Chapter 3
RDF Syntax

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

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

```
<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

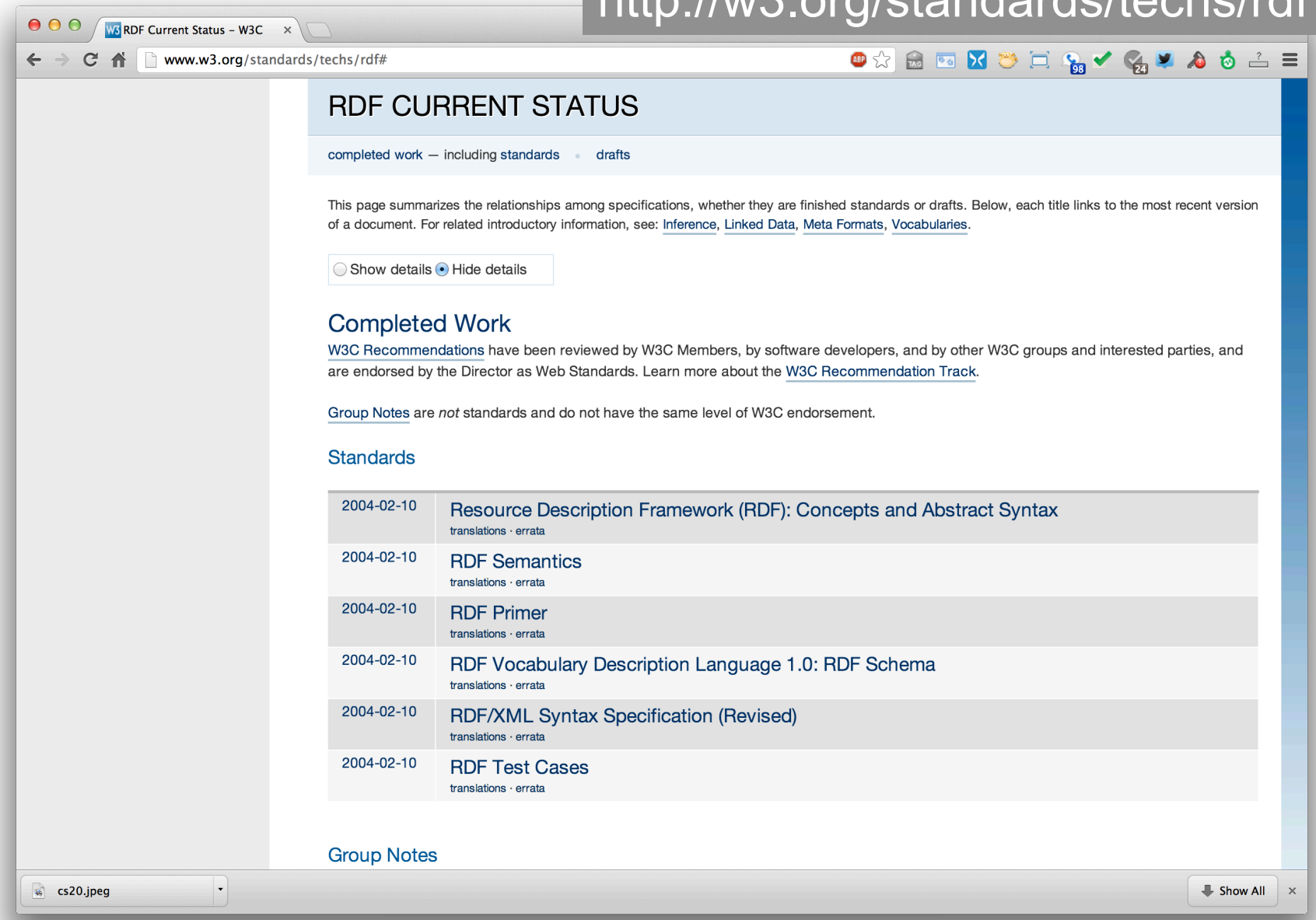
Example

