Chapter 3

RDF

Introduction

- Problem: What does an XML document mean?
  - XML is about data structures
  - Their meaning (semantics) is not apparent to a machine
- RDF is more a data model than a language
  - Is realized in many different formats
- RDF define basic semantics
  - RDFS and OWL define more RDF vocabulary for building rich data models
- RDF remains domain independent

Example

- What does this mean?
  - Are professors also academic staff members?
  - If someone teaches a course, are they an academic staff member?
- Can’t say in XML, but can say so in RDFS

Example

- Embedding of elements is just a syntactic constraint
- No meaning is defined
- It’s in the documentation or the mind of the viewer
- Does the machine have a mind?
Key Documents

All at http://www.w3.org/RDF/
- RDF/XML Syntax Specification (Revised)
  Dave Beckett, ed.
- RDF Vocabulary Description Language 1.0: RDF Schema
  Dan Brickley, R.V. Guha, eds.
- RDF Primer
  Frank Manola, Eric Miller, eds.
- Resource Description Framework (RDF): Concepts and Abstract Syntax
  Graham Klyne, Jeremy Carroll, eds.
- RDF Semantics
  Patrick Hayes, ed.
- RDF Test Cases
  Jan Grant, Dave Beckett, eds.

RDF is the first SW language

XML Encoding

```xml
<rdf:RDF ...>
  ...
  ...
</rdf:RDF>
```

Graph

```
Good for Human Viewing
```

Triples

```
stmt(docInst, rdf_type, Document)
stmt(personInst, rdf_type, Person)
stmt(inroomInst, rdf_type, InRoom)
stmt(personInst, holding, docInst)
stmt(inroomInst, person, personInst)
```

Good for Machine Processing

```
xml:docInst, rdf_type, Document
xml:personInst, rdf_type, Person
xml:inroomInst, rdf_type, InRoom
xml:personInst, holding, docInst
xml:inroomInst, person, personInst
```

Good for Reasoning

```
subject, predicate, object
```

The RDF Data Model

- An RDF document is an unordered collection of statements, each with a subject, predicate and object (aka triples)
- A triple can be thought of as a labelled arc in a graph
- Statements describe properties of web resources
- A resource is any object that can be pointed to by a URI:
  - a document, a picture, a paragraph on the Web, ...
  - E.g., http://umbc.edu/~finin/cv.html
  - a book in the library, a real person (?)
  - isbn://5031-4444-3333
  - ...
- Properties themselves are also resources (URIs)

RDF Building Blocks

- Resources
  - Things we can talk about, URIs
- Properties
  - Special things that represent binary relations
- Literal data
  - Strings, integers, dates, ... xmldatatypes
- Statements, aka triples
  - Subject Predicate Object or
  - Subject Property Value
URIs are a foundation

- **URI = Uniform Resource Identifier**
  - “The generic set of all names/addresses that are short strings that refer to resources”
  - URLs (Uniform Resource Locators) are a subset of URIs, used for resources that can be accessed on the web
- **URIs look like “normal” URLs**, often with fragment identifiers to point to a document part:
  - http://foo.com/bar/mumble.html#pitch
- **URIs are unambiguous, unlike natural language terms**
  - the web provides a global namespace
  - We assume references to the same URI are to the same thing

What does a URI mean?

- Sometimes URIs denote a web resource
  - http://umbc.edu/~finin/finin.jpg denotes a file
  - We can use RDF to make assertions about the resource, e.g., it’s an image and depicts a person with name Tim Finin, …
- Sometimes concepts in the external world
  - E.g., http://umbc.edu/ denotes a particular University located in Baltimore
  - This is done by social convention
- **Cool URIs don’t change**
  - http://www.w3.org/Provider/Style/URI

Simple RDF Example

```
http://umbc.edu/~finin/talks/idm02/  
  dc:Title “Intelligent Information Systems on the Web” 
  dc:Creator 
  bib:Aff bib:name “Tim Finin” bib:email “finin@umbc.edu” 
```

RDF Data Model is a Graph

- **Graphs only allow binary relations**
- Higher arity relations must be “reified” (i.e., turned into objects)
- Represent give(John,Mary,Book32) as three binary relations all involving a common object, giveEvent32
  - giver( giveEvent45 , John )
  - recipient( giveEvent45 , Mary )
  - gift( giveEvent45 , Book32 )
- When using RDF, this has to be part of your vocabulary design
- This is a price we have to pay for using a simple representation
RDF Statements

- RDF has one predefined scheme (syntax and semantics) for the reification of RDF statements themselves
- Needed to support assertions about triples
  - Document32 asserts "John gave Mary a book"
  - Tom believes John gave Mary a book
  - "John gave Mary a Book" has 0.33 probability

