Semantic Web Basics (cont.)

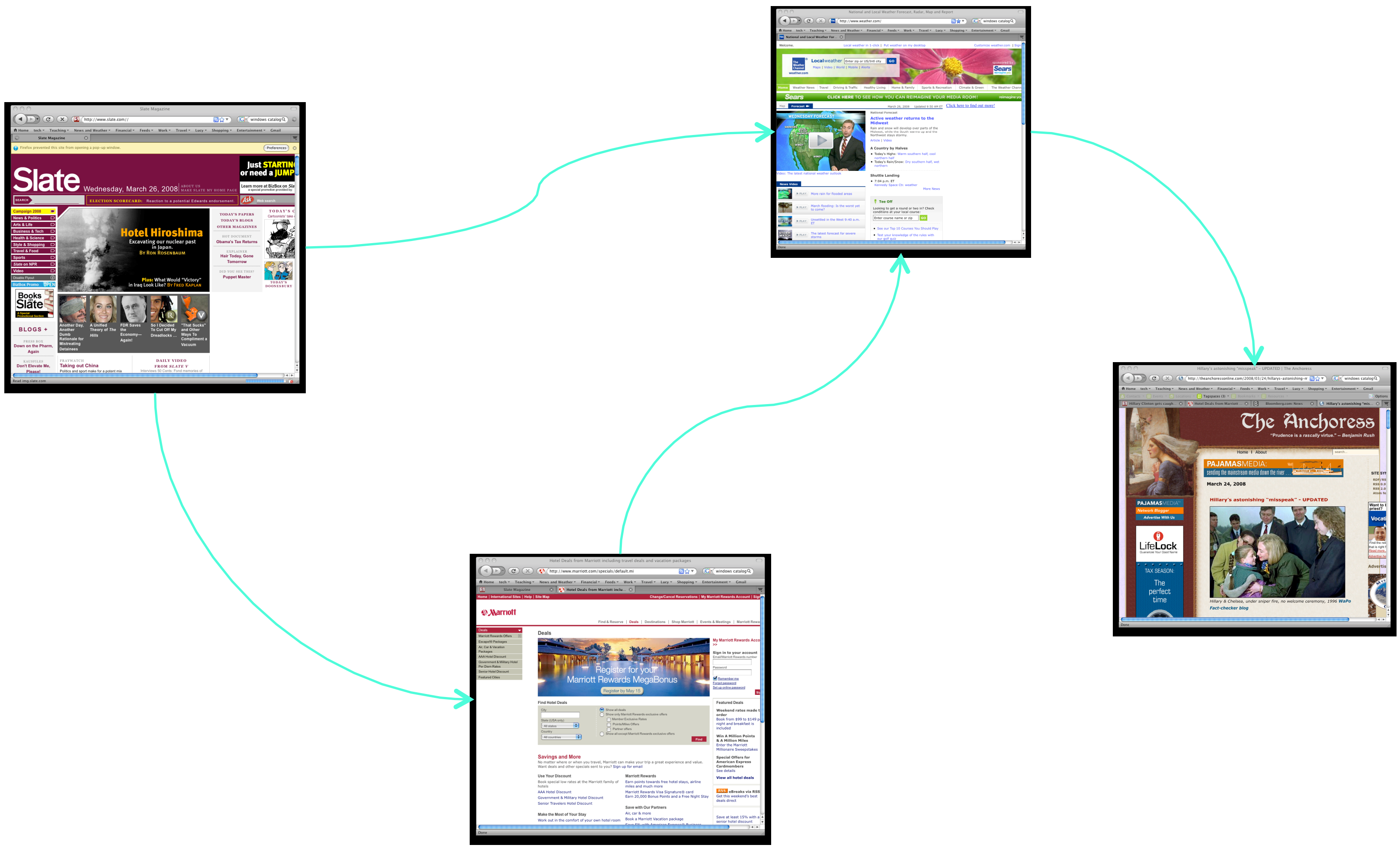
CS 431 – March 26, 2008

Carl Lagoze – Cornell University

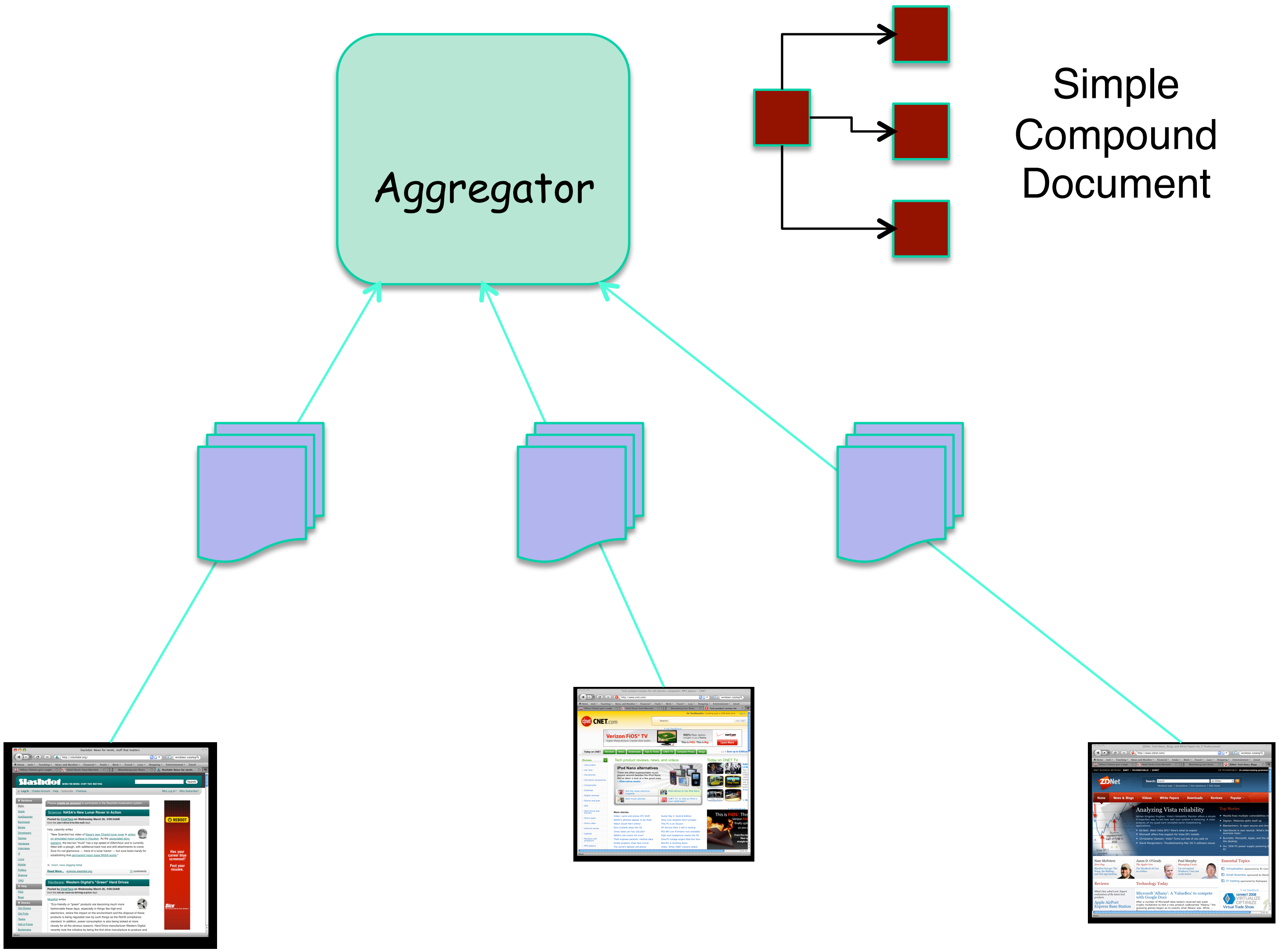
Acknowledgements for various slides and ideas

- Ian Horrocks (Manchester U.K.)
- Eric Miller (W3C)
- Dieter Fensel (Berlin)
- Volker Haarslev (Montreal)

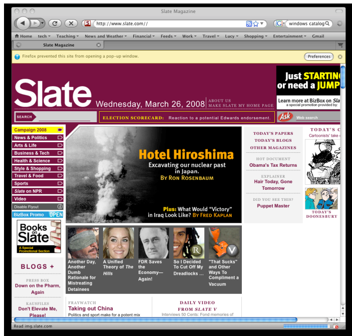
We started with the Presentation Web



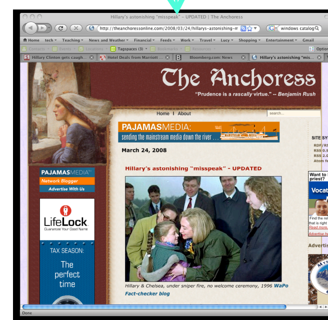
Atom/RSS gave us data extraction from the Web

