites">
        <rdf:li rdf:resource="http://www.foo.com.au/cool.html"/>
        <rdf:li rdf:resource="http://www.foo.com.it/cool.html"/>
      </rdf:Bag>
     </dc:Identifier>

     <dc:Title>
       <rdf:Alt>
         <rdf:li xml:lang="en">The Coolest Web Page</rdf:li>
         <rdf:li xml:lang="it">Il Pagio di Web Fuba</rdf:li>
         </rdf:Alt>
     </dc:Title>
  </rdf:Description>
</rdf:RDF>
```

Example 17 - Non-Binary Relations

- The RDF data model intrinsically only supports binary relations. To represent higher arity relations in RDF, we use just binary relations: using an intermediate resource with additional properties of this resource giving the remaining relations.

Consider: a ternary relation



Example 17'

```
<RDF
  xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns:I="http://mycorp.com/schemas/my-schema#">
  <Description about="http://www.webnuts.net/Jan97.html">
    <dc:Subject rdf:value="020 - Library Science"
   I:Classification="Dewey Decimal Code"/>
  </Description>
</RDF>
```

Example 18 - Dublin Core Metadata

- Web site home page

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/metadata/dublin_core#">
  <rdf:Description about="http://www.dlib.org">
    <dc:Title>D-Lib Program - Research in Digital Libraries</dc:Title>
    <dc:Description>The D-Lib program supports the community of people with research interests in digital libraries
      and electronic publishing.</dc:Description>
    <dc:Publisher>Corporation For National Research Initiatives</dc:Publisher>
    <dc:Date>1995-01-07</dc:Date>
    <dc:Subject>
      <rdf:Bag>
        <rdf:li>Research; statistical methods</rdf:li>
        <rdf:li>Education, research, related topics</rdf:li>
        <rdf:li>Library use Studies</rdf:li>
      </rdf:Bag>
    </dc:Subject>
    <dc:Type>World Wide Web Home Page</dc:Type>
    <dc:Format>text/html</dc:Format>
    <dc:Language>en</dc:Language>
  </rdf:Description>
</rdf:RDF>
```

Example 18' - a published magazine

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns:dcq="http://purl.org/metadata/dublin_core_qualifiers#">
  <rdf:Description about="http://www.dlib.org/dlib/may98/05contents.html">
    <dc:Title>DLIB Magazine - The Magazine for Digital Library Research - May 1998</dc:Title>
    <dc:Description>D-LIB magazine is a monthly compilation of contributed stories, commentary, and
      briefings.</dc:Description>
    <dc:Contributor rdf:parseType="Resource">
      <dcq:AgentType rdf:resource="http://purl.org/metadata/dublin_core_qualifiers#Editor"/>
      <rdf:value>Amy Friedlander</rdf:value>
    </dc:Contributor>
    <dc:Publisher>Corporation for National Research Initiatives</dc:Publisher>
    <dc:Date>1998-01-05</dc:Date>
    <dc:Type>electronic journal</dc:Type>
    <dc:Subject>
      <rdf:Bag>
        <rdf:li>library use studies</rdf:li>
        <rdf:li>magazines and newspapers</rdf:li>
        </rdf:Bag>
     </dc:Subject>
     <dc:Format>text/html</dc:Format>
     <dc:Identifier>urn:issn:1082-9873</dc:Identifier>
     <dc:Relation rdf:parseType="Resource">
     <dcq:RelationType rdf:resource="http://purl.org/metadata/dublin_core_qualifiers#IsPartOf"/>
     <rdf:value resource="http://www.dlib.org"/>
     </dc:Relation>
  </rdf:Description>
</rdf:RDF>
```

Example 18” - a specific article

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns:dcq="http://purl.org/metadata/dublin_core_qualifiers#">
  <rdf:Description about= "http://www.dlib.org/dlib/may98/miller/05miller.html">
  <dc:Title>An Introduction to the Resource Description Framework</dc:Title>
  <dc:Creator>Eric J. Miller</dc:Creator>
  <dc:Description>The Resource Description Framework (RDF) is …  . </dc:Description>
  <dc:Publisher>Corporation for National Research Initiatives</dc:Publisher>
  <dc:Subject>
    <rdf:Bag>
      <rdf:li>machine-readable catalog record formats</rdf:li>
      <rdf:li>applications of computer file organization and access methods</rdf:li>
    </rdf:Bag>
   </dc:Subject>
  <dc:Rights>Copyright @ 1998 Eric Miller</dc:Rights>
  <dc:Type>Electronic Document</dc:Type>
  <dc:Format>text/html</dc:Format>
  <dc:Language>en</dc:Language>
  <dc:Relation rdf:parseType="Resource">
  <dcq:RelationType rdf:resource="http://purl.org/metadata/dublin_core_qualifiers#IsPartOf"/>
    <rdf:value resource="http://www.dlib.org/dlib/may98/05contents.html"/>
  </dc:Relation>
  </rdf:Description>
</rdf:RDF>
```

Example 19 - PICS Labels

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pics="http://www.w3.org/TR/xxxx/WD-PICS-labels#"
  xmlns:gcf="http://www.gcf.org/v2.5">
  <rdf:Description about="http://www.w3.org/PICS/Overview.html" bagID="L01"
      gcf:suds="0.5"
      gcf:density="0"
      gcf:color.hue="1"/>

  <rdf:Description about="http://www.w3.org/PICS/Underview.html" bagID="L02"
    gcf:subject="2"
    gcf:density="1"
    gcf:color.hue="1"/>

