Chapter 3
RDF Syntax

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

RDF Overview

- RDF Syntax -- the XML encoding
- RDF Syntax – variations including N3
- RDF Schema (RDFS)
- Semantics of RDF and RDFS
  - Axiomatic Semantics
  - Operational semantics based on rules
- Querying RDF via RQL and SPARQL

Example

<academicStaffMember> Grigoris Antoniou </academicStaffMember>
<professor> Michael Maher </professor>
<course name="Discrete Mathematics">
  <isTaughtBy> David Billington </isTaughtBy>
</course>

- 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
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?

RDF is the first SW language

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.

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, … `xml:datatypes`
- **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”

http://umbc.edu/~finin/  dc:Creator  “Tim Finin”

http://umbc.edu/~finin/  bib:Aff  “finin@umbc.edu”

http://umbc.edu/  dc:Title  “Intelligent Information Systems on the Web”

http://umbc.edu/  dc:Creator  “Tim Finin”

http://umbc.edu/  bib:Aff  “finin@umbc.edu”
```

http://umbc.edu/  dc:Title  “Intelligent Information Systems on the Web”

http://umbc.edu/  dc:Creator  “Tim Finin”

http://umbc.edu/  bib:Aff  “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 \texttt{give(John,Mary,Book32)} as three binary relations all involving a common object, \texttt{giveEvent32}
  - \texttt{giver(giveEvent45 , John )}
  - \texttt{recipient( giveEvent45 , Mary )}
  - \texttt{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 based on binary relations

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 \texttt{John gave Mary a book}
  - "John gave Mary a Book" has 0.33 probability

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/">
  <rdf:Description about="http://umbc.edu/~finin/talks/idm02/">
    <dc:title>Intelligent Information Systems on the Web</dc:title>
    <dc:creator>
      <rdf:Description>
        <bib:name>Tim Finin</bib:name>
        <bib:email>finin@umbc.edu</bib:email>
        <bib:aff resource="http://umbc.edu/"/>
      </rdf:Description>
    </dc:creator>
  </rdf: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
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>
```

- An empty prefix means that this is the default namespace for the document
- Any non-literal symbols without a prefix are in this namespace
  E.g., `<Description>`

Descriptions

- Every description makes a statement about a resource
- There are different ways:
  - an about attribute: referencing to an existing resource
    `<rdf:Description rdf:about="http..."> ...`
  - an id attribute: creating a new resource
    `<rdf:Description rdf:ID="foo3456"> ...`
  - without a name: creating an anonymous resource
    `<rdf:Description> ...`

rdf:about versus rdf:ID

- An element `<rdf:Description>` has
  - an `<rdf:about>` attribute indicating that the resource has been “defined” elsewhere
  - An `<rdf:ID>` attribute indicating that the resource is defined
- Formally, there is no such thing as “defining” an object in one place and referring to it elsewhere
  - Sometimes is useful (for human readability) to have a defining location, while other locations state “additional” properties
- A Description with neither produces a “blank node”
  - It can not be referred to either from within or outside the rdf document

• 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
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#" >
  <rdf:Description about="http://umbc.edu/~finin/talks/idm02/">
    <dc:title>Intelligent Information Systems on the Web </dc:Title>
    <dc:creator>
      <rdf:Description>
        <bib:name>Tim Finin</bib:name>
        <bib:email>finin@umbc.edu</bib:Email>
        <bib:aff resource="http://umbc.edu/" />
      </rdf:Description>
    </dc:creator>
  </rdf:Description>
</rdf:RDF>
```

- dc:title is 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

```xml
• ex:age rdf:datatype="&xsd;integer"> 22 </ex:age>
```

- The value of creator is defined by the nested RDF
- The nameless description produces a “blank node”istent Information Systems on the Web”
- This style of XML encoding is called “striped”

N triple representation

```xml
RDF can be encoded as a set of triples.  
<subject> <predicate> <object> .

