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
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

Resource → Property → Resource

Statement
RDF Model Example #2
<?xml version="1.0"?>
<rdf:RDF xmlns:gss="http://www.w3.org/2001/11/IsaViz/graphstylesheets#"
  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
<?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
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

```
\[
\text{URI}_1
\]
```

```
street
```

```
\text{"blank node" (think of as local variable)}
```

```
\text{address}
```

```
\text{city}
```

```
\text{state}
```

```
\text{zip}
```

```
\text{"301 College Ave"}
```

```
\text{"Ithaca"}
```

```
\text{"NY"}
```

```
\text{"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/dajoeb/"/>
  </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
<?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/elements/1.1/
    <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
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
<?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
      <mailto:lagoze@cs.cornell>

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

RDF: 
- Subject (WYA) 
- Predicate (creator) 
- Object (Digital Libraries)
## 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 rdf:type rdf:Resource)</td>
</tr>
<tr>
<td></td>
<td>(:p rdf:type rdf:Property)</td>
</tr>
<tr>
<td></td>
<td>(:o rdf:type rdf: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="#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

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

URI:R

ms:director \isA{} dc:Creator

ms:director -> "John Smith"

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>(:p :r)</td>
</tr>
<tr>
<td>(:p rdfs:subPropertyOf :q)</td>
<td>(:p :r)</td>
</tr>
<tr>
<td>(:q rdfs:subPropertyOf :r)</td>
<td>(:p :q :r)</td>
</tr>
</tbody>
</table>
Inferences from Property Relationships

\[
\begin{align*}
(:alice :has-child :betty) \\
(:alice :has-child :charles) \\
(:betty :has-child :doris) \\
(:betty :has-child :eve) \\
(:charles : has-sibling :betty) \\
(:doris :has-sister :eve) \\
(:eve :has-sister :doris)
\end{align*}
\]
Sub-Property Semantics

- Using the intended semantics, we can infer:

\[
\begin{align*}
(:\text{alice} & \text{:has-descendant} :\text{betty}) \\
(:\text{alice} & \text{:has-descendant} :\text{charles}) \\
(:\text{alice} & \text{:has-descendant} :\text{doris}) \\
(:\text{alice} & \text{:has-descendant} :\text{eve})
\end{align*}
\]
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>(:o rdf:type :t)</td>
</tr>
<tr>
<td>(:s :p :c')</td>
<td></td>
</tr>
<tr>
<td>(:p rdfs:range :t)</td>
<td></td>
</tr>
</tbody>
</table>
Inferences from Constraints

\[
\begin{align*}
(:&\text{has-child} \text{ rdfs:domain} \text{ parent}) \\
(:&\text{has-child} \text{ rdfs:range} \text{ person}) \\
(:&\text{has-sibling} \text{ rdfs:domain} \text{ person}) \\
(:&\text{has-brother} \text{ rdfs:range} \text{ :male-person}) \\
(:&\text{has-sister} \text{ rdfs:range} \text{ :female-person})
\end{align*}
\]

- Using the intended semantics, we can infer:

\[
\begin{align*}
(:&\text{alice} \text{ rdf:type} \text{ parent}) \\
(:&\text{betty} \text{ rdf:type} \text{ parent}) \\
(:&\text{eve} \text{ rdf:type} \text{ female-person}) \\
(:&\text{charles} \text{ rdf:type} \text{ :person})
\end{align*}
\]
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

<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>(:s rdfs:subClassOf :c)</td>
</tr>
<tr>
<td>(:s rdfs:subClassOf :o)</td>
<td>(:s rdfs:subClassOf :c)</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/