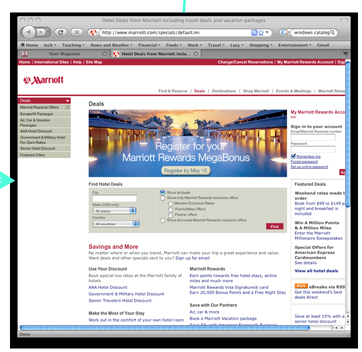


But we want to do more - Beyond just web pages



Events

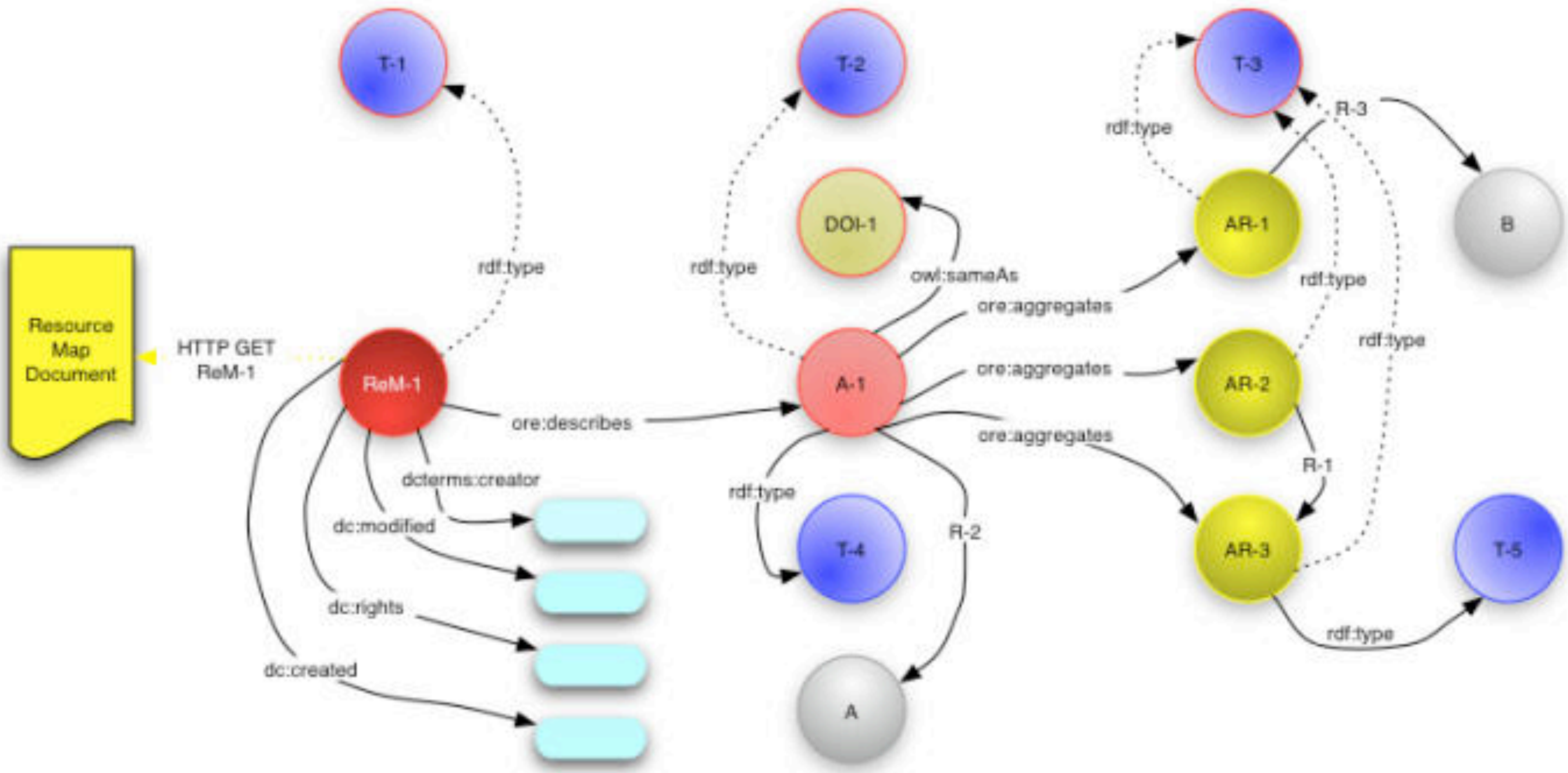




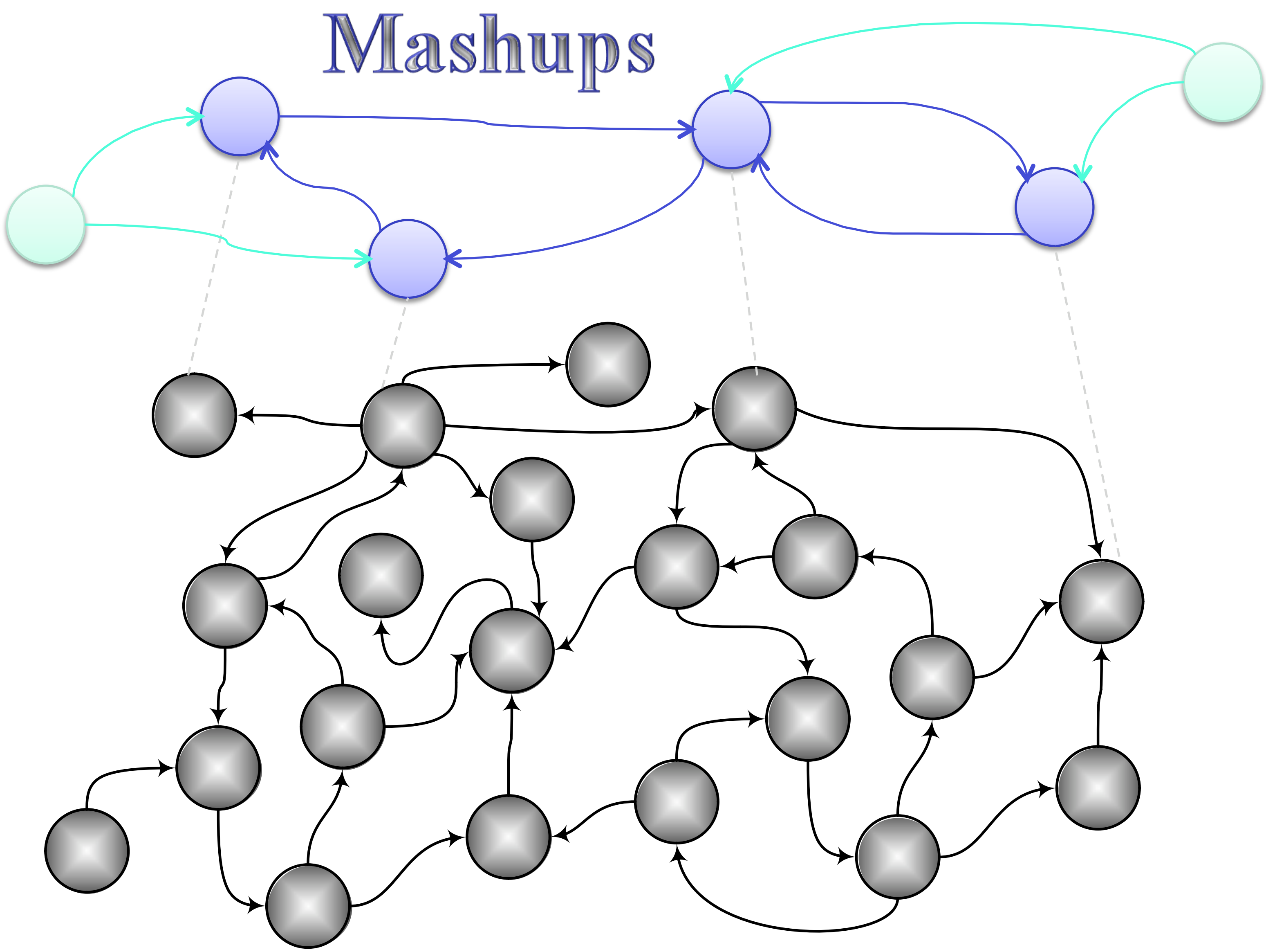
People

Events

We want to do more – Compound/Complex Relationships



Motivating the problem: Integrating Web Resources in new ways



Standards/mechanisms for doing this

- Stuff we've learned so far
  - URIs – keys for unique identity and joining distributed information
  - XML – Markup for serialization of knowledge bases
  - Namespaces – URIs for vocabulary terms
- Stuff we'll learn from here
  - RDF – basic model for representing knowledge via binary relationships
  - Ontologies – definitions of vocabulary terms and their relationships
  - OWL – RDF-based model for expressing Ontologies
  - Description logic – Formal way to represent ontologies and reason with them