<http://umbc.edu/~finin/talks/idm02/> <http://purl.org/dc/elements/1.1/Title>  
"Intelligent Information Systems on the Web" .
_110949 <http://daml.umbc.edu/ontologies/bib/Name> "Tim Finin" .
_110949 <http://daml.umbc.edu/ontologies/bib/Email> "finin@umbc.edu" .
_110949 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type><Description> .
<http://umbc.edu/~finin/talks/idm02/> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type><Description> .
```

Note the gensym for the anonymous node _110949.
Triple Notes

- RDF triples have one of two forms:
  - `<URI> <URI> <URI>`
  - `<URI> <URI> <quoted string>`

- Triples are also easily mapped into logic
  - `<subject> <predicate> <object>` becoming:
    - `<predicate>(<subject>,<object>)`
    - With type(<S>,<O>) becoming `<O>(<S>)`
  - Example:
    - `subclass(man,person)`
    - `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 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

Example of University Courses

```xml
@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/ .
< http://umbc.edu/~finin/talks/idm02/ >
dc:title "Intelligent Information Systems on the Web" ;
dc:creator
  [ bib:Name "Tim Finin" ;
    bib:Email finin@umbc.edu ;
```

N3 Example

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:uni="http://www.mydomain.org/uni-ns">
  <rdf:Description rdf:about="949318">
    <uni:name>David Billington</uni:name>
    <uni:title>Associate Professor</uni:title>
    <uni:age rdf:datatype="&xsd:integer">27</uni:age>
  </rdf:Description>
```
Example of University Courses (2)

```xml
<rdf:Description rdf:about="CIT1111">
  <uni:courseName>Discrete Maths</uni:courseName>
  <uni:isTaughtBy>David Billington</uni:isTaughtBy>
</rdf:Description>

<rdf:Description rdf:about="CIT2112">
  <uni:courseName>Programming III</uni:courseName>
  <uni:isTaughtBy>Michael Maher</uni:isTaughtBy>
</rdf:Description>
</rdf:RDF>

Data Types for Literals

- Data types are used in programming languages to allow interpretation
- In RDF, typed literals are used
- You can specify this with a special ^^ syntax
  
  ```xml
  
  or using the rdf:datatype attribute
  ```

XMLSchema Datatypes

- ^^^-notation indicates the type of a literal
- In practice, the most widely used data typing scheme will be the one by XML Schema
  - But the use of any externally defined data typing scheme is allowed in RDF documents
- XML Schema predefines a large range of data types
  - E.g. Booleans, integers, floating-point numbers, times, dates, etc.

http://www.w3.org/TR/xmlschema-2/
The rdf:resource Attribute

- The relationships between courses and lecturers (in the example) were not formally defined but existed implicitly through the use of the same name
- The use of the same name may just be a coincidence for a machine
- We can denote that two entities are the same using the rdf:resource attribute
- By design, RDF explicitly rules out the “unique name assumption” common in many KR systems

Referencing Externally Defined Resources

- Refer to the externally defined resource CIT1111 using http://www.mydomain.org/uni-ns#CIT1111 as the value of rdf:about
- Assuming that www.mydomain.org/uni-ns is the URI where the definition of CIT1111 is found
- A description with an ID defines a fragment URI, which can be used to reference the defined description

The rdf:resource Attribute

```xml
<rdf:Description rdf:about="CIT1111">
  <uni:courseName>Discrete Mathematics</uni:courseName>
  <uni:isTaughtBy rdf:resource="949318"/>
</rdf:Description>

<rdf:Description rdf:about="949318">
  <uni:name>David Billington</uni:name>
  <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

Nested Descriptions: Example

```xml
<rdf:Description rdf:about="CIT1111">
  <uni:courseName>Discrete Maths</uni:courseName>
  <uni:isTaughtBy>
    <rdf:Description rdf:ID="949318">
      <uni:name>David Billington</uni:name>
      <uni:title>Associate Professor</uni:title>
    </rdf:Description>
  </uni:isTaughtBy>
</rdf:Description>
```
Nested Descriptions