```
<course name="Discrete Mathematics">
   <lecturer>David Billington</lecturer>
</course>
<lecturer name="David Billington">
   <teaches>Discrete Mathematics</teaches>
</lecturer>
<teachingOffering>
    <lecturer>David Billington</lecturer>
    <course>Discrete Mathematics</course>
</teachingOffering>
```

- 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?

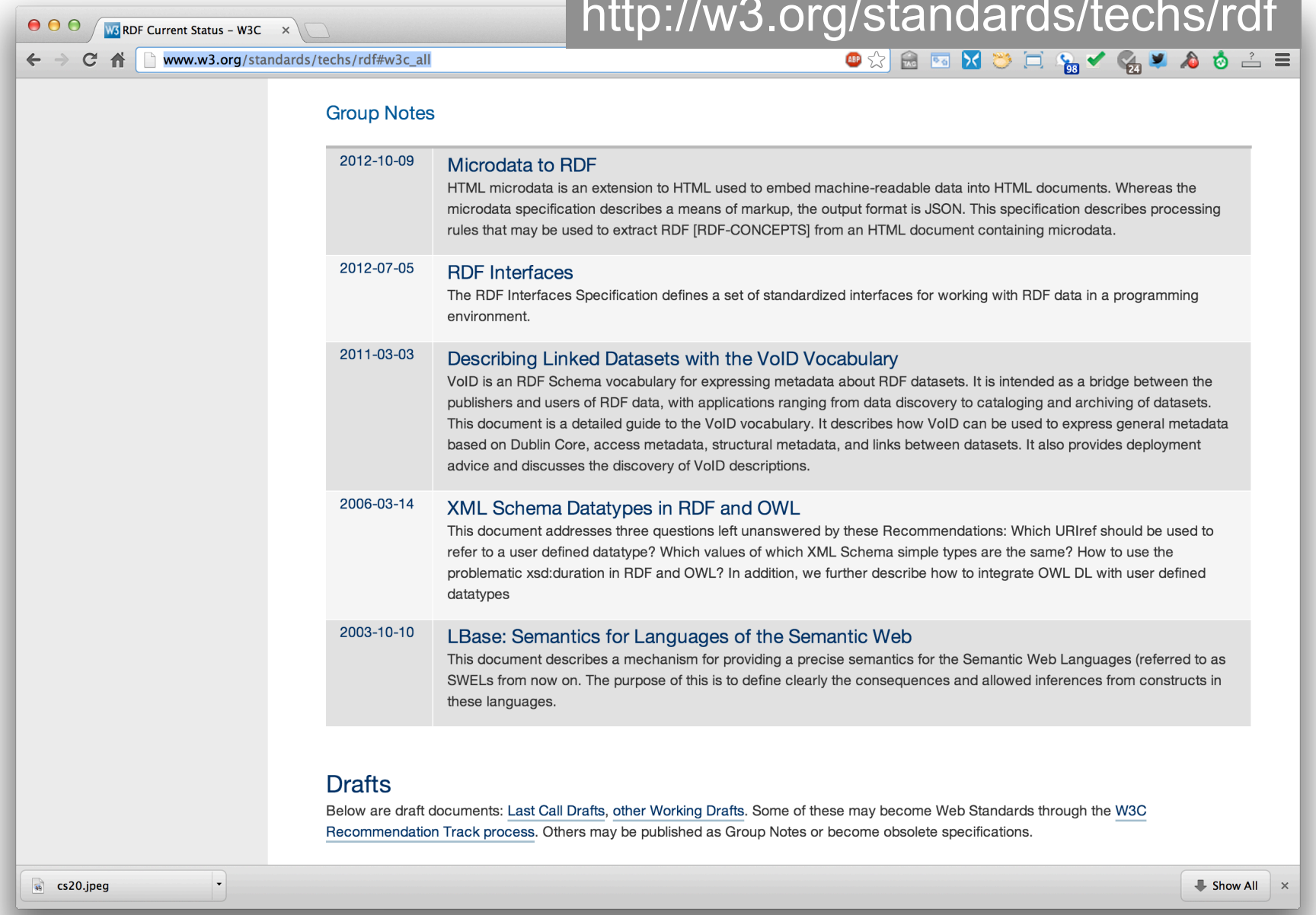
Key RDF documents: standards

http://w3.org/standards/techs/rdf



Key RDF documents: notes

http://w3.org/standards/techs/rdf



Group Notes

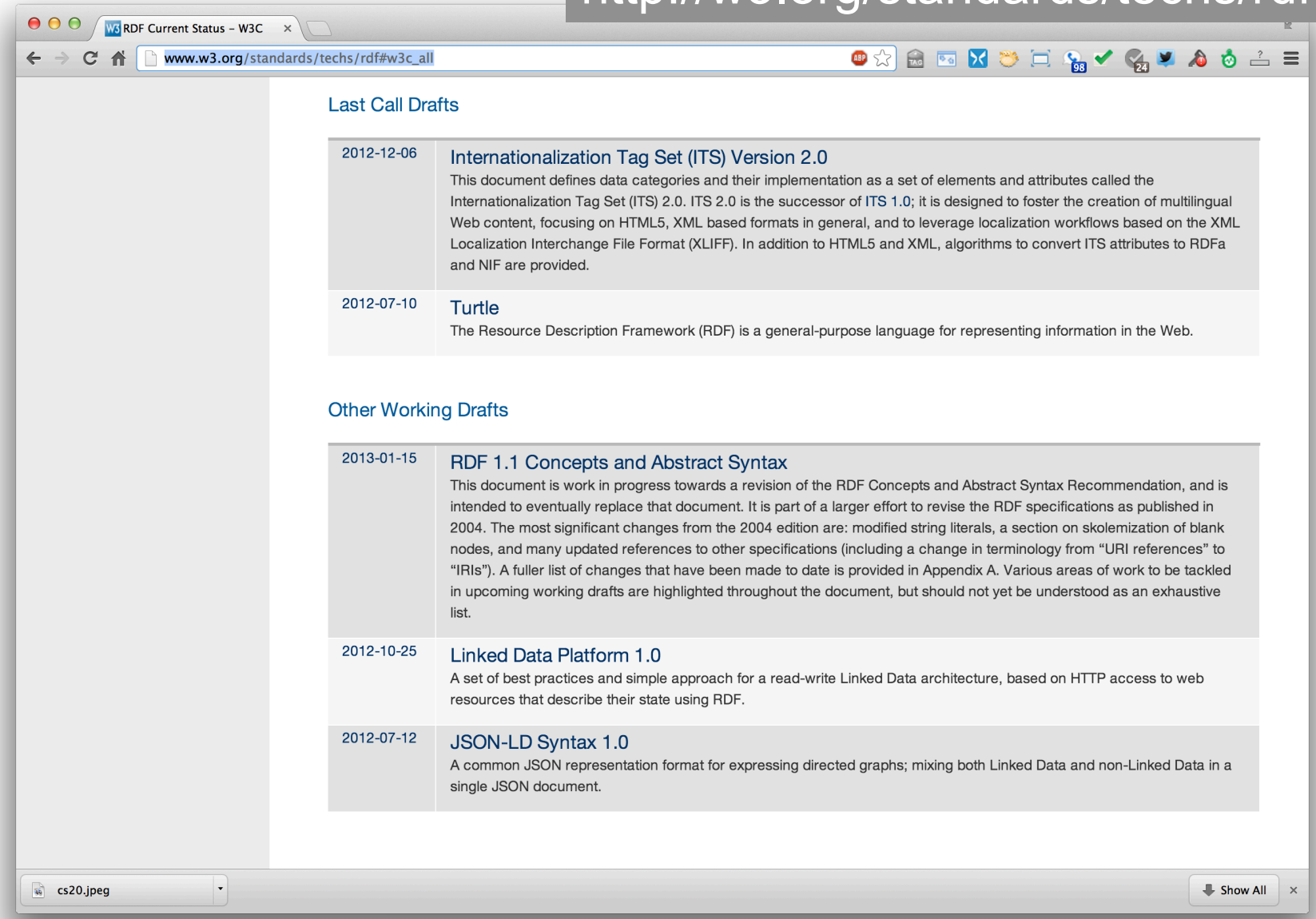
2012-10-09	**Microdata to RDF** HTML microdata is an extension to HTML used to embed machine-readable data into HTML documents. Whereas the microdata specification describes a means of markup, the output format is JSON. This specification describes processing rules that may be used to extract RDF [RDF-CONCEPTS] from an HTML document containing microdata.
2012-07-05	**RDF Interfaces** The RDF Interfaces Specification defines a set of standardized interfaces for working with RDF data in a programming environment.
