Semantic Web - RDF

Prof. Dr. Steffen Staab
Dipl.-Inf. Med. Bernhard Tausch

Outline

• Motivation: Why XML is not enough
• Introduction to RDF
  – Requirements for KR on the Web
  – The RDF Data Model
  – RDF Schema
• Extensions of RDF(S)
  • Tools for RDF and RDF Schema
    – Parser, Query, and Inference Engines

Applications and Technologies on top of XML

Extensible Markup Language (XML) Revisited

• Key idea: separate structure from presentation
• XML DTDs or Schema define document structure
  Replace HTML with two things
  • A domain specific markup language (defined in XML)
  • A map from that markup language to HTML (defined using XSL)
• DTD enables document recipients to tell whether they’ve received a well-formed document
  – Gives a minimal level of validation
Why XML is not enough

- Only advantage of using XML is reusing the parser and document validation
- Many different possibilities to encode a domain of discourse
- Leads to difficulties when understanding of foreign documents is required

==> Next step: separate content from structure!

Encoding of Knowledge: Example

"The Creator of the Resource "http://www.w3.org/Home/Lassila" is Ora Lassila"

Endless encoding possibilities in XML:

```xml
<Creator>
  <uri>http://www.w3.org/Home/Lassila</uri>
  <name>Ora Lassila</name>
<Creator>
```

```xml
<Document uri="http://www.w3.org/Home/Lassila">
  <Creator>Ora Lassila</Creator>
</Document>
```

```xml
<Document uri="http://www.w3.org/Home/Lassila" Creator="Ora Lassila"/>
```

Introduction to RDF

- RDF (Resource Description Framework)
  - Beyond Machine readable to Machine understandable
- RDF unites a wide variety of stakeholders:
  - Digital librarians, content-raters, privacy advocates, B2B industries, AI...
  - Significant (but less than XML) industrial momentum, lead by W3C
- RDF consists of two parts
  - RDF Model (a set of triples)
  - RDF Syntax (different XML serialization syntaxes)
- RDF Schema for definition of Vocabularies (simple Ontologies) for RDF (and in RDF)

RDF Data Model

- **Resources**
  - A resource is a thing you talk about (can reference)
  - Resources have URI’s
  - RDF definitions are itself Resources (linkage)
- **Properties**
  - slots, defines relationship to other resources or atomic values
- **Statements**
  - “Resource has Property with Value”
  - (Values can be resources or atomic XML data)
- **Similar to Frame Systems**
A simple Example

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

- **Structure**
  - Resource (subject) http://www.w3.org/Home/Lassila
  - Property (predicate) http://www.schema.org/#Creator
  - Value (object) “Ora Lassila”

- **Directed graph**
  - http://www.w3.org/Home/Lassila
  - s:Creator
  - Ora Lassila

Another Example

- To add properties to Creator, point through an intermediate Resource.

- RDF graph:
  - http://www.w3.org/Home/Lassila
  - s:Creator
  - Person://fi/654645635
  - Name: Ora Lassila
  - Email: lassila@w3.org

Collection Containers

- Multiple occurrences of the same PropertyType doesn’t establish a relation between the values
  - The Millers own a boat, a bike, and a TV set
  - The Millers need (a car or a truck)
  - (Sarah and Bob) bought a new car

- RDF defines three special Resources:
  - **Bag** unordered values rdf:Bag
  - **Sequence** ordered values rdf:Seq
  - **Alternative** single value rdf:Alt

- Core RDF does not enforce ‘set’ semantics amongst values

Example: Bag

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

- RDF graph:
  - /courses/6.001
  - rdf:type
  - /students
    - rdf:Bag
    - /Students/Amy
    - /Students/Tim
    - /Students/John
    - /Students/Mary
    - /Students/Sue
  - bagid1
  - rdf:_1
  - rdf:_2
  - rdf:_3
  - rdf:_4
  - rdf:_5
Example: Alternative

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

Example: Reification

• Ralph Swick believes that
  – the creator of the resource http://www.w3.org/Home/Lassila is Ora Lassila

A Formal Model of RDF

• RDF itself is mathematically straightforward:
  – Basic Definitions
    • Resources.
    • Properties \( \subseteq \) Resources called
    • Literals
  – Statements = Resources \( \times \) Properties \( \times \{ \text{Resources} \cup \text{Literals} \} \)
  – Typing
    • rdf:type \( \in \) Properties
    • \{ RDF:type, sub, obj \} \( \in \) Statements \( \Rightarrow \) obj \( \in \) Resources

Statements about Statements (Requirement 2: Dispute Statements)

• Making statements about statements requires a process for transforming them into Resources
  – subject the original referent
  – predicate the original property type
  – object the original value
  – type rdf:Statement
Formal Model of RDF II