XML encoding for RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bib="http://daml.umbc.edu/ontologies/bib/">
  <description about="http://umbc.edu/~finin/talks/idm02/">
    <dc:title>Intelligent Information Systems on the Web </dc:Title>
    <dc:creator>
      <description>
        <bib:name>Tim Finin</bib:name>
        <bib:email>finin@umbc.edu</bib:Email>
        <bib:aff resource="http://umbc.edu" />
      </description>
    </dc:creator>
  </description>
</rdf:RDF>
```

Note that the document is a single RDF element which has attributes defining several namespaces.
- One for the rdf vocabulary
- One for the dublin core
- One for the bib vocabulary

Here's the general way to introduce a "named subject" about which we want to assert some properties and values.
- We name subjects by referring to their URI.
- An element in the description tag specify a property and its value
**Descriptions**

- Every description makes a statement about a resource.
- There are different ways:
  - an about attribute: referencing to an existing resource
    
  ```xml
description rdf:about="http://umbc.edu/~finin/talks/idm02/"
  ```
  - an id attribute: creating a new resource
    
  ```xml
description ID="foo3456"
  ```
  - without a name: creating an anonymous resource
    
  ```xml
description
  ```

**XML encoding for RDF**

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bib="http://daml.umbc.edu/ontologies/bib/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
  <description about="http://umbc.edu/~finin/talks/idm02/">
    <dc:title>Intelligent Information Systems on the Web</dc:title>
    <dc:creator>
      <description>
        <bib:name>Tim Finin</bib:name>
        <bib:email>finin@umbc.edu</bib:email>
        <bib:aff resource="http://umbc.edu/" />
      </description>
    </dc:creator>
  </description>
</rdf:RDF>
```

- The value of creator is defined by the nested RDF.
- The nameless description produces a “blank node”:
  
  ```xml
ting the property (or predicate)
  • It’s value is the literal string “Intelligent Information Systems on the Web”
  • By default we assume the datatype is string
  ```
  - This style of XML encoding is called “striped”

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bib="http://daml.umbc.edu/ontologies/bib/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
  <description about="http://umbc.edu/~finin/talks/idm02/">
    <dc:title>Intelligent Information Systems on the Web</dc:title>
    <dc:creator>
      <description>
        <bib:name>Tim Finin</bib:name>
        <bib:email>finin@umbc.edu</bib:email>
        <bib:aff resource="http://umbc.edu/" />
      </description>
    </dc:creator>
  </description>
</rdf:RDF>
```

- Note the “self closing” tag
- The value of the bib:aff property is a resource, not a string
- Every resource has a URI, every URI refers to a resource
- How would this be interpreted?
  
  ```xml
  <bib:aff resource="http://umbc.edu/" />
  ```
N triple representation

- RDF can be encoded as a set of triples.
  \(<\text{subject}> <\text{predicate}> <\text{object}>\).

\(<\text{http://umbc.edu/~finin/talks/idm02/> <\text{http://purl.org/dc/elements/1.1/Title}> "\text{Intelligent Information Systems on the Web}" .
\(_{\text{j10949}} <\text{http://daml.umbc.edu/ontologies/bib/Name}\ textbox{\"Tim Finin\"}.\)
\(_{\text{j10949}} <\text{http://daml.umbc.edu/ontologies/bib/Email}\ textbox{\text{"finin@umbc.edu"}} .\)
\(_{\text{j10949}} <\text{http://daml.umbc.edu/ontologies/bib/Aff} <\text{http://umbc.edu/> .\)
\(_{\text{j10949}} <\text{http://www.w3.org/1999/02/22-rdf-syntax-ns#}\text{t}\text{ype}> <\text{Description}> .\)
\(_{\text{j10949}} <\text{http://umbc.edu/~finin/talks/idm02/> <\text{http://purl.org/dc/elements/1.1/Cr}	ext{eator}> _{\text{j10949}} .\)
\(_{\text{j10949}} <\text{http://umbc.edu/~finin/talks/idm02/> <\text{http://www.w3.org/1999/02/22-rdf-syntax-ns#}\text{t}\text{ype}> <\text{Description}> .\)

Note the gensym for the anonymous node \(_{\text{j10949}}\).

N3 notation for RDF

- N3 is a compact notation for RDF that is easier for people to read, write and edit.
- Aka Notation 3, developed by TBL himself.
- Translators exist between N3 and the XML encoding, such as the web form on
  \- http://www.w3.org/DesignIssues/Notation3.html
- So, it's just "syntactic sugar"
- But, XML is largely unreadable and even harder to write

Triple Notes

- RDF triples have one of two forms:
  \- <\text{URI}> <\text{URI}> <\text{URI}>
  \- <\text{URI}> <\text{URI}> <\text{quoted string}>

- Triples are also easily mapped into logic
  \- <\text{subject}> <\text{predicate}> <\text{object}> becoming:
    \- <\text{predicate}>(<\text{subject}>,<\text{object}>)
    \- With type(<\text{S}>,<\text{O}>) becoming <\text{O}>(<\text{S}>)
  \- Example:
    \- subclass(man,person) ; Note: we're not showing the actual URIs for clarity
    \- sex(man,male)
    \- domain(sex,animal)
    \- man(adam)
    \- age(adam,100)

- Triples are easily stored and managed in DBMS
  \- Flat nature of a triple a good match for relational DBs

N3 Example

@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# .
@prefix dc: http://purl.org/dc/elements/1.1/ .
@prefix bib: http://daml.umbc.edu/ontologies/bib/ .

\< \text{http://umbc.edu/~finin/talks/idm02/> >

\text{dc:Title} "\text{Intelligent Information Systems on the Web}" ;
\text{dc:Creator}
  [ \text{bib:Name} "\text{Tim Finin}" ;
    \text{bib:Email} \text{finin@umbc.edu} ;
    \text{bib:Aff} "\text{http://umbc.edu/}" ] .

Note special […] syntax for an anonymous node
RDF types

- RDF has a trivial type system
- RDFS and OWL extend it greatly

RDF Container Elements

- rdf:Bag
  - unordered
  - may contain multiple occurrences
- rdf:Seq
  - ordered
  - may contain multiple occurrences
- rdf:Alt
  - a set of alternatives
- Content of container elements are named rdf:_1, rdf:_2, ...
- Containers seem a bit messy in RDF, but are needed

RDF Container Example

RDF types – Syntax

- This abbreviated syntax is very common
Bags and Seqs are never full!

- RDF's semantics is "open world", so…
  - There is no possibility "to close" the container, to say: "these are all elements, there are no more"
  - RDF is a graph, so: there is no way to exclude the possibility that there is another graph somewhere that describes additional members
- Collections for groups with only the specified members are described via a predefined collection vocabulary of the types:
  - rdf:List, rdf:first, rdf:rest, rdf:nil

Yuck!

RDF Lists

CIT 2112 is exclusively taught by teachers 949111, 949352, 949381