  <rdf:Description aboutEach="#L01"
    pics:by="John Doe"
    pics:on="1994.11.05T08:15-0500"
    pics:until="1995.12.31T23:59-0000"/>

  <rdf:Description aboutEach="#L02"
    pics:by="Jane Doe"
    pics:on="1994.11.05T08:15-0500"
    pics:until="1995.12.31T23:59-0000"/>
</rdf:RDF>
```

RDFS

What is the RDF Schema Specification?

- Specifies how to use RDF to describe RDF vocabularies.
- Defines a basic vocabulary.
- Defines an extensibility mechanism in anticipation of additions to RDF.

If we don't use schemata, many representations are possible

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
     syntax-ns#“ />
 <rdf:Description
     about="http://www.w3.org/Home/Lassila">
  <Creator>
   <rdf:Description
    about="http://www.w3.org/staffId/85740">
    <rdf:type
    resource="http://desc.org/schema/Person"/
    >
    <Name>Ora Lassila</Name>
    <Email>lassila@w3.org<Email>
   </rdf:Description>
  </Creator>
 </rdf:Description>
</rdf:RDF>
```

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
     syntax-ns#“ />
 <rdf:Description
     about="http://www.w3.org/Home/Lassila">
  <author>
   <rdf:Description
    about="http://www.w3.org/staffId/85740">
    <rdf:type
    resource="http://desc.org/schema/Person"/
    >
    <name>
     <surname>Lassila</surname>
     <given>Ora</given>
    </name>
    <email>lassila@w3.org</email>
   </rdf:Description>
  </author>
 </rdf:Description>
</rdf:RDF>
```

Not *another* schema scheme?

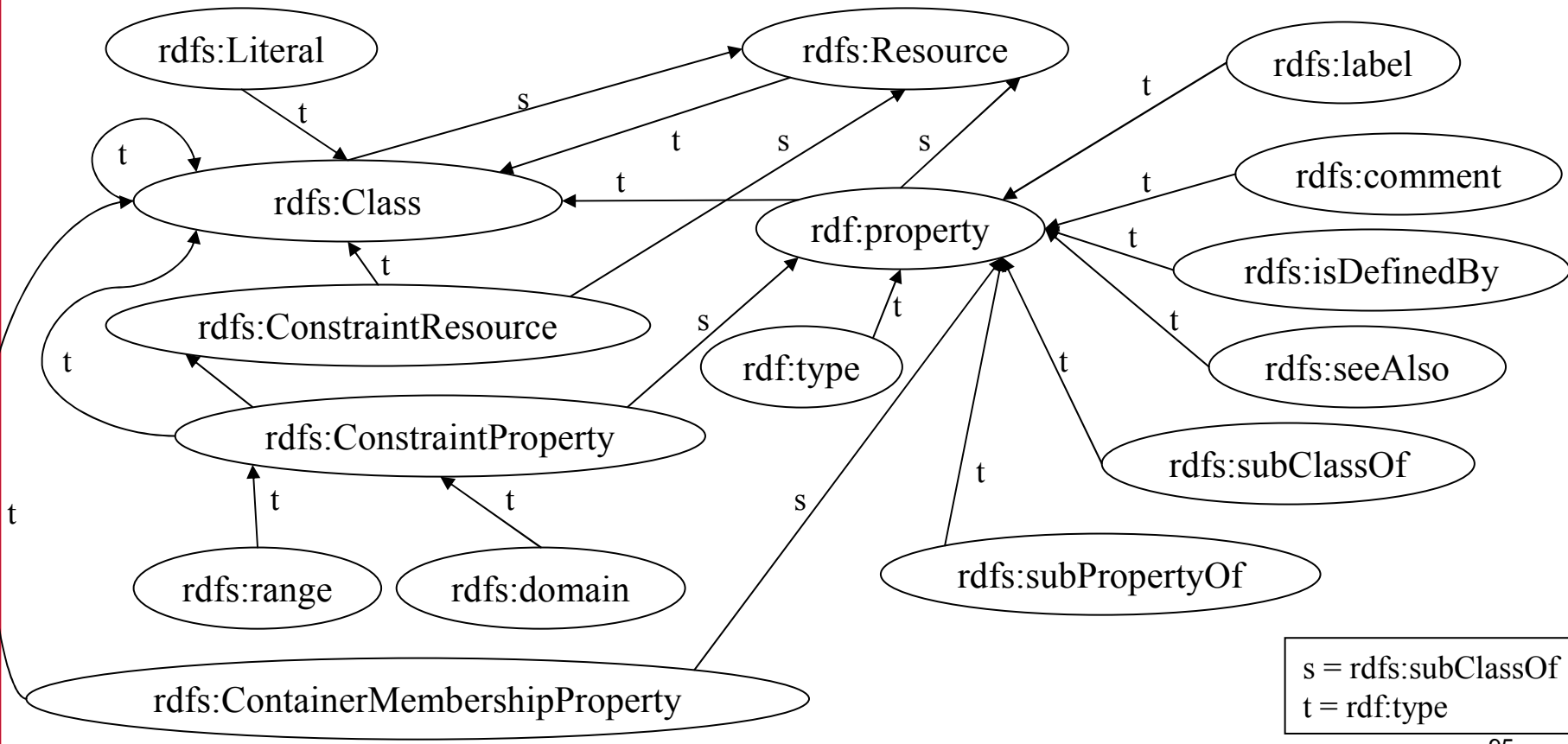
- Why another schema definition?
  Can’t we just use DTDs or XML Schema?

Not *another* schema scheme?

- Why another schema definition?
  Can't we just use DTDs or XML Schema?

  DTD and XML Schema definitions –
  - only define syntax.
  - don't have the power to describe things like class membership in a robust manner.
  - Don't refer to things outside of XML.

RDFS Class Hierarchy



Core Classes - rdfs:Resource

- Things described by RDF expressions are called resources, and are considered to be instances of the class rdfs:Resource. The RDF class rdfs:Resource represents the set called 'Resources' in the formal model.

Core Classes - rdf:Property

- This represents the subset of RDF resources that are properties, i.e., all the elements of the set called 'Properties' the formal model.

Core Classes - rdfs:Class

- This corresponds to the generic concept of a Type or Category, similar to the notion of a Class in object-oriented programming languages such as Java. When a schema defines a new class, the resource representing that class must have an rdf:type property whose value is the resource rdfs:Class. RDF classes can be defined to represent almost anything, such as Web pages, people, document types, databases or abstract concepts.

The Core Properties

Every RDF model which uses the schema mechanism also (implicitly) includes the core properties. These are instances of the rdf:Property class and provide a mechanism for expressing relationships between classes and their instances or superclasses.

Core Properties - rdf:type

This indicates that a resource is a member of a class, and thus has all the characteristics that are to be expected of a member of that class. The value of an rdf:type property for some resource is another resource which must be an instance of rdfs:Class. The resource known as rdfs:Class is itself a resource of rdf:type rdfs:Class. Individual classes (for example, 'Dog') will always have an rdf:type property whose value is rdfs:Class (or some subclass of rdfs:Class, as described in section 2.3.2). A resource may be an instance of more than one class.

Core Properties - rdfs:subClassOf

- This specifies a subset/superset relation between classes. The rdfs:subClassOf property is transitive. If class A is a subclass of some broader class B, and B is a subclass of C, then A is also implicitly a subclass of C.
- Only instances of rdfs:Class can have the rdfs:subClassOf property and the property value is always of rdf:type rdfs:Class. A class may be a subclass of more than one class.
- A class cannot be a subclass of itself, nor of any of its own subclasses.

Core Properties - rdfs:subPropertyOf

- This instance of rdf:Property specifies that one property is a specialization of another. A property may be a specialization of zero, one or more properties. If some property P2 is a subPropertyOf another more general property P1, and if a resource A has a P2 property with a value B, this implies that the resource A also has a P1 property with value B.
- A property can never be declared to be a subproperty of itself, nor of any of its own subproperties.

Core Properties - rdfs:seeAlso

- This specifies a resource that might provide additional information about the subject resource.