- Reification
  - \( \text{rdf:Statement} \in \text{Resource-Properties} \)
  - \( \{ \text{rdf:predicate}, \text{rdf:subject}, \text{rdf:object} \} \subset \text{Properties} \)
  - Reification of a triple \( (\text{pred, sub, obj}) \) of Statements is an element \( r \) of Resources representing the reified triple and the elements \( s_1, s_2, s_3, \) and \( s_4 \) of Statements such that
    - \( s_1: \{ \text{RDF:predicate}, r, \text{pred} \} \)
    - \( s_2: \{ \text{RDF:subject}, r, \text{subj} \} \)
    - \( s_3: \{ \text{RDF:object}, r, \text{obj} \} \)
    - \( s_4: \{ \text{RDF:type}, r, [\text{RDF:Statement}] \} \)
- Collections
  - \( \{ \text{RDF:Seq}, \text{RDF:Bag}, \text{RDF:Alt} \} \subset \text{Resources-Properties} \)
  - 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, ...

RDF Syntax I

- Datamodel does not enforce particular syntax
- Specification suggests many different syntaxes based on XML

General form:

```xml
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
    <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
  </rdf:Description>
</rdf:RDF>
```

RDF Syntax II: Syntactic Varieties

Typing Information

```xml
<s:Homepage rdf:about="http://www.w3.org/Home/Lassila">
  <s:Creator="Ora Lassila"/>
  <s:Title>Ora's Home Page</s:Title>
  <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
</s:Homepage>
```

Subject (OID)

```xml
<s:Homepage rdf:about="http://www.w3.org/Home/Lassila">
  <s:Creator="Ora Lassila"/>
  <s:Title>Ora's Home Page</s:Title>
  <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
</s:Homepage>
```

In-Element Property

```xml
<s:Homepage rdf:about="http://www.w3.org/Home/Lassila">
  <s:Creator="Ora Lassila"/>
  <s:Title>Ora's Home Page</s:Title>
  <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
</s:Homepage>
```
RDF Schema (RDFS)

- RDF just defines the datamodel
- Need for definition of vocabularies for the datamodel - an Ontology Language!
- RDF schemas are Web resources (and have URIs) and can be described using RDF

Most Important Modeling Primitives

- Core Classes
  - Root-Class rdfs:Resource
  - MetaClass rdfs:Class
  - Literals rdfs:Literal
  - rdfs:subClassOf-property
- Inherited from RDF: properties (slots)
  - rdfs:domain & rdfs:range
  - rdfs:label, rdfs:comment, etc.
- Inherited from RDF: InstanceOf (rdfs:type)

RDF-Schema: Example

```
<rdfs:subClassOf about="Xyz:Minivan">
  <rdfs:subClassOf about="Xyz:Van"/>
</rdfs:subClassOf>
```

```
<rdfs:subClassOf about="myvan">
  <rdf:type about="Xyz:MiniVan"/>
</rdfs:subClassOf>
```

Predicate Logic Consequences:

Forall X: type(X,MiniVan) -> type(X,Van).
Forall X: subclassOf(X,MiniVan) -> subclassOf(X,Van).
Rdf:property

```xml
<rdf:description about="possesses">
  <rdf:type about="...property"/>
  <rdfs:domain about="person"/>
  <rdfs:range about="vehicle"/>
</rdf:description>

<rdf:description about="peter">
  <possesses>petersminivan</possesses>
</rdf:description>
```

**Predicate Logic Consequences:**
For all X, Y: possesses (X, Y) -> (type(X, person) & type(Y, vehicle)).

**Simplification rules**
- People specifying text for arbitrary RDF processors can use any simplification
- Processors of arbitrary RDF therefore must accept all simplifications
- Special-purpose XML formats can be RDF-compliant while disallowing simplifications, requiring them, or exploiting them in specific ways

**Container examples**
- Bag: committee members, documents in a folder, checks in a bag
- Seq: book authors (order counts!), chapters in a book, items in an agenda
- Alt: document home and mirrors, mailing-list moderators, translations of a document
A Bag: one variant

```xml
<rdf:Description ID="committee">
  <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"/>
  <rdf:_1>Jack Robinson</rdf:_1>
  <rdf:_2>John Doe</rdf:_2>
  <rdf:_3>Richard Roe</rdf:_3>
</rdf:Description>
```

A Bag: another variant

```xml
<rdf:Bag ID="committee">
  <rdf:li>Jack Robinson</rdf:li>
  <rdf:li>John Doe</rdf:li>
  <rdf:li>Richard Roe</rdf:li>
</rdf:Bag>
```

- Using an “rdf:Bag” element means the value of “type” is “http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag”

What is the “rdf:type” property?

