Semantic Web Basics (cont.)

CS 431 – March 29, 2006
Carl Lagoze – Cornell University
Acknowledgements for various slides and ideas

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

- [http://www.w3.org/2001/sw/](http://www.w3.org/2001/sw/)
- [http://www.w3.org/RDF/](http://www.w3.org/RDF/)
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
- 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
M. Doe illustrated the book “Best Stories”

Mary Doe animated the cartoon “Best Stories – the movie”

Illustration is a type of contribution

Cartoons and Books are types of Works

animation is a type of contribution

M. Doe and Mary Doe are pseudonyms for Susan Mann

Show me the works to which Susan Mann contributed?
RDF Model Primitives

- Resource
- Property
- Value
- Resource
- Statement
RDF Containers

• Permit aggregation of several values for a property

• Different container semantics
  – Bag
    • unordered grouping (e.g., students in this class)
  – Sequence
    • ordered grouping (e.g., authors of a paper)
  – Alternatives
    • alternate values (e.g., measurement in different units)
RDF Reification

- Treat a statement as a first-class object (resource)
- It then can become a graph element (and be used as subject and object of statements)
Typed Literals

```xml
<?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/staffid/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>
```
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
Formalizing RDF

• There is a meta-model that bootstraps RDF
• Set of basic types (Classes) and properties
• Allows basic inferencing
RDF meta-model basis elements

- All defined in rdf namespace
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#

- Properties
  - rdf:type - subject is an *instance* of that category or class defined by the value

- 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
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>
<?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: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

• Expressible as:
  – triple (ns1:s ns2:p ns3:o)
RDF statements and basic types

- **WYA**
  - rdf:subject
  - rdf:predicate
  - rdf:statement
  - rdf:property

- **Digital Libraries**
  - rdf:object

- **creator**
  - rdf:property
Simple type inferencing

<table>
<thead>
<tr>
<th>explicit triple</th>
<th>Allows inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>(:s :p :o)</td>
<td>(:s rdfs:resource)</td>
</tr>
<tr>
<td></td>
<td>(:p rdfs:resource)</td>
</tr>
<tr>
<td></td>
<td>(:o rdfs:resource)</td>
</tr>
</tbody>
</table>
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
<?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="#item12345">
    <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>
```
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

URI₁

- address
- street
- city
- state
- zip

“blank node” (think of as local variable)

“301 College Ave”
“1thaca”
“NY”
“14850”
Another n-ary relation example

```xml
<?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:extems="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>
        <extems:editor rdf:nodeID="abc"/>
    </rdf:Description>

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

</rdf:RDF>
```
Expressing Collection Primitives in Binary Relations

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

```xml
<?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 and blank nodes

The students in course 6.001 are Amy, Mohamed, and Johann.
Why Schema (1)?

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

```
dc:Creator
```

“Nom” ← dc:Creator → “Author”

“$100 $a”
Why Schema (2)?
Relationships among vocabularies

- dc:Creator
- marc:100
- bib:Author
- ms:director
Why Schema(3)?

Relationships among vocabulary elements

URI: R

ms:director

isA

dc:Creator

"John Smith"

ms:director

dc:Creator
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

<table>
<thead>
<tr>
<th>If M contains</th>
<th>Then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>(:s rdfs:subPropertyOf :o)</td>
<td>(:s rdf:type rdf:Property)</td>
</tr>
<tr>
<td></td>
<td>(:o rdf:type rdf:Property)</td>
</tr>
<tr>
<td>(:s :p :o)</td>
<td>(:s :q :o)</td>
</tr>
<tr>
<td>(:p rdfs:subPropertyOf :q)</td>
<td>(:q rdfs:subPropertyOf :r)</td>
</tr>
<tr>
<td>(:p rdfs:subPropertyOf :q)</td>
<td>(:p rdfs:subPropertyOf :r)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>If M contains</th>
<th>Then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>(:s :p :o)</td>
<td>(:s rdf:type :t)</td>
</tr>
<tr>
<td>(:p rdfs:domain :t)</td>
<td></td>
</tr>
<tr>
<td>(:s :p :c)</td>
<td>(:o rdf:type :t)</td>
</tr>
<tr>
<td>(:p rdfs:range :t)</td>
<td></td>
</tr>
</tbody>
</table>
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
ex: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
ex:things:companyCar rdf:type ex:SUV
```
Sub-Class Inferencing

<table>
<thead>
<tr>
<th>If M contains</th>
<th>Then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>(:s rdf:type :o)</td>
<td>(:o rdf:type rdfs:Class)</td>
</tr>
<tr>
<td>(:s rdf:type :o)</td>
<td>(:s rdf:type :c)</td>
</tr>
<tr>
<td>(:o rdfs:subClassOf :c)</td>
<td></td>
</tr>
<tr>
<td>(:s rdfs:subClassOf :o)</td>
<td>(:s rdfs:subClassOf :c)</td>
</tr>
<tr>
<td>(:o rdfs:subClassOf :c)</td>
<td></td>
</tr>
<tr>
<td>(:s rdfs:subClassOf :o)</td>
<td>(:s rdf:type rdfs:Class)</td>
</tr>
<tr>
<td>(:o rdf:type rdfs:Class)</td>
<td>(:s rdfs:subClassOf rdf:Resource)</td>
</tr>
</tbody>
</table>
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/