- This property may be specialized using rdfs:subPropertyOf to more precisely indicate the nature of the information the object resource has about the subject resource.
- The object and the subject resources are constrained only to be instances of the class rdfs:Resource.

Core Properties - rdfs:isDefinedBy

- This is a subproperty of rdfs:seeAlso, and indicates the resource defining the subject resource.
- As with rdf:seeAlso, this property can be applied to any instance of rdfs:Resource and may have as its value any rdfs:Resource.

Constraints -rdfs:ConstraintResource

- This resource defines a subclass of rdfs:Resource whose instances are RDF schema constructs involved in the expression of constraints. This provides a mechanism that allows RDF processors to assess their ability to use the constraint information associated with an RDF model.
- The 1.0 specification doesn't provide a mechanism for the dynamic discovery of new forms of constraint.

Constraints - rdfs:ConstraintProperty

- This resource defines a subclass of rdf:Property, all of whose instances are properties used to specify constraints.
- This class is a subclass of rdfs:ConstraintResource and corresponds to the subset of that class representing properties.
- Both rdfs:domain and rdfs:range are instances of rdfs:ConstraintProperty.

Constraints - rdfs:range

- An instance of ConstraintProperty that is used to indicate the class(es) that the values of a property must be members of. The value of a range property is always a Class. Range constraints are only applied to properties.
- A property can have at most one range property. It is possible for it to have no range, in which case the class of the property value is unconstrained.

Constraints - rdfs:domain

- This is an instance of ConstraintProperty that is used to indicate the class(es) on whose members a property can be used.
- If a property has no domain property, it may be used with any resource. If it has exactly one domain property, it may only be used on instances of that class (which is the value of the domain property). If it has more than one domain property, the constrained property can be used with instances of any of those classes.

Constraints on ranges and domains

- The rdfs:domain of rdfs:range is the class rdf:Property. This indicates that the range property applies to resources that are themselves properties.
- The rdfs:range of rdfs:range is the class rdfs:Class. This indicates that any resource that is the value of a range property will be a class.
- The rdfs:domain of rdfs:domain is the class rdf:Property. This indicates that the domain property is used on resources that are properties.
- The rdfs:range of rdfs:domain is the class rdfs:Class. This indicates that any resource that is the value of a domain property will be a class.

Documentation Tags

These support simple documentation and user-interface related annotations. Multilingual documentation of schemas is supported at the syntactic level through use of the xml:lang tagging facility.

rdfs:comment

- This is used to provide a human-readable description of a resource.

rdfs:label

- This is used to provide a human-readable version of a resource name.

Container Membership

rdfs:ContainerMembershipProperty

- This class has as members the properties _1, _2, _3 ... used to indicate container membership, as described in the formal model.

Other Things

rdfs:Literal - This corresponds to the set called the 'Literals' in the formal model.

rdf:Statement - This corresponds to the set called the 'Statement' in the formal model.

rdf:subject - This corresponds to the property called the 'subject' in the formal model. Its rdfs:domain is rdf:Statement and rdfs:range is rdfs:Resource.

rdf:predicate - This corresponds to the property called the 'predicate' in the formal model. Its rdfs:domain is rdf:Statement and rdfs:range is rdf:Property. This is used to identify the property used in the modeled statement.

Other Things

rdfs:Container - This class is used to represent the Container classes described in the model. It is an instance of rdfs:Class and rdfs:subClassOf of rdfs:Resource.

rdf:Bag - This corresponds to the class called 'Bag' in the formal model. It is an instance of rdfs:Class and rdfs:subClassOf rdfs:Container.

rdf:Seq - This corresponds to the class called 'Sequence' in the formal model. It is an instance of rdfs:Class and rdfs:subClassOf rdfs:Container.

rdf:Alt - This corresponds to the class called 'Alternative' in the formal model. It is an instance of rdfs:Class and rdfs:subClassOf rdfs:Container.

Other Things

rdf:object - This corresponds to the property called the 'object' in the formal model. Its rdfs:domain is rdf:Statement. This is used to identify the property value in the modeled statement

rdf:value - This corresponds to the 'value' property described in the specification.

Example