- It specifies a class (there may be more than one) to which the resource belongs
- Its value is always a Web resource representing the class
- It can be expressed as a “type” attribute on a Description element
- It can also be implied by using a special element instead of a Description element

Kinds of “about” attributes

- “about”: specifies the URL of the Web resource directly
- “aboutEach”: specifies the URL of a container; the properties apply to the individual members of the container
- “aboutEachPrefix”: specifies an URL prefix; the properties apply to all Web resources with that prefix
Containers vs. multiple values

- A property can appear more than once with different values
- What is true of a container isn’t necessarily true of its contents and vice versa
- “aboutEach” lets us get to the contents when we already have a container
- “aboutEachPrefix” in effect manufactures a container based on URLs

Reified statements

- We reify statements so that we can talk about them rather than asserting them
- “Charles Dickens is the author of Bleak House” asserts a property of Charles Dickens
- “Jack believes that Charles Dickens is the author of War and Peace” asserts a property of Jack, not Charles Dickens

Reification properties

```xml
<rdf:Description about="...">
  <xx:creator>Charles Dickens</xx:creator>
</rdf:Description>
```

reifies as:

```xml
<rdf:Statement>
  <rdf:subject resource="...">
    <rdf:predicate resource="#creator/">
      <rdf:object>Charles Dickens</rdf:object>
    </rdf:predicate>
  </rdf:subject>
</rdf:Statement>
```

RDF Schemas

- Describe rules for using RDF properties
- Are expressed in RDF
- Are not to be confused with XML Schemas
RDF Classes

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

Property-centric classes

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

Specifying classes

• To specify a class, create an RDF resource of type rdfs:Class
  `<rdfs:Class id="MyClass">`  
  `<rdfs:label>My Class</rdfs:label>`  
  `<rdfs:comment>John Cowan’s demonstration Class</rdfs:comment>`  
  `</rdfs:Class>`

Specifying properties

• To specify a property, create an RDF resource of type rdfs:Property
  `<rdfs:Property id="myProperty">`  
  `<rdfs:comment>John Cowan’s demo property</rdfs:comment>`  
  `<rdfs:domain resource="#MyClass"/>`  
  `<rdfs:range resource="#Literal"/>`  
  `</rdfs:Property>`
Schema URIs

- Ordinary XML namespace URIs are just to guarantee uniqueness: there is no assumption that the URI refers to anything useful (or even refers at all)
- URIs for namespaces used in RDF, though, should refer to an RDF schema document

Properties (1)

- "rdf:type" relates any resource to its class
- "rdfs:subClassOf" relates a subclass to its superclass (multiple inheritance is OK)
- "rdfs:subPropertyOf" relates a subproperty to its superproperty

Properties (2)

- "rdfs:seeAlso" relates a resource to another resource explaining it (use a subproperty to specify the nature of the explanation)
- "rdfs:isDefinedBy" is a subproperty of "rdfs:seeAlso" and relates a resource to its definition, typically an RDF schema

Properties (3)

- "rdfs:domain" specifies the domain of a property (the classes of its subjects); if unknown, anything can be a subject
- "rdfs:range" specifies the range of a property (the single class of its objects); if unknown, anything can be an object
Properties (4)

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

Properties (5)

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

Classes (1)

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

Classes (2)

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

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

Dublin Core

- “Title”: the name given to the resource
- “Creator”: the person or organization primarily responsible for the resource
- “Subject”: what the resource is about
- “Description”: a description of the content

Dublin Core

- “Publisher”: the person or organization responsible for making the resource available
- “Contributor”: someone who has provided content to the resource other than the creator
- “Date”: date of creation or publication

Dublin Core

- “Type”: type of resource, such as home page, technical report, novel, photograph...
- “Format”: data format of the resource
- “Identifier”: URL, ISBN number, ...
- “Source”: another resource that this resource is derived from
Dublin Core

- "Language": the language of the content
- "Relation": another resource and its relationship to this one
- "Coverage": the portion of time or space described by this resource (atlases, histories, etc.)

Dublin Core

- "Rights": the intellectual property rights adhering to this resource, or a pointer to them

Where to look next

- RDF Syntax & Model theory
  [http://www.w3.org/RDF/](http://www.w3.org/RDF/)

Extensibility of RDF

- Define an Ontology of your Language with RDF Schema (like RDF-Schema itself)
- Describe Instance Data using your new Vocabulary
  Advantage: all Languages use the same Data Model (simplifies Interoperability)
Formal Models of RDF I

• Official Semantics: RDF Model Theory
  [ → http://www.w3.org/TR/rdf-mt/ ]

  Specification of a precise semantics for RDF (and RDFS),
  and of corresponding entailment and inference rules which
  are sanctioned by the semantics.