```xml
<rdf:Description rdf:about="CIT2112">
  <uni:isTaughtBy>
    <rdf:List>
      <rdf:first><rdf:Description rdf:about="949111" /></rdf:first>
      <rdf:rest><rdf:List>
        <rdf:first><rdf:Description rdf:about="949352" /></rdf:first>
        <rdf:rest><rdf:List>
          <rdf:first><rdf:Description rdf:about="949381" /></rdf:first>
          <rdf:rest><rdf:Description rdf:about="&rdf;nil" /></rdf:rest>
        </rdf:List>
      </rdf:rest>
    </rdf:List>
  </uni:isTaughtBy>
</rdf:Description>
```

RDF Lists Syntactic Sugar

The the rdf:parseType attribute helps

```xml
<rdf:Description rdf:about="CIT2112">
  <uni:isTaughtBy rdf:parseType="Collection">
    <rdf:Description rdf:about="949111" />
    <rdf:Description rdf:about="949352" />
    <rdf:Description rdf:about="949381" />
  </uni:isTaughtBy>
</rdf:Description>
```

Reification

The description

```xml
<rdf:Description rdf:about="949318">
  <uni:name>David Billington</uni:name>
</rdf:Description>
```

reifies to

```xml
<rdf:Statement rdf:about="StatementAbout949318">
  <rdf:subject rdf:resource="949318" />
  <rdf:predicate rdf:resource="&uni;name" />
  <rdf:object rdf:resource="David Billington" />
</rdf:Statement>
```

- The statement ID can be used to refer to it
- If more than one property elements is contained in a description element: they belong to more than one statement! In this case, these statements can be places in a bag, or can be reified separately
## Conclusions

- RDF is a simple data model based on a graph
  - Independent on any serialization (e.g., XML or N3)
- RDF has a formal semantics providing a dependable basis for reasoning about the meaning of RDF expressions
- RDF has an extensible URI-based vocabulary
- RDF has an XML serialization and can use values represented as XML schema datatypes
- Anyone can make statements about any resource (open world assumption)
- RDFS and OWL build on RDF’s foundation by adding vocabulary with well defined semantics (e.g., Class, subClass, etc.)