2011-03-03	**Describing Linked Datasets with the VoID Vocabulary** VoID is an RDF Schema vocabulary for expressing metadata about RDF datasets. It is intended as a bridge between the publishers and users of RDF data, with applications ranging from data discovery to cataloging and archiving of datasets. This document is a detailed guide to the VoID vocabulary. It describes how VoID can be used to express general metadata based on Dublin Core, access metadata, structural metadata, and links between datasets. It also provides deployment advice and discusses the discovery of VoID descriptions.
2006-03-14	**XML Schema Datatypes in RDF and OWL** This document addresses three questions left unanswered by these Recommendations: Which URIref should be used to refer to a user defined datatype? Which values of which XML Schema simple types are the same? How to use the problematic xsd:duration in RDF and OWL? In addition, we further describe how to integrate OWL DL with user defined datatypes
2003-10-10	**LBase: Semantics for Languages of the Semantic Web** This document describes a mechanism for providing a precise semantics for the Semantic Web Languages (referred to as SWELs from now on. The purpose of this is to define clearly the consequences and allowed inferences from constructs in these languages.

Drafts

Below are draft documents: Last Call Drafts, other Working Drafts. Some of these may become Web Standards through the W3C Recommendation Track process. Others may be published as Group Notes or become obsolete specifications.

Key RDF documents: drafts

http://w3.org/standards/techs/rdf



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 referenced 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
- A graph defined by a collection of triples

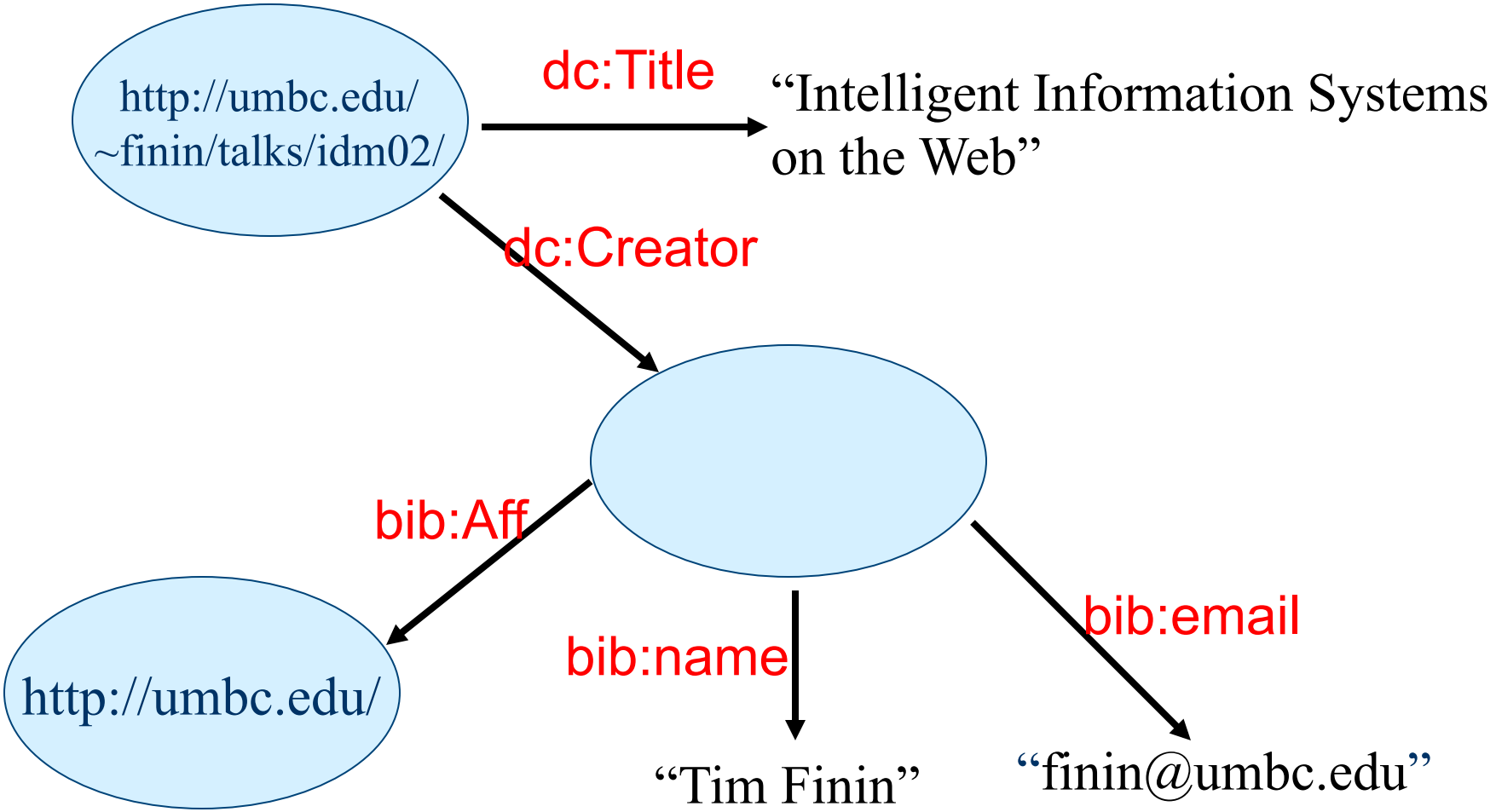
URIs are a foundation

- 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



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 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 *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/">
<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>
</rdfdescription>
</rdf:RDF>
```