```
<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#">
    <rdfs:Class rdf:ID="Person">
          <rdfs:comment>The class of people.</rdfs:comment>
          <rdfs:subClassOf
          rdf:resource="http://www.w3.org/2000/03/example/classes#Animal"/>
    </rdfs:Class>
    <rdf:Property ID="maritalStatus">
          <rdfs:range rdf:resource="#MaritalStatus"/>
          <rdfs:domain rdf:resource="#Person"/>
    </rdf:Property>
```

Example'

```
<rdf:Property ID="ssn">
        <rdfs:comment>Social Security Number</rdfs:comment>
        <rdfs:range

        rdf:resource="http://www.w3.org/2000/03/example/classes#Integer"
/>
        <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property ID="age">
        <rdfs:range
        rdf:resource="http://www.w3.org/2000/03/example/classes#Integer"
/>
        <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
```

Example ’’

```
    <rdfs:Class rdf:ID="MaritalStatus"/>
    <MaritalStatus rdf:ID="Married"/>
    <MaritalStatus rdf:ID="Divorced"/>
    <MaritalStatus rdf:ID="Single"/>
    <MaritalStatus rdf:ID="Widowed"/>
</rdf:RDF>
```

Beyond RDF: OIL & DAML

- OIL extends RDF Schema to a fully-fledged knowledge representation language.
  - logical expressions
  - data-typing
  - cardinality
  - quantifiers
  - http://www.ontoknowledge.org
- DAML = US sister of OIL
- Merged as DAML+OIL in 2001
- Becomes OWL W3C standard in March '03

DAML + OIL

DAML Objectives

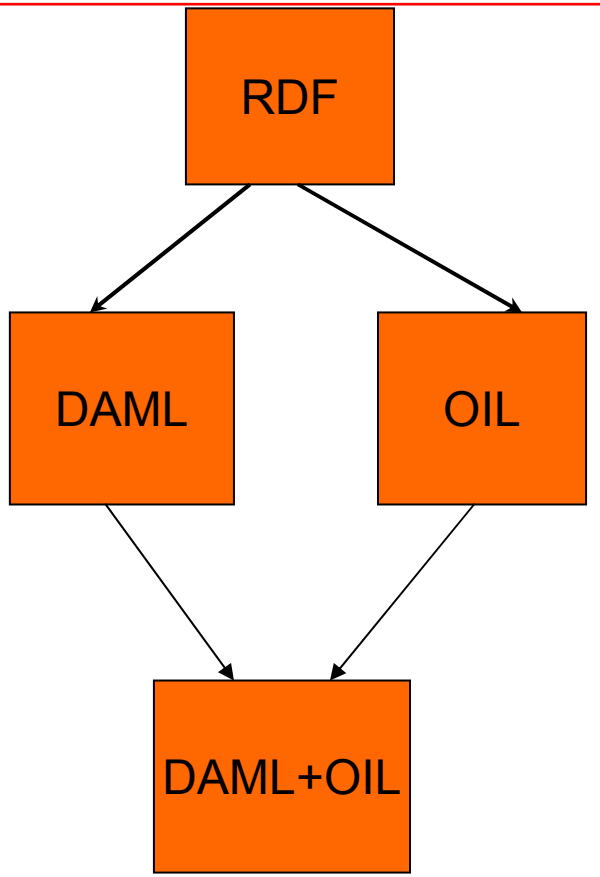
- support World Wide Web (WWW) content that is easily used by intelligent agents and other programs
- enable the Semantic Web

Approach: Languages and Tools

- leverage existing work
  - W3C Resource Description Framework (RDF & RDFS)
  - European Ontology Inference Layer (OIL)
  - UMaryland Simple HTML Ontology Extensions (SHOE)
- work with W3C to define languages (standards transition)
- work openly (www.daml.org)
- develop prototype tools to span the DAML lifecycle
  - ontology creation and editing, translation and mapping
  - distributed knowledge bases markup editors
  - validation services
  - DAML APIs
  - DAML-aware browsers
  - applications

Layer

- RDF
- RDF Schema
- DARPA Agent Markup Language (DAML)
- Ontology Inference Layer (OIL)
- DAML+OIL



DAML+OIL

- Joint Committee
  - US DAML and EU Semantic Web Technologies participants
- released March 2001

DAML+OIL Language Features

- grounded in the WWW (URI naming, XML syntax, HTTP) -- strong implications for diversity, scalability, security, etc.
- classes and instances
- class construction primitives (intersection, union, complement)
- fine grained property constraints
- cardinality constraints
- formal semantics: model-theoretic and axiomatic –> enables inferencing

Language Feature Comparison

	XML DTD	XML Schema	RDF(S)	DAML+ OIL	RDF(S) 2002
bounded lists				X	X
cardinality constraints	X	X		X	
class expressions				X	
data types		X		X	?
defined classes				X	
enumerations	X	X		X	
equivalence				X	
extensibility			X	X	X
formal semantics				X	X
inheritance			X	X	X
inference				X	
local restrictions				X	
qualified constraints				X	
reification			X	X	X

Why DAML+OIL or RDF Schema?

- Similar to RDFS in providing infrastructure to allow machines to make inferences

  Given