• Earlier Proposals:
  – RDFS(FA)
    [ → http://dl-web.man.ac.uk/rdfsfa/ ]
    UML-Like Stratification
  – RDF in First-Order Logic
    [ → http://nestroy.wi-inf.uni-essen.de/rdf/logical_interpretation/ ]
    (Outdated RDF semantics in FOL)

Formal Models of RDF II

• RDF-MT is based on classical Tarski-style Model Theory

• Some Entailment rules for RDFS:

<table>
<thead>
<tr>
<th>If E contains:</th>
<th>then add:</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf1</td>
<td>xxx aaa yyy .</td>
</tr>
<tr>
<td></td>
<td>aaa rdf:type</td>
</tr>
<tr>
<td></td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs2</td>
<td>xxx aaa yyy .</td>
</tr>
<tr>
<td></td>
<td>aaa rdfs:domain zzz .</td>
</tr>
<tr>
<td></td>
<td>xxx rdf:type zzz .</td>
</tr>
<tr>
<td>rdfs3</td>
<td>xxx aaa uuu .</td>
</tr>
<tr>
<td></td>
<td>aaa rdfs:range zzz .</td>
</tr>
<tr>
<td></td>
<td>uuu rdf:type zzz .</td>
</tr>
<tr>
<td>rdfs4a</td>
<td>xxx aaa yyy .</td>
</tr>
<tr>
<td></td>
<td>xxx rdf:type rdfs:Resource</td>
</tr>
<tr>
<td>rdfs4b</td>
<td>xxx aaa uuu .</td>
</tr>
<tr>
<td></td>
<td>uuu rdf:type rdfs:Resource</td>
</tr>
<tr>
<td>rdfs5a</td>
<td>aaa rdfs:subPropertyOf bbb .</td>
</tr>
<tr>
<td></td>
<td>aaa rdf:type ccc .</td>
</tr>
<tr>
<td>rdfs5b</td>
<td>xxx rdf:type rdf:Property .</td>
</tr>
<tr>
<td></td>
<td>xxx rdfs:subPropertyOf xxx</td>
</tr>
<tr>
<td>rdfs6</td>
<td>xxx aaa yyy .</td>
</tr>
<tr>
<td></td>
<td>aaa rdfs:subPropertyOf bbb .</td>
</tr>
<tr>
<td></td>
<td>xxx bbb yyy .</td>
</tr>
<tr>
<td>rdfs7a</td>
<td>xxx rdf:type rdfs:Class .</td>
</tr>
<tr>
<td></td>
<td>xxx rdfs:subClassOf xxx .</td>
</tr>
<tr>
<td>rdfs7b</td>
<td>xxx rdf:type rdfs:Class .</td>
</tr>
<tr>
<td></td>
<td>xxx rdfs:subClassOf xxx .</td>
</tr>
<tr>
<td></td>
<td>xxx rdfs:subClassOf yyy .</td>
</tr>
<tr>
<td></td>
<td>yyy rdfs:subClassOf zzz .</td>
</tr>
<tr>
<td></td>
<td>xxx rdfs:subClassOf zzz .</td>
</tr>
<tr>
<td>rdfs8</td>
<td>xxx rdfs:subClassOf yyy .</td>
</tr>
<tr>
<td></td>
<td>aaa rdf:type xxx .</td>
</tr>
<tr>
<td></td>
<td>xxx rdf:type</td>
</tr>
<tr>
<td>rdfs9</td>
<td>xxx rdf:typerdfs:ContainerMembershipProperty .</td>
</tr>
<tr>
<td></td>
<td>xxx rdfs:subPropertyOf rdfs:member .</td>
</tr>
<tr>
<td>rdfs10</td>
<td>rdfs: ContainerMembershipProperty .</td>
</tr>
<tr>
<td></td>
<td>aaa rdf:type yyy .</td>
</tr>
</tbody>
</table>

Formal Model of RDF III

• The entailment process terminates on any finite RDF graph
  ➔ only finitely many possible triples can be formed from a given
  finite vocabulary.

• Example Graph (Single Triple): [foo bar baz].

  Closure (for mentioned rules only!):

  1. foo bar baz .
     Source
  2. foo rdf:type rdfs:Resource .
     Rule 4a on (1)
  3. baz rdf:type rdfs:Resource .
     Rule 4a on (1)
  4. bar rdf:type rdf:Property .
     Rule 1 on (1)
  5. rdf:type rdf:type rdf:Property .
     Rule 1 on (4)
  6. rdf:type rdfs:subPropertyOf rdf:type .
     Rule 5b on (5)
  7. bar rdfs:subPropertyOf bar .
     Rule 5b on (4)
  8. rdfs:subPropertyOf rdf:type rdf:Property .
     Rule 1 on (6)
Blank nodes

• Nodes need not be named.
• Unnamed nodes are interpreted as having unique names.
• Implication: graph matching that fulfills a particular condition becomes NP hard
• Conclusion: not so simple as it may seem!