Assertions are statements

- Resource1 “is about” Resource2
- Resource1 “annotates” Resource2
- Resource1 “illustrates” Resource2
- Organization1 “owns” Resource2
- Person1 “recommends” Resource2

- RDF is a model for making assertions
  - Subject → Predicate → Object

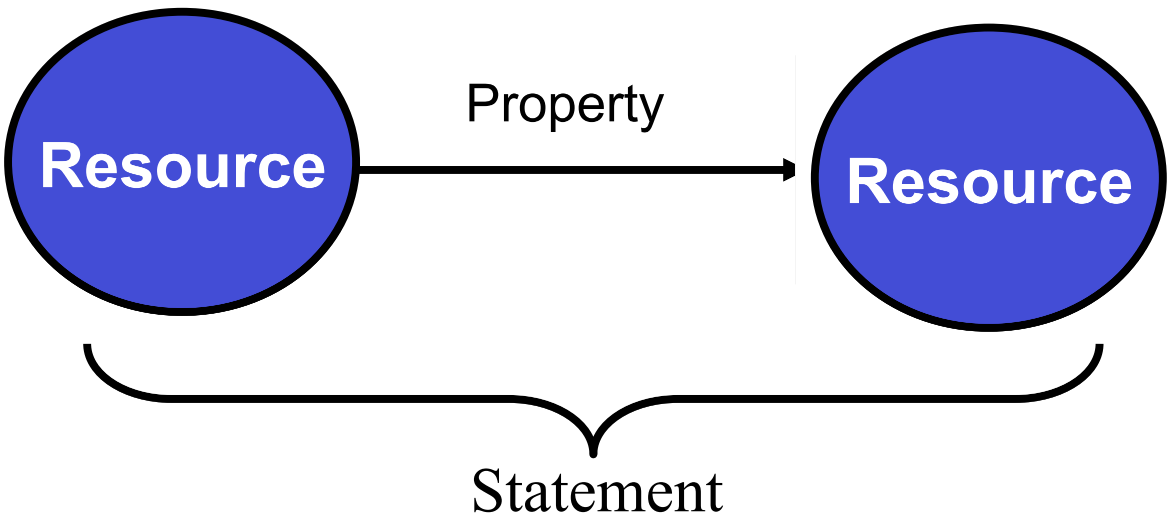
RDF Data Model

- Directed Graph expressing typed binary relations between typed resources
- Relations are:
  - P(S,O) or (:s :p :o)
- Primitives
  - resource
  - property
  - literal
  - statement
- Other constructs
  - container
  - reification
  - collection
- URI's for everything **except** literals
  - "bnodes" are a special case, but more about that later
- Common serialization is RDF/XML

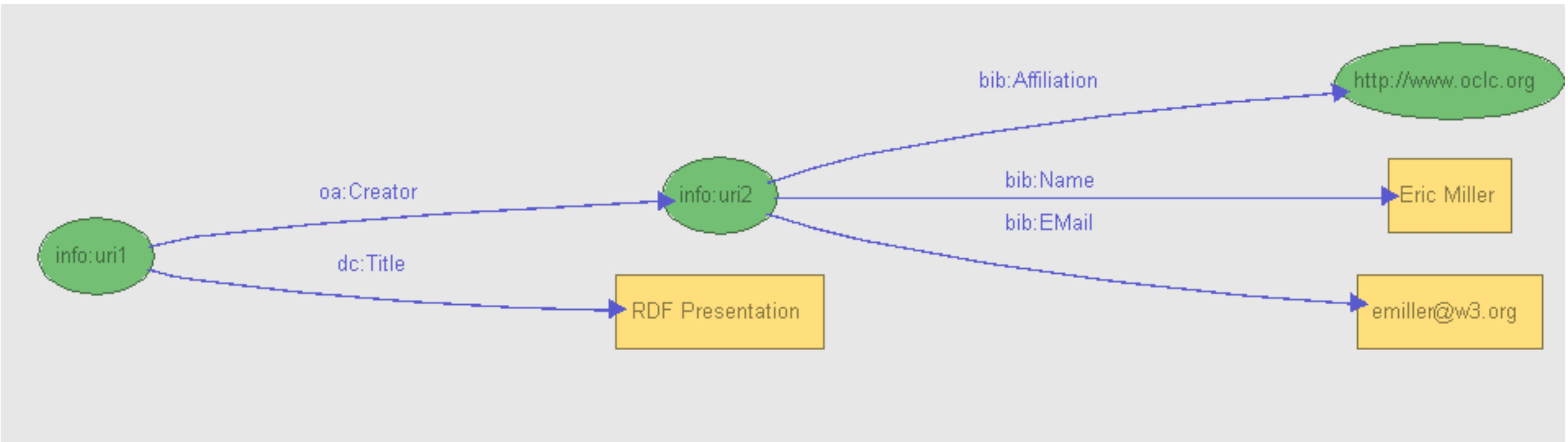
Why URIs

- Purpose of RDF is integrating information from multiple sources
  - Existing web
  - Introduced entities (people, organizations, taxonomies)
- URI's form basis of joins of graph
- Instance data combines into larger graphs
- Inferences can be made based on:
  - RDF primitives
  - Ontology definitions
    - RDFs
    - OWL