  ```
  (motherOf subProperty parentOf)
  (Mary motherOf Bill)
  ```

  when stated in DAML, allows you to conclude

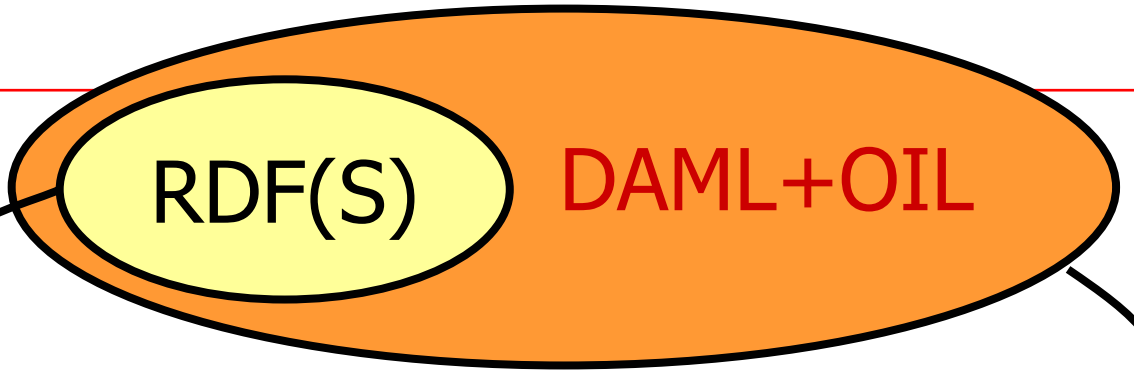
  ```
  (Mary parentOf Bill)
  ```

- RDFS is limited in expressiveness that DAML+OIL addresses

DAML+OIL as RDF(S) extension

```
<rdfs:Class rdf:ID="herbivore">
  <rdf:type
      rdf:resource="http://www.ontoknowledge.org/#DefinedClass"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
  <rdfs:subClassOf>



  </rdfs:subClassOf>
</rdfs:Class>
```

DAML+OIL as RDFS extension



- class-def
- subclass-of
- slot-def
- subslot-of
- domain
- range

- class-expressions
  - AND, OR, NOT
- slot-constraints
  - has-value, value-type
  - cardinality
- slot-properties
  - trans, symm

DAML+OIL: Classes

```
<daml:Class rdf:ID="Male">
     <rdfs:subClassOf rdf:resource="#Animal"/>
</daml:Class>

<daml:Class rdf:ID="Female">
     <rdfs:subClassOf rdf:resource="#Animal"/>
     <daml:disjointWith rdf:resource="#Male"/>
</daml:Class>

<daml:Class rdf:ID="Man">
  ^!  <rdfs:subClassOf rdf:resource="#Person"/>
     <rdfs:subClassOf rdf:resource="#Male"/>
</daml:Class>
```

Class-Building Operations

- Relation to other Classes
  - **`rdfs:subClassOf`**
  - **`daml:disjointWith`**
  - **`daml:disjointUnionOf`**
  - **`daml:sameClassAs`**
  - **`daml:equivalentTo`**
- Contained Elements:
  - **`daml:oneOf`**
- Boolean combinations:
  - **`daml:intersectionOf`**
  - **`daml:unionOf`**
  - **`daml:complementOf`**

DAML+OIL: Properties

```
<daml:ObjectProperty rdf:ID="hasParent">
     <rdfs:domain rdf:resource="#Animal"/>
     <rdfs:range rdf:resource="#Animal"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="hasChild">
     <daml:inverseOf
          rdf:resource="#hasParent"/>
</daml:ObjectProperty>

<daml:UniqueProperty rdf:ID="hasMother">
     <rdfs:subPropertyOf
          rdf:resource="#hasParent"/>
     <rdfs:range rdf:resource="#Female"/>
</daml:UniqueProperty>
```

Property-Building Operations

- Basic Types
  - `daml:ObjectProperty`
  - `daml:DataTypeProperty`
- Special Types
  - `daml:TransitiveProperty`
  - `daml:UniqueProperty`
  - `daml:UnambigousProperty`
- Further Restrictions
  - `rdfs:subPropertyOf`
  - `rdfs:domain`
  - `rdfs:range`
  - `daml:samePropertyAs`
  - `daml:inverseOf`

DAML+OIL: Property Restrictions

```
<daml:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty
          rdf:resource="#hasFather"/>
      <daml:toClass rdf:resource="#Man"/>
    </daml:Restriction>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty
          rdf:resource="#hasFather"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

DAML+OIL: Property Restrictions

```
<daml:Class rdf:ID="Person">
   <rdfs:subClassOf rdf:resource="#Animal"/>
   <rdfs:subClassOf>
      <daml:Restriction daml:cardinalityQ="1">
        <daml:onProperty
            rdf:resource="#hasFather"/>
        <daml:hasClassQ rdf:resource="#Man"/>
      </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>
```