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/">
<rdf:Description about="http://umb .edu/~finin/talks/idm02/">
  <dc:title>Intelligent Information  stems on the Web </dc:Title>
  <dc:creator>
    <rdf:Description >
      <bib:name>Tim Finin<                    >
      <bib:email>f
      <bib:aff reso
    </rdf:Descript
  </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 vocabulary
- One for the Bib vocabulary

XML encoding for RDF

```
<rdf:RDF xmlns:="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/www02/">
  <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>

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/">
<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:Na
      <bib:email>finin@um
      <bib:aff resource="htt
    </rdf:Description>
  </dc:creator>
</rdf:Description>
</rdf:RDF>
```

- 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 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 outside the rdf document

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/"
   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:Na
      <bib:email>finin@um
      <bib:aff resource="htt
    </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
  - <ex:age rdf:datatype="&xsd;integer> 22 </ex:age>
  - <ex:age> "27"^^xsd:integer> 22 </ex:age>

XML encoding for RDF

```
<rdf:RDF xmlns:rdf="
    xmlns:dc="http://pu
    xmlns:bib="http://d
    xmlns:xsd="http://ww
<rdf:Description ab
  <dc:title>Intellige
  <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>
```

- The value of creator is defined by the nested RDF
- The nameless description produces a “blank node”
- In this case, “a thing with a name=“Tim Finin” and …”
- This style of XML encoding is called “striped”

```
<thing>
  <property>
    <thing>
      <property>
```

XML encoding for RDF

```
<rdf:RDF xmlns:rdf="
    xmlns:dc="http://pu
    xmlns:bib="http://d
   xmlns:xsd="http://ww
<description about="
  <dc:title>Intelligent
  <dc:creator>
    <description >
      <bib:name>Tim Finin</b      me>
      <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?
  <bib:aff> http://umbc.edu/ </bib:aff>

N triple representation

- 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" .
_:j10949 <http://daml.umbc.edu/ontologies/bib/Name> "Tim Finin" .
_:j10949 <http://daml.umbc.edu/ontologies/bib/Email> "finin@umbc.edu" .
_:j10949 <http://daml.umbc.edu/ontologies/bib/Aff> <http://umbc.edu/> .
_:j10949 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type><Description> .
<http://umbc.edu/~finin/talks/idm02/> <http://purl.org/dc/elements/1.1/Creator> _:j10949 .
<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 _:j10949

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)

; Note: we're not
; showing the actual
; URIs for clarity

- **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

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/ .

< 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 ;
        bib:Aff: "http://umbc.edu/" ]   .
```

Note special […] syntax for an anonymous node

thing
  $prop_1$ = value ;
  $prop_2$ = value ;
  …
  $prop_n$ = value .