RDF Model Primitives



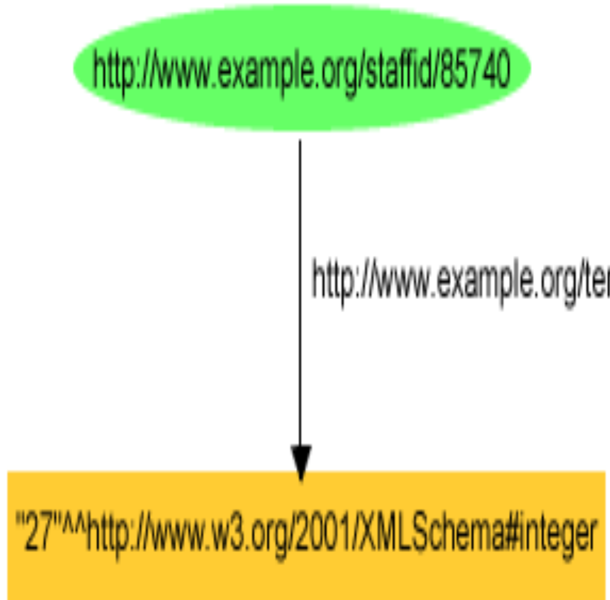
RDF Model Example #2



RDF/XML Syntax Example #2

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:gss="http://www.w3.org/2001/11/IsaViz/graphstylesheets#"
 xmlns:oa="http://agents.org/elements#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:bib="http://www.bib.org/persons#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:dc="http://purl.org/dc/elements/1.0/"
 xml:base="file:/C:/IsaViz/tmp/tmp41406.rdf">
 <rdf:Description rdf:about="info:uri2">
  <bib:Affiliation rdf:resource="http://www.oclc.org"/>
  <bib:EMail>emiller@w3.org</bib:EMail>
  <bib:Name>Eric Miller</bib:Name>
 </rdf:Description>
 <rdf:Description rdf:about="info:uri1">
  <oa:Creator rdf:resource="info:uri2"/>
  <dc:Title>RDF Presentation</dc:Title>
 </rdf:Description>
</rdf:RDF>
```

Typed Literals

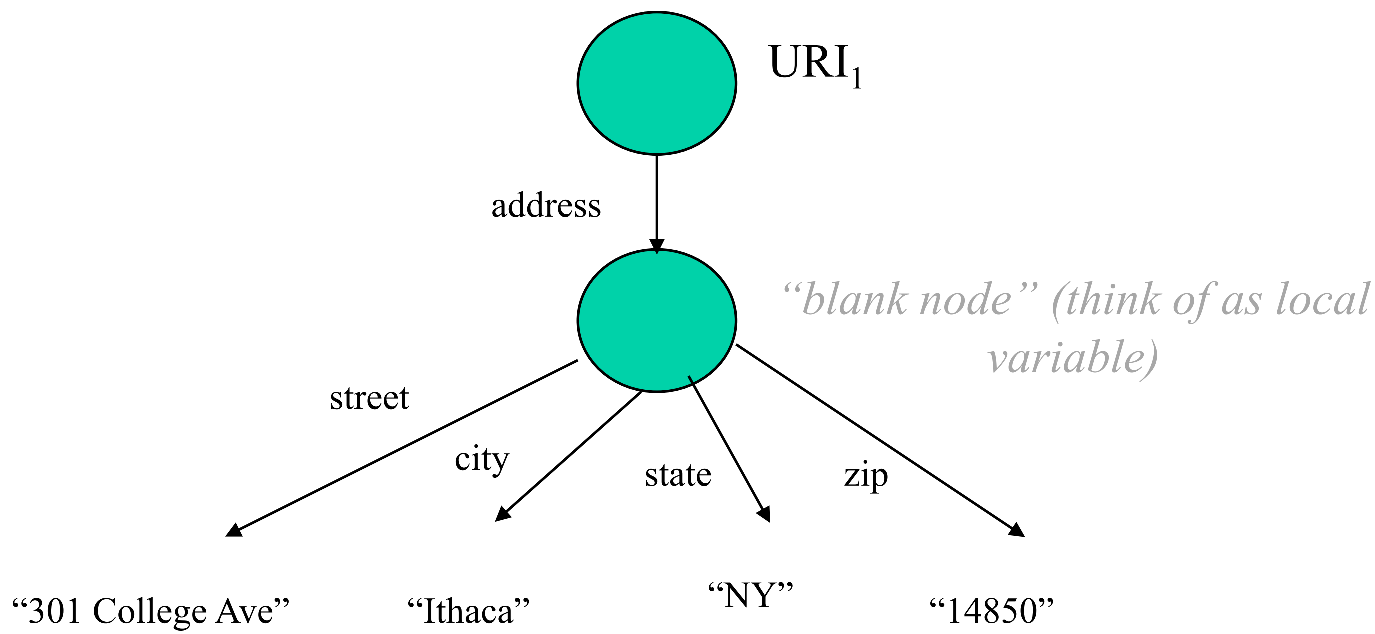



```
  <?xml version="1.0" ?>
- <rdf:RDF xmlns:gss="http://www.w3.org/2001/11/IsaViz/graphstylesheets#"
    xmlns:core="http://www.example.org/terms/"
    xmlns:s="http://example.org/students/vocab#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ex="http://example.org/terms/"
    xml:base="file:/C:/cygwin/tmp/tmp2978.rdf">
  - <rdf:Description rdf:about="http://www.example.org/staffid/85740">
      <core:age
        rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">27</core:age>
    </rdf:Description>
  </rdf:RDF>
```

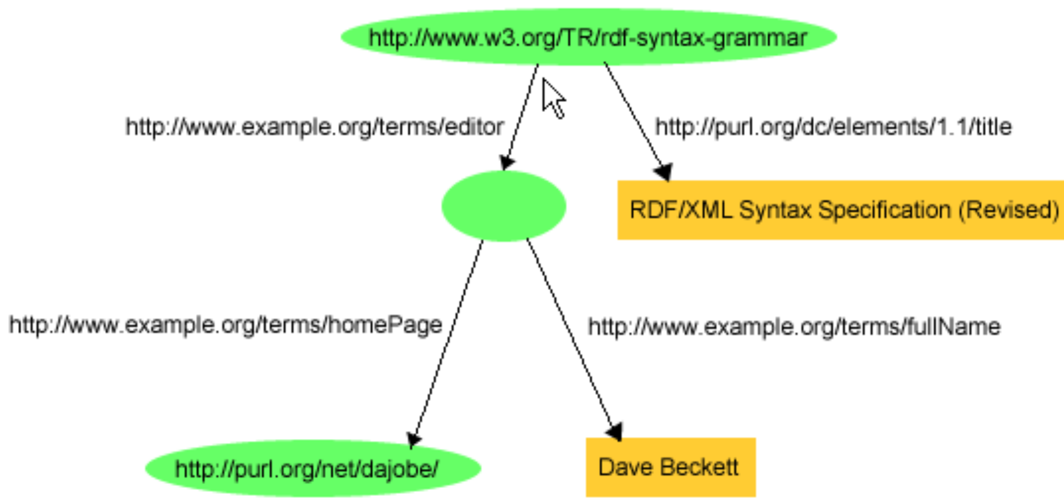
Beyond binary relations

- Note mapping of RDF statements to binary relations that could be stored in a database:
  - (:s :p :o) maps to P(S,O) – e.g., Title(R, “War & Peace”)
- But the world is more complex and statements are arbitrary n-tuples
  - Carl Lagoze has his office at 301 College Ave., Ithaca, NY 14850
  - (“Carl Lagoze” “hasOffice” “301 College Ave, Ithaca, NY 14850”)
  - (“Carl Lagoze” “address” “301 College Ave” “Ithaca” “NY” “14850”)

Expressing n-ary relations with blank nodes



Another n-ary relation 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/1.1/"
            xmlns:exterms="http://example.org/stuff/1.0/">

   <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
     <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
     <exterms:editor rdf:nodeID="abc"/>
   </rdf:Description>

   <rdf:Description rdf:nodeID="abc">
       <exterms:fullName>Dave Beckett</exterms:fullName>
       <exterms:homePage rdf:resource="http://purl.org/net/dajobe/"/>
   </rdf:Description>

</rdf:RDF>
```

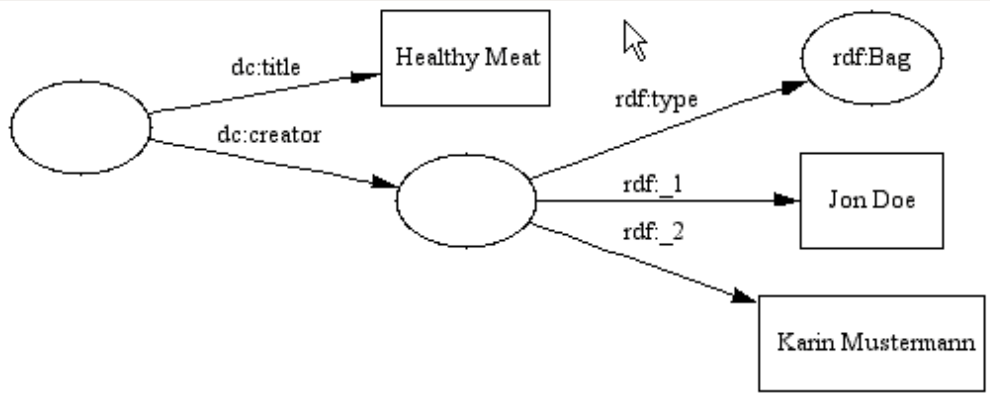
RDF Containers

- Permit the aggregation of several values for a property
- Express multiple aggregation semantics
  - unordered
  - sequential or priority order
  - alternative

RDF Containers

- Bag
  - unordered grouping
- Sequence
  - ordered grouping
- Alternatives
  - alternate values
    - need to choose
  - at least one value
  - first value is default or preferred value

Expressing Container Primitives in Binary Relations



Jon Doe and Karin Mustermann joint their forces to create a gadget with title *Healthy Meat*