Restrictions

- General
  - daml:Restriction
  - daml:onProperty
- Number Restrictions
  - daml:cardinality
  - daml:maxCardinality
  - daml:minCardinality
- Value and Type Restrictions
  - daml:toClass
  - daml:hasValue
  - daml:hasClass
- Combinations
  - daml:cardinalityQ
  - daml:maxCardinalityQ
  - daml:minCardinalityQ

DAML+OIL: Individuals

```
<rdf:Description rdf:ID="Asia">
   <rdf:type>
      <rdfs:Class rdf:about="#continent"/>
   </rdf:type>
   <daml:differnetIndividualFrom
      rdf:resource="#Europe"/>
</rdf:Description>

<continent rdf:ID="Asia"/>

<rdf:Description rdf:ID="India">
   <is_part_of rdf:resource="#Asia"/>
   <daml:sameIndividual
      rdf:resource="#IndianSubcontinent"/>
   <inhabitants>
      <xsd:integer rdf:value="700.000.000"/>
   </inhabitants>
</rdf:Description>
```

DAML+OIL: Defined Datatypes

```
<xsd:simpleType name="over17">
   <xsd:restriction
      base="xsd:positiveInteger">
      <xsd:minInclusive value="18"/>
   </xsd:restriction>
</xsd:simpleType>

<daml:Class rdf:ID="Adult">
   <daml:intersectionOf
      rdf:parseType="daml:collection">
      <daml:Class rdf:about="#Person"/>
      <daml:Restriction>
         <daml:onProperty
            rdf:resource="#age"/>
         <daml:hasClass
            rdf:resource="#over17"/>
      </daml:Restriction>
   </daml:intersectionOf>
</daml:Class>
```

Available Tools

- DAML Ontology Library: clearinghouse
- Stanford RDF API
- OilEd: ontology editor
- Chimaera: ontology analyzer
- DAML Validator (under development)
- DAML Crawler: spider
- DAML Viewer, PalmDAML, and HyperDAML: navigator GUIs

DAML Languages: Future Work

- incorporate feedback
- proposal(s) to new W3C Semantic Web Activity
- rules (working with RuleML Initiative)
- logic, proof-checking
- ...

WebOnt and OntoWeb

- W3C WebOnt working group Nov 2001–Mar 2003
  Work continuing where DAML+OIL left off
  http://www.w3.org/2001/sw/WebOnt/charter

- WebOnt is part of W3C Semantic Web activity which also includes RDF

- OntoWeb
  EU funded thematic network
  > 80 partners, including CWI, UvA and VU
  http://www.ontoweb.org

DAML+OIL ↔ RDF

- DAML+OIL ontology is a set of RDF statements
  - DAML+OIL defines semantics for certain statements
  - Does **NOT** restrict what can be said

    Ontology can include arbitrary RDF
  - But no semantics for non-DAML+OIL statements
- Adds capabilities common to description logics:
  - cardinality constraints, defined classes (=> classification), equivalence, local restrictions, disjoint classes, etc.
- More support for ontologies
  - Ontology imports ontology
- But not (yet) variables, quantification, and general rules

DAML-S

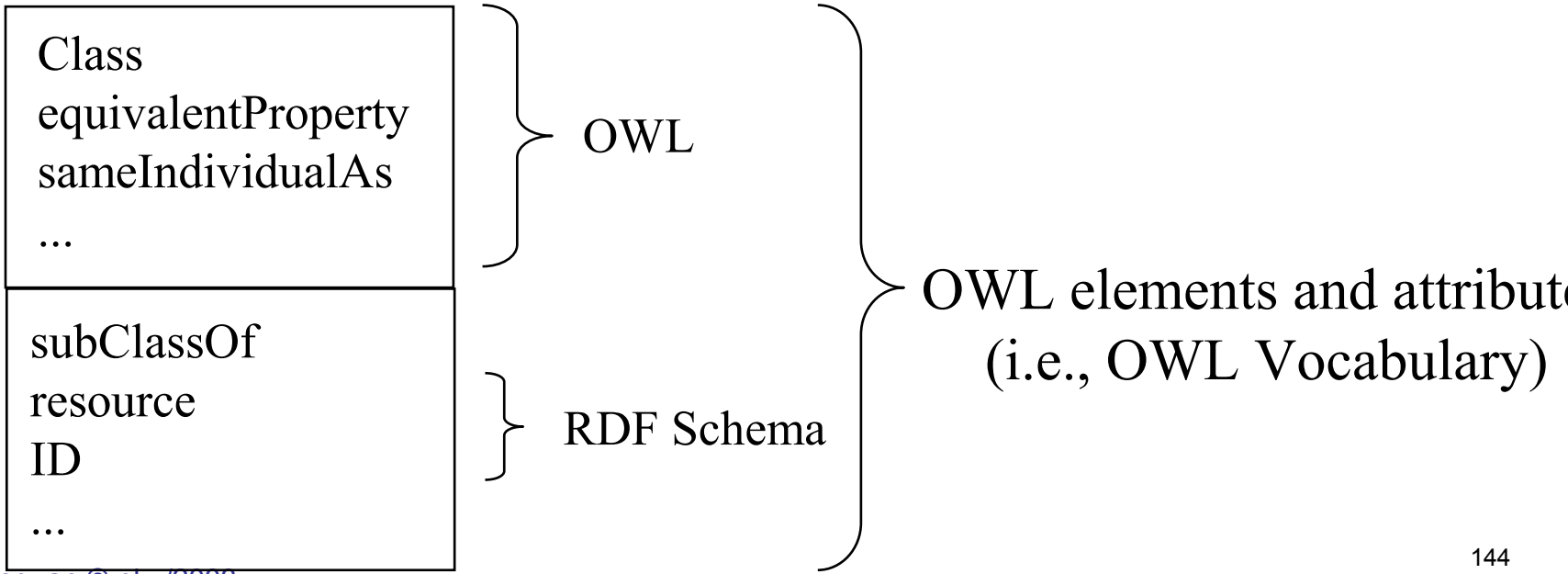
- DAML-S is an ontology for describing properties and capabilities of web services
- Desiderata:
  - Ease of expressiveness
  - Enables automation of service use by agents
  - Enables reasoning about service properties and capabilities
- Also appropriate for describing services in a mobile/pervasive computing environment
- See http://daml.org/services/