- Descriptions may be defined within other descriptions
- Other courses, such as CIT3112, can still refer to the new resource with ID 949318
- Although a description may be defined within another description, its scope is global

RDF types

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

RDF types, another syntax

```xml
<rdf:Description rdf:ID="CIT1111">
  <rdf:type rdf:resource="http://www.mydomain.org/uni-ns#course"/>
  <uni:courseName>Discrete Maths</uni:courseName>
  <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>

<rdf:Description rdf:ID="949318">
  <rdf:type rdf:resource="http://www.mydomain.org/uni-ns#lecturer"/>
  <uni:name>David Billington</uni:name>
  <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

RDF types, yet another Syntax

```xml
<uni:course rdf:ID="CIT1111">
  <uni:courseName>Discrete Mathematics</uni:courseName>
  <uni:isTaughtBy rdf:resource="949318"/>
</uni:course>

<uni:lecturer rdf:ID="949318">
  <uni:name>David Billington</uni:name>
  <uni:title>Associate Professor</uni:title>
</uni:lecturer>
```

- This abbreviated syntax is very common
Abbreviated Syntax

So we have two simplification rules:
1. Childless property elements within description elements may be replaced by XML attributes
2. For description elements with a typing element we can use the name specified in the rdf:type element instead of rdf:Description

These rules create syntactic variations of the same RDF statement
- They are equivalent according to the RDF data model, although they have different XML syntax

Abbreviated Syntax: Example

<rdf:Description rdf:id="CIT1111">
  <rdf:type rdf:resource="http://www.mydomain.org/uni-ns#course"/>
  <uni:courseName>Discrete Maths</uni:courseName>
  <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>

Application of First Simplification Rule

.rdf:Description rdf:id="CIT1111"
  uni:courseName="Discrete Maths">
  <rdf:type rdf:resource="http://www.mydomain.org/uni-ns#course"/>
  <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>

Application of 2nd Simplification Rule

<uni:course rdf:id="CIT1111"
  uni:courseName="Discrete Maths">
  <uni:isTaughtBy rdf:resource="#949318"/>
</uni:course>
Container Elements

- Collect a number of resources or attributes about which we want to make statements as a whole
- E.g., we may wish to talk about the courses given by a particular lecturer
- The content of container elements are named `rdf:_1`, `rdf:_2`, etc.
  - Alternatively `rdf:li`
- Containers seem a bit messy in RDF, but are needed

Three Types of Container Elements

- `rdf:Bag` an unordered container, allowing multiple occurrences
  - E.g. members of the faculty board, documents in a folder
- `rdf:Seq` an ordered container, which may contain multiple occurrences
  - E.g. modules of a course, items on an agenda, an alphabetized list of staff members (order is imposed)
- `rdf:Alt` a set of alternatives
  - E.g. the document home and mirrors, translations of a document in various languages

Example for a Bag

```
<uni:lecturer
  rdf:ID="949352" uni:name="Grigoris Antoniou"
  uni:title="Professor">
  <uni:coursesTaught>
    <rdf:Bag>
      <rdf:_1 rdf:resource="#CIT1112"/>
      <rdf:_2 rdf:resource="#CIT3116"/>
    </rdf:Bag>
  </uni:coursesTaught>
</uni:lecturer>
```

Example for Alternative

```
<uni:course rdf:ID="CIT1111"
  uni:courseName="Discrete Mathematics">
  <uni:lecturer>
    <rdf:Alt>
      <rdf:li rdf:resource="#949352"/>
      <rdf:li rdf:resource="#949318"/>
    </rdf:Alt>
  </uni:lecturer>
</uni:course>
```
Rdf:ID Attribute for Container Elements