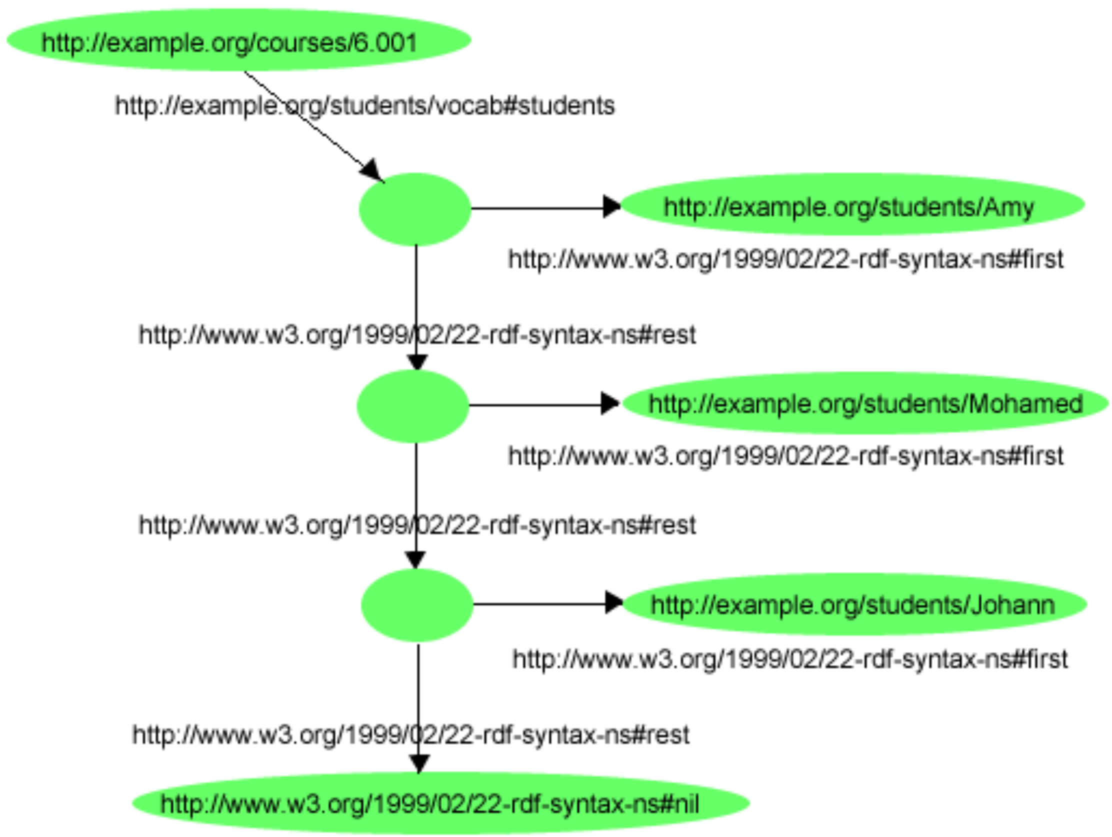
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:dcterms="http://purl.org/dc/terms/">
<rdf:Description dc:title="Healthy Meat">
 <dc:creator>
   <rdf:Bag>
     <rdf:li>Jon Doe</rdf:li>
     <rdf:li>Karin Mustermann</rdf:li>
   </rdf:Bag>
 </dc:creator>
</rdf:Description>
</rdf:RDF>
```

RDF Collections

- Containers are not closed
  - open world assumption in all of them
- Collections use lisp-like primitives (first, rest, nil) to express a close list.

RDF Collections

The students in course 6.001 are Amy, Mohamed, and Johann



Looking behind the curtain: RDF Meta-model

RDF Meta-Model provides base level for **inferences**

- Given a set of facts…
- Derive additional facts

- Some facts
  - Sam has a Prius
  - A Prius is a car
  - A Car is a type of vehicle
  - Sam has a bicycle
  - A bicycle is a type of vehicle

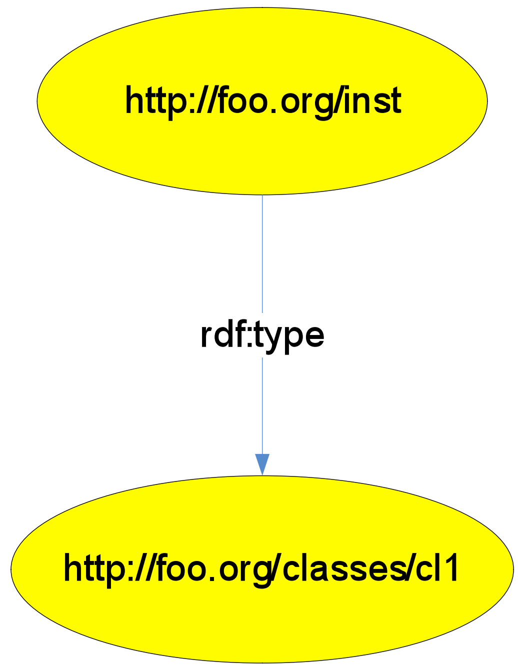
- Inference by subsumption: Sam has two vehicles
- Inference by human judgment: Sam is an environmentalist.

RDF meta-model basic elements

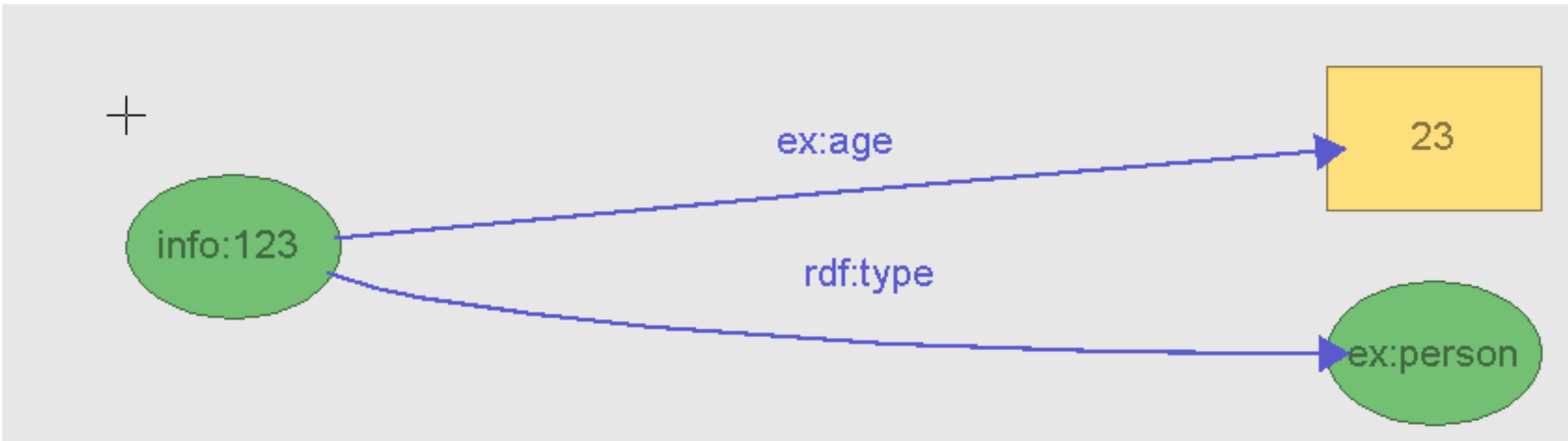
- All defined in rdf namespace
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#
- Types (or classes)
  - rdf:Resource – everything that can be identified (with a URI)
  - rdf:Property – specialization of a resource expressing a binary relation between two resources
  - rdf:statement – a triple with properties rdf:subject, rdf:predicate, rdf:object
- Properties
  - rdf:type - subject is an **instance** of that category or class defined by the value
  - rdf:subject, rdf:predicate, rdf:object – relate elements of statement tuple to a resource of type statement.

Use of rdf:type

- “Resource named http://foo.org/inst is member of class http://foo.org/classes/cl1”
- <http://foo.org/inst> <rdf:type> <http://foo.org/classes/cl1>



Typing the Resources in Statements

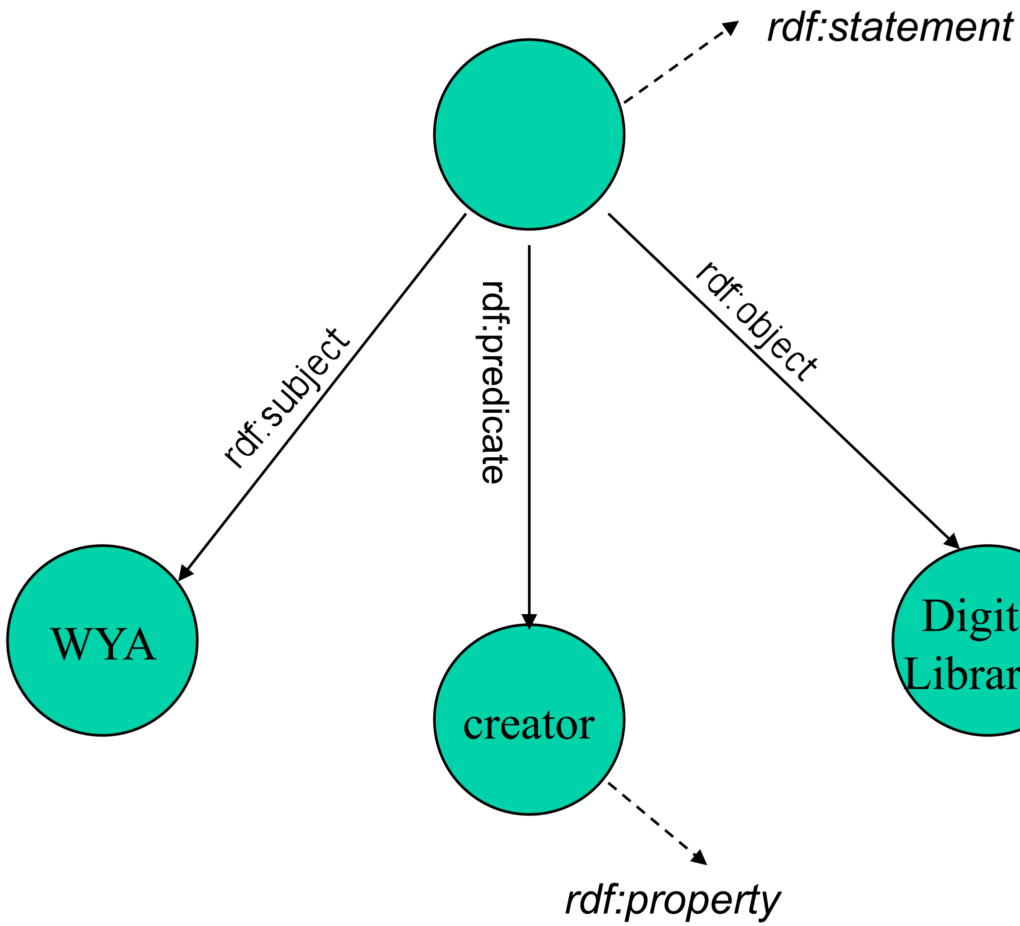