OWL

What is OWL?

OWL is a set of XML elements and attributes,with standardized meaning, that are used to define terms and their relationships.

OWL extends RDF Schema:



Example of using OWL to define two terms and their relationship

Example: Define the terms "Camera" and "SLR". State that SLRs are a type of Camera.

Here's how these two terms (classes) and their relationship is defined using the OWL vocabulary:

```
<owl:Class rdf:ID="Camera"/>
```

```
<owl:Class rdf:ID="SLR">
    <rdfs:subClassOf rdf:resource="#Camera"/>
</owl:Class>
```

Example: Bridging the Terminology Gap using OWL

- A key problem in achieving interoperability is to be able to recognize that two pieces of data are talking about the same thing, even though different terminology is being used.
- The following slides presents an example to show how OWL may be used to bridge the "terminology gap".

Interested in Purchasing a Camera

- Scenario:
  - I am interested in purchasing a camera with a 75-300mm zoom lens size, that has an aperture of 4.5-5.6, and a shutter speed that ranges from 1/500 sec. to 1.0 sec.
  - I launch my personal "Web Bot" which crawls the Web looking for Web sites that can fulfill my request.
  - Assume that there exists an OWL Camera Ontology, which the Web Bot can "consult" upon its travels across the Web.

Is this document relevant?

The Web Bot finds this document at a Web site:

```
<PhotographyStore rdf:ID="Hunts"
                  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <store-location>Malden, MA</store-location>
  <phone>617-555-1234</phone>
  <catalog rdf:parseType="Collection">
    <SLR rdf:ID="Olympus-OM-10"
         xmlns="http://www.camera.org#">
      <lens>
        <Lens>
          <focal-length>75-300mm zoom</focal-length>
          <f-stop>4.5-5.6</f-stop>
        </Lens>
      </lens>
      <body>
        <Body>
          <shutter-speed rdf:parseType="Resource">
            <min>0.002</min>
            <max>1.0</max>
            <units>seconds</units>
          </shutter-speed>
        </Body>
      </body>
      <cost rdf:parseType="Resource">
        <rdf:value>325</rdf:value>
        <currency>USD</currency>
      </cost>
    </SLR>
  </catalog>
</PhotographyStore>
```

Is it relevant?

(Note: SLR = Single Lens Reflex)

A Match?

```
<PhotographyStore rdf:ID="Hunts"
                  xmlns:rdf="&rdf;#">
  <store-location>Malden, MA</store-location>
  <phone>617-555-1234</phone>
  <catalog rdf:parseType="Collection">
    <SLR rdf:ID="Olympus-OM-10"
         xmlns="http://www.camera.org#">
      <lens>
        <Lens>
          <focal-length>75-300mm zoom</focal-length>
          <f-stop>4.5-5.6</f-stop>
        </Lens>
      </lens>
      <body>
        <Body>
          <shutter-speed rdf:parseType="Resource">
            <min>0.002</min>
            <max>1.0</max>
            <units>seconds</units>
          </shutter-speed>
        </Body>
      </body>
      <cost rdf:parseType="Resource">
        <rdf:value>325</rdf:value>
        <currency>USD</currency>
      </cost>
    </SLR>
  </catalog>
</PhotographyStore>
```

← Match? →

I am interested in purchasing a camera with a 75-300mm zoom lens size, that has an aperture of 4.5-5.6, and a shutter speed that ranges from 1/500 sec. to 1.0 sec.

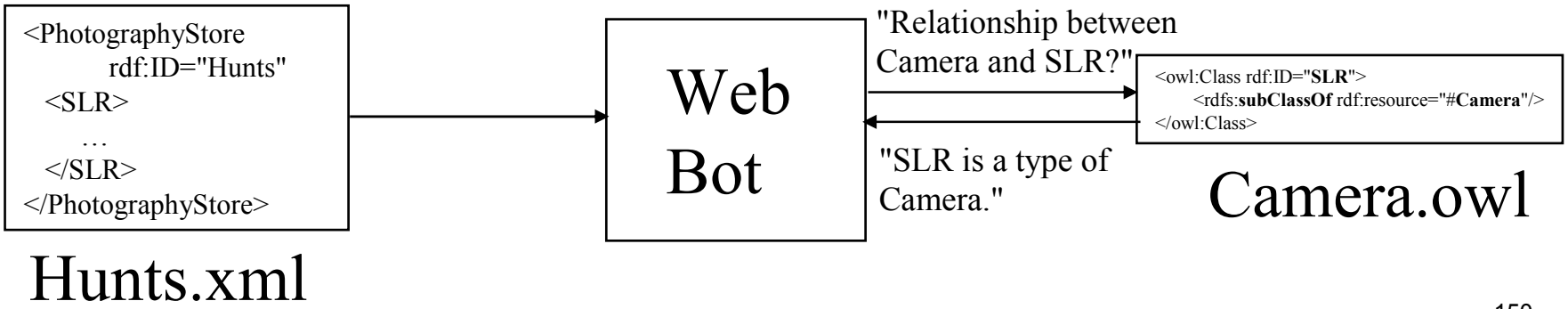
To determine if there is a match, these questions must be answered:

1. What's the relationship between "SLR" and "Camera"?
2. What's the relationship between "focal-length" and "size"?
3. What's the relationship between "f-stop" and "aperture"?