```xml
<uni:lecturer rdf:id="949318"
    uni:name="David Billington">
    <uni:coursesTaught>
        <rdf:Bag rdf:id="DBcourses">
            <rdf:_1 rdf:resource="#CIT1111"/>
            <rdf:_2 rdf:resource="#CIT3112"/>
        </rdf:Bag>
    </uni:coursesTaught>
</uni:lecturer>
```

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

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:uni="http://www.mydomain.org/#">
    <uni:lecturer rdf:about="949352" uni:name="Grigoris Antoniou" uni:title="Professor">
        <uni:coursesTaught>
            <rdf:Bag>
                <rdf:_1 rdf:resource="#CIT1112"/>
                <rdf:_2 rdf:resource="#CIT1113"/>
            </rdf:Bag>
        </uni:coursesTaught>
    </uni:lecturer>
    <uni:course rdf:about="CIT1111" uni:courseName="Discrete Mathematics">
        <uni:lecturer>
            <rdf:Alt>
                <rdf:_1 rdf:resource="949352"/>
                <rdf:_2 rdf:resource="949318"/>
            </rdf:Alt>
        </uni:lecturer>
    </uni:course>
</rdf:RDF>
```

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
RDF Lists

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

<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="949318"/></rdf:first>
              <rdf:rest><rdf:Description rdf:about="&rdf;nil"></rdf:rest>
            </rdf:List>
          </rdf:rest>
        </rdf:List>
      </rdf:rest>
    </rdf:List>
  </uni:isTaughtBy>
</rdf:Description>

RDF Lists Syntactic Sugar

The the rdf:parseType attribute helps

<rdf:Description rdf:about="CIT2112">
  <uni:isTaughtBy rdf:parseType="Collection">
    <rdf:Description rdf:about="949111"/>
    <rdf:Description rdf:about="949352"/>
    <rdf:Description rdf:about="949318"/>
  </uni:isTaughtBy>
</rdf:Description>

Reification

- Sometimes we wish to make statements about other statements
- We must be able to refer to a statement using an identifier
- RDF allows such reference through a reification mechanism which turns a statement into a resource

Reify

- Etymology: Latin res thing
- Date: 1854
- to regard (something abstract) as a material or concrete thing
Reification is the act of making an abstract concept or low-level implementation detail of a programming language accessible to the programmer, often as a first-class object. For example,
- The C programming language reifies the low-level detail of memory addresses.
- The Scheme programming language reifies continuations (approximately, the call stack).
- In C#, reification is used to make parametric polymorphism implemented as generics a first-class feature of the language.
- ...
RDF Critique: Binary Predicates

- RDF uses only binary properties
  - This is a restriction because often we use predicates with more than 2 arguments
  - But binary predicates can simulate these
- Example: \( \text{referee}(X,Y,Z) \)
  - \( X \) is the referee in a chess game between players \( Y \) and \( Z \)

RDF Critique: Reification

- The reification mechanism is quite powerful
- It appears misplaced in a simple language like RDF
- Making statements about statements introduces a level of complexity that is not necessary for a basic layer of the Semantic Web
- Instead, it would have appeared more natural to include it in more powerful layers, which provide richer representational capabilities

RDF Critique: Binary Predicates

- We introduce:
  - a new auxiliary resource \( \text{chessGame} \)
  - the binary predicates \( \text{ref}, \text{player1}, \text{and player2} \)
- We can represent \( \text{referee}(X,Y,Z) \) as:

```
chessGame
/\      |
|       |
\referee
```

RDF Critique: Graph Representation

- The simple graph or network representation has more drawbacks
- Linear languages introduce ways to represent this with parentheses or a way to represent a block structure
- Scoping, for example, is clumsy at best in RDF
- Some of these are addressed through the notion of a \textit{named graph} in RDF
RDF Critique: Summary

- RDF has its idiosyncrasies and is not an optimal modeling language **but**
- It is already a de facto standard
- It has sufficient expressive power
  - At least as for more layers to build on top
- Using RDF offers the benefit that information maps unambiguously to a model