```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:gss="http://www.w3.org/2001/11/IsaViz/graphstylesheets#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ex="http://example.org/terms#">
  - <ex:person rdf:about="info:123">
      <ex:age
        rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">23</ex:age>
    </ex:person>
  </rdf:RDF>
```

Formalizing a statement

- An RDF statement is a triple consisting of:
  - subject → rdf:type resource
  - property → rdf:type property
  - object → rdf:type resource | literal
  - Examples
    - <http://www.cs.cornell.edu/lagoze> <http://purl.org/dc/elements/creator>
      “Carl Lagoze”
    - <http://www.cs.cornell.edu/lagoze> <http://purl.org/dc/elements/creator>
      <mailto:lagoze@cs.cornell>
- Expressible as:
  - triple (ns1:s ns2:p ns3:o)

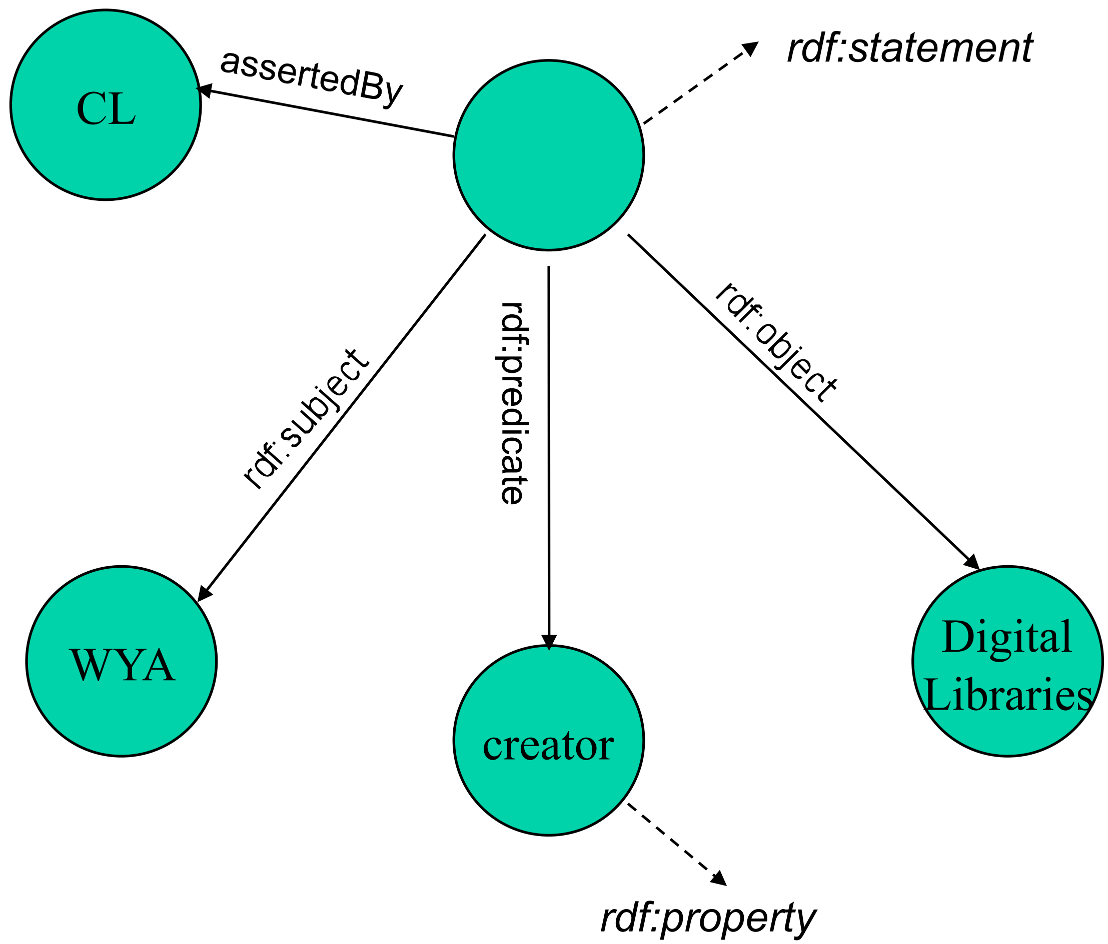
RDF statements and basic types



Simple type inferencing

explicit triple	Allows inference
(:s :p :o)	(:s rdf:type rdf:Resource) (:p rdf:type rdf:Property) (:o rdf:type rdf:Resource)

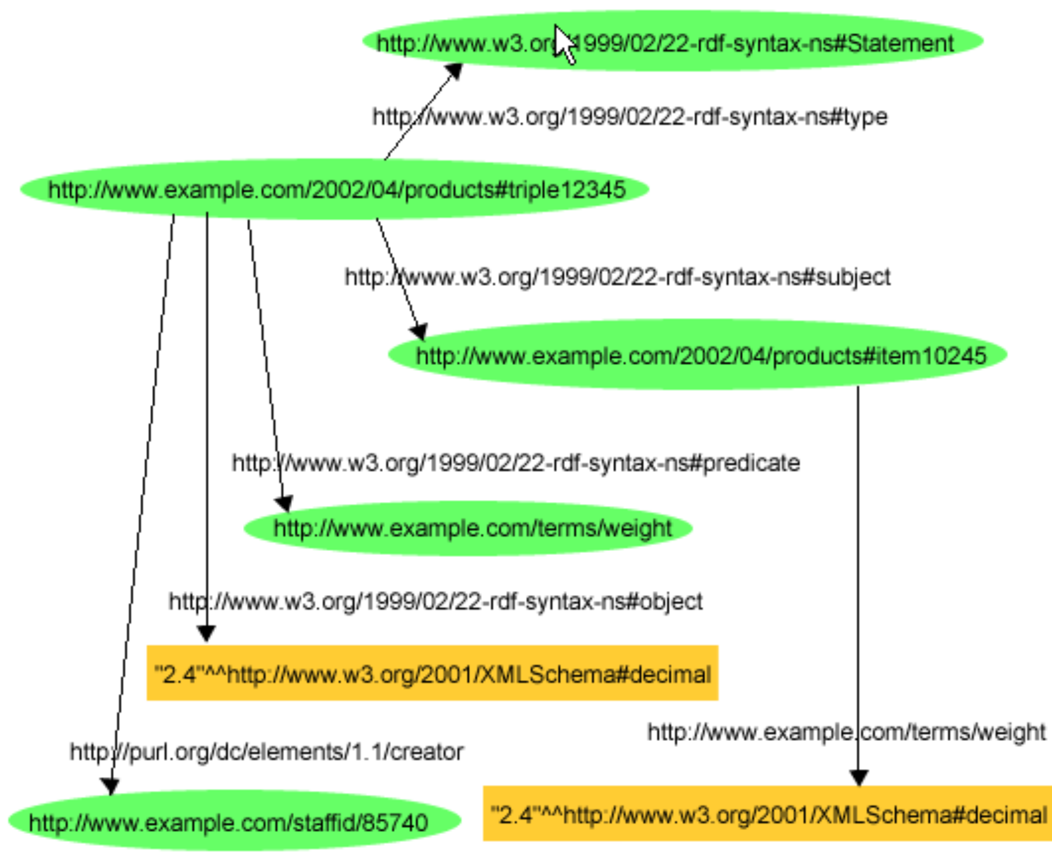
Reification – Statements about statements



"CL says 'WYA wrote Digital Libraries'"

Reification Structure

Staff member 85740 said the weight of item 10245 is 2.4 units

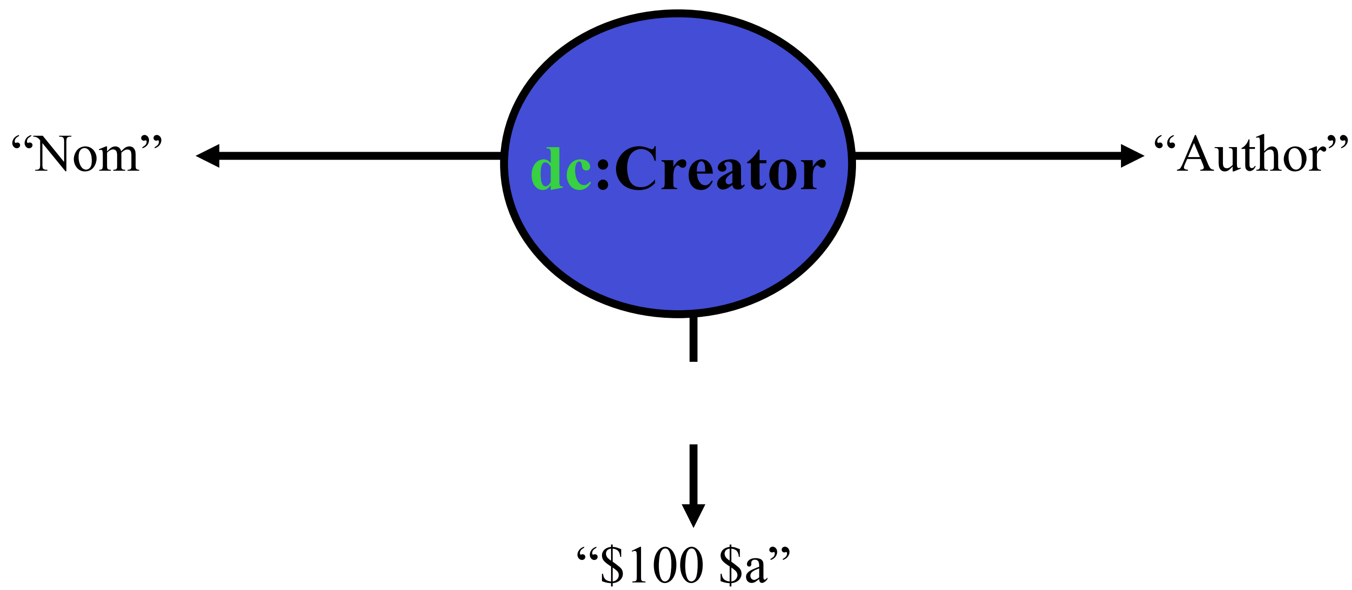


Reification XML

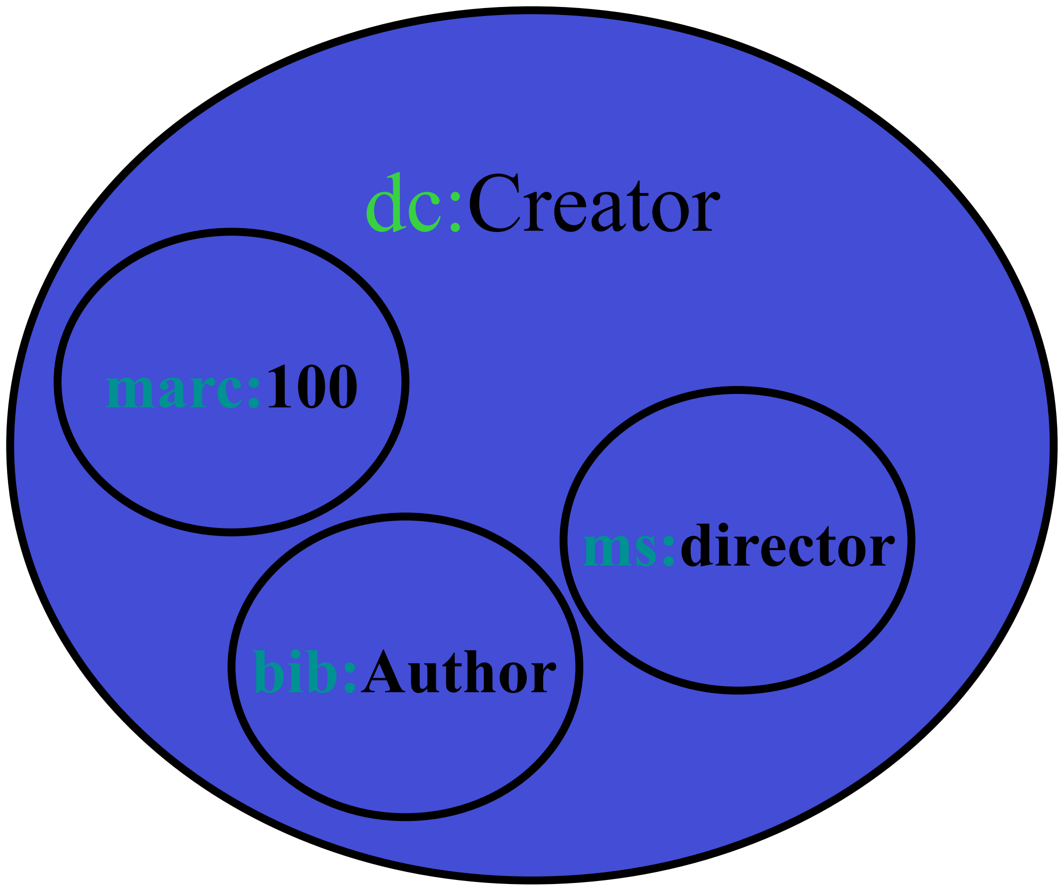
```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:exterms="http://www.example.com/terms/"
            xml:base="http://www.example.com/2002/04/products">