Relationship between SLR and Camera?

The Web Bot "consults" the OWL Camera Ontology. This OWL statement tells the Web Bot that a SLR is a type of Camera

```
<owl:Class rdf:ID="SLR">
    <rdfs:subClassOf rdf:resource="#Camera"/>
</owl:Class>
```



Relationship between focal-length and lens size?

This OWL statement tells the Web Bot that focal-length is equivalent to lens size:

```
<owl:DatatypeProperty rdf:ID="focal-length">
        <owl:equivalentProperty rdf:resource="#size"/>
        <rdfs:domain rdf:resource="#Lens"/>
        <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

"focal-length is synonymous with (lens) size.
focal-length is to be used within a Lens.
focal-length has a value that is a string."

Relationship between f-stop and aperture?

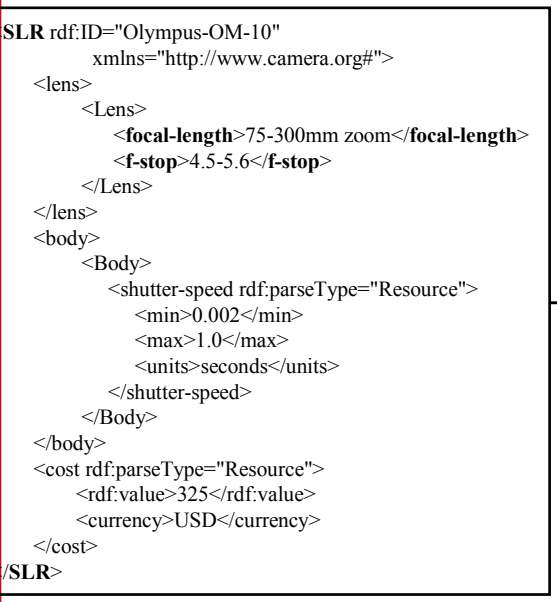
This OWL statement tells the Web Bot that f-stop is equivalent to aperture:

```
<owl:DatatypeProperty rdf:ID="f-stop">
        <owl:equivalentProperty rdf:resource="#aperture"/>
        <rdfs:domain rdf:resource="#Lens"/>
        <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

The Web Bot now recognizes that the XML document it found at the Web site
- is talking about Cameras, and it
- does show the lens size, and it
- does show the aperture for the camera, and
- the values for lens size, aperture, and shutter speed are met.

Thus the Web Bot recognizes that the XML document is a match!

Semantic Definitions Separate from Application!



Hunts.xml

Web Bot
(application)

"Relationship between Camera and SLR?"

"SLR is a type of Camera."

"Relationship between aperture and f-stop?"

"f-stop is synonymous with aperture."

"Relationship between size and focal-length?"

"focal-length is synonymous with size."

Semantic Definition

Camera.owl

See the article "*Why use OWL?*" for a discussion of why it is good practice to separate the semantic definitions

Summary: Interoperability despite terminology differences!

- The example demonstrated how a Web Bot was able to dynamically process an XML document from a Web site, despite the fact that the XML document used terminology different than was used to express the request. This interoperability was achieved by using the OWL Camera Ontology!
- This example also demonstrated the architectural design principle of cleanly separating the application code (e.g., Web Bot) from the semantic definitions (e.g., Camera.owl).

The OWL Camera Ontology is Online!

Here is the URL to a pictorial view of the Camera Ontology:

http://www.xfront.com/owl/ontologies/camera/sld001.htm

Here is the URL to the camera.owl document:

http://www.xfront.com/owl/ontologies/camera/camera.owl

Here are the URLs to 7 physical expressions (instance documents):

http://www.xfront.com/owl/ontologies/camera/Query1.xml
http://www.xfront.com/owl/ontologies/camera/Hunts.xml
http://www.xfront.com/owl/ontologies/camera/Query2.xml
http://www.xfront.com/owl/ontologies/camera/Hunts2.xml
http://www.xfront.com/owl/ontologies/camera/RJs.xml
http://www.xfront.com/owl/ontologies/camera/OlympusOutletStore.xml
http://www.xfront.com/owl/ontologies/camera/OlympusCorp.xml

OWL Goals

- The WOWG has identified the following goals in developing OWL
  - Shared ontologies
  - Ontology evolution
  - Ontology interoperability
  - Inconsistency detection
  - Balance of expressivity and scalability
  - Ease of use
  - XML syntax
  - Internationalization

OWL status and publications

- Current plan is to have three compliance levels: OWL lite, OWL, OWL plus
- WebOnt has published
  - Requirements for a Web Ontology Language
  - Feature Synopsis for OWL Lite and OWL
  - OWL Web Ontology Language 1.0 Reference
  - OWL Web Ontology Language 1.0 Abstract Syntax
- OWL is roughly equivalent to DAML with some renaming of properties
  - (forthcoming) OWL Guide

Readings

- See the course homepage