  <rdf:Description rdf:ID="item10245">
     <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
  </rdf:Description>

  <rdf:Statement rdf:about="#triple12345">
     <rdf:subject rdf:resource="http://www.example.com/2002/04/products#item10245"/>
     <rdf:predicate rdf:resource="http://www.example.com/terms/weight"/>
     <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>

     <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
  </rdf:Statement>

</rdf:RDF>
```

Why Schema (1)?

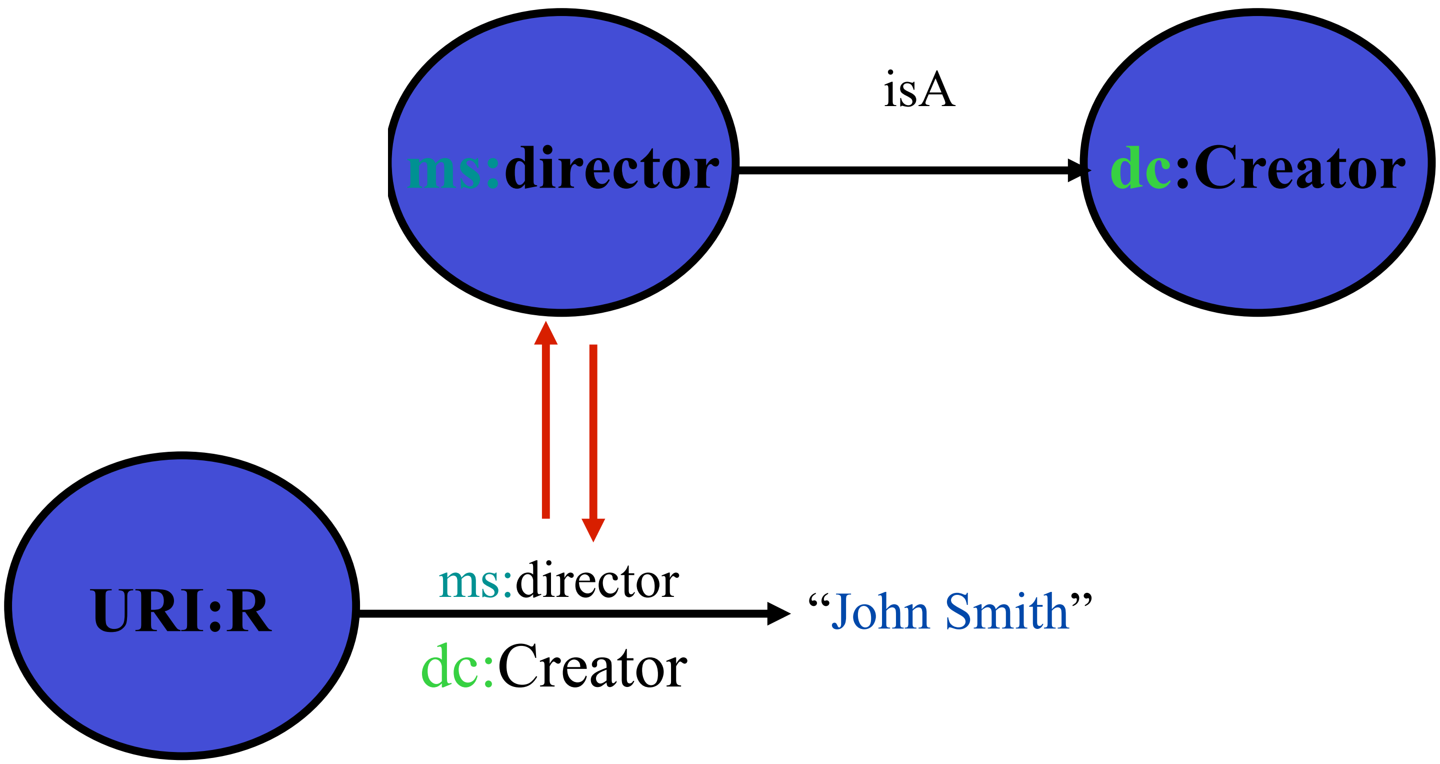
- Enables communities to share machine readable tokens and locally define human readable labels.



Why Schema (2)?
Relationships among vocabularies



Why Schema(3)?
Relationships among vocabulary elements



RDF Schemas

- Declaration of vocabularies
  - classes, properties, and structures defined by a particular community
  - relationship of properties to classes
- Provides substructure for inferences based on existing triples
- NOT prescriptive, but descriptive
- Schema language is an expression of basic RDF model
  - uses meta-model constructs
  - schema are “legal” rdf graphs and can be expressed in RDF/XML syntax

RDFs Namespace

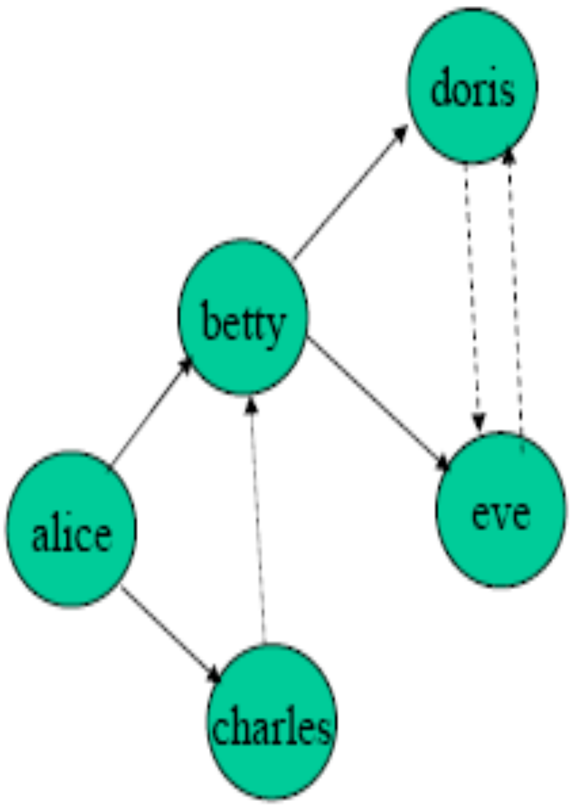
- Class-related
  - rdfs:Class, rdfs:subClassOf
- Property-related
  - rdfs:subPropertyOf, rdfs:domain, rdfs:range

RDF Schema: Specializing Properties

- rdfs:subPropertyOf – allows specialization of relations
  - E.g., the property "father" is a subPropertyOf the property parent
- subProperty semantics

If M contains	Then add
(:s rdfs:subPropertyOf :o)	(:s rdf:type rdf:Property) (:o rdf:type rdf:Property)
(:s :p :o) (:p rdfs:subPropertyOf :q)	(:s :q :o)
(:p rdfs:subPropertyOf :q) (:q rdfs:subPropertyOf :r)	(:p rdfs:subPropertyOf :r)

Inferences from Property Relationships




```
(:alice :has-child :betty)
(:alice :has-child :charles)

(:betty :has-child :doris)
(:betty :has-child :eve)

(:charles : has-sibling :betty)

(:doris :has-sister :eve)
(:eve :has-sister :doris)
```

Sub-Property Semantics

```
(:has-sister rdfs:subPropertyOf :has-sibling)
(:has-brother rdfs:subPropertyOf :has-sibling)

(:has-child rdfs:subPropertyOf :has-descendant)
```

- Using the intended semantics, we can infer:

```
(:alice :has-descendant :betty)
(:alice :has-descendant :charles)

(:alice :has-descendant :doris)
(:alice :has-descendant :eve)
```

Property-based semantics

- Provide basis for type inference from properties
- Not restrictive like xml schema constraints
- rdfs:domain
  - classes of resources that have a specific property
- rdfs:range
  - classes of resources that may be the value of a specific property

If M contains	Then add
(:s :p :o) (:p rdfs:domain :t)	(:s rdf:type :t)
(:s :p :c` (:p rdfs: range :t)	(:o rdf:type :t)

Inferences from Constraints

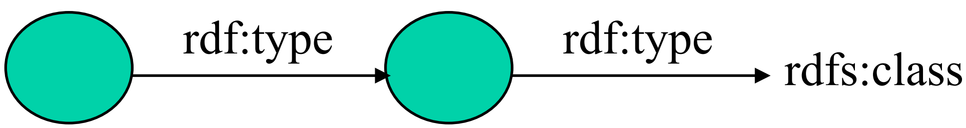
```
(:has-child rdfs:domain parent)
(:has-child rdfs:range person)

(:has-sibling rdfs:domain person)

(:has-brother rdfs:range :male-person)
(:has-sister rdfs:range :female-person)
```

- Using the intended semantics, we can infer:

```
(:alice rdf:type parent)
(:betty rdf:type parent)

(:eve rdf:type femal-person)

(:charles rdf:type :person)
```

Class Declaration

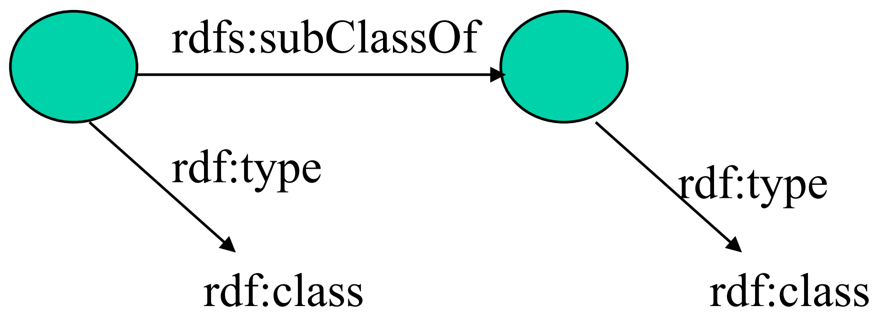
- rdfs:Class
  - Resources denoting a set of resources; range of rdf:type



ex:MotorVehicle rdf:type rdfs:Class
exthings:companyCar rdf:type ex:MotorVehicle

Class Hierarchy

- rdfs:subClassOf
  - Create class hierarchy



ex:MotorVehicle rdf:type rdfs:Class
ex:SUV rdf:type rdfs:Class
ex:SUV rdf:subClassOf ex:MotorVehicle
exthings:companyCar rdf:type ex:SUV

Sub-Class Inferencing

If M contains	Then add
(:s rdf:type :o)	(:o rdf:type rdfs:Class)
(:s rdf:type :o) (:o rdfs:subClassOf :c)	(:s rdf:type :c)
(:s rdfs:subClassOf :o) (:o rdfs:subClassOf :c)	(:s rdfs:subClassOf :c)
(:s rdfs:subClassOf :o)	(:s rdf:type rdfs:Class) (:o rdf:type rdfs:Class)
(:s rdf:type rdfs:Class)	(:s rdfs:subClassOf rdf:Resource)

Sub-class Inferencing Example

```
(:parent rdfs:subClassOf :person)

(:male-person rdfs:subClassOf :person)
(:female-person rdfs:subClassOf :person)

(:mother rdfs:subClassOf :parent)
(:mother rdfs:subClassOf :female-person)
```

- Using the intended semantics, we can infer:

```
(:betty rdf:type person)
```

Jena Toolkit

- Robust tools for building and manipulating RDF models
  - HP Labs Bristol
  - Capabilities
    - Model construction
    - XML and N3 parsing
    - Model persistence (DB foundation)
    - Model querying
    - Ontology building
    - Inferencing
- http://www.hpl.hp.com/semweb/jena2.htm

IsaViz

- Visualizing and constructing RDF models
- http://www.w3.org/2001/11/IsaViz/