Example of University Courses

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XLMSchema#"
   xmlns:uni="http://example.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)

```
<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

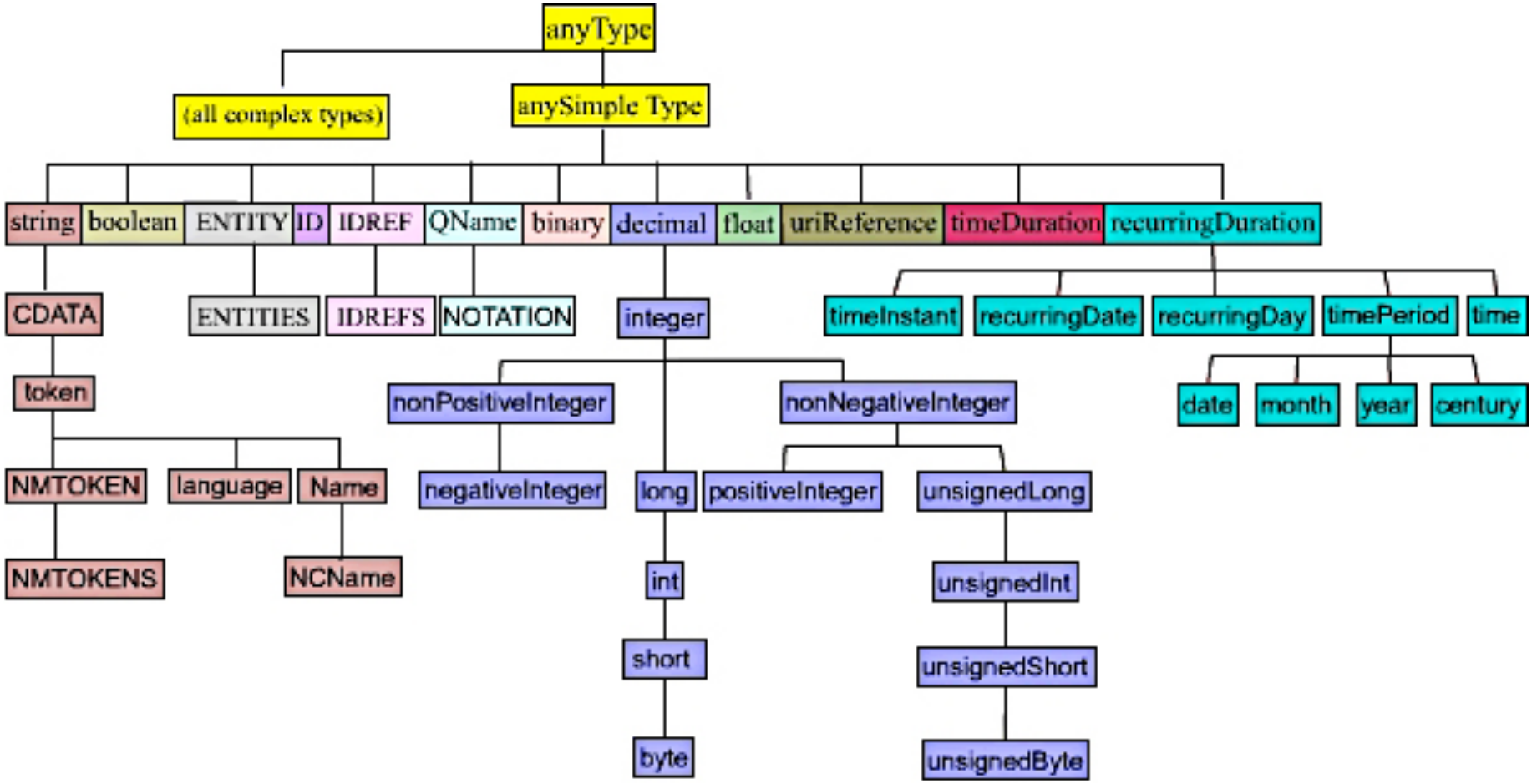
  (“David Billington”,
  http://example.org/age,
  “27”^^http://www.w3.org/2001XMLSchema#integer)
- or using the rdf:datatype attribute

  <uni:age rdf:datatype="&xsd:integer">27<uni:age>

Data Types for Literals

- **^^**-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.

XMLSchema Datatypes



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 via the use of the same name
- The use of the same name may just be a coincidence
- We can denote that two entities are the same using the **rdf:resource** attribute
  - Later we'll see that we can use an owl:sameAs assertion
- By design, RDF explicitly rules out the common *unique name assumption* found in many representation systems

The rdf:resource Attribute

```
<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>
```

Referencing Externally Defined Resources

- Refer to the externally defined resource CIT1111 using http://example.org/uni-ns#CIT1111 as the value of rdf:about
- Assuming that example.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

Nested Descriptions: Example

```
<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:Description rdf:about="CIT1111">
   <rdf:type rdf:resource="&uni:Course"/>
   <uni:courseName>Discrete Mathematics</uni:courseName>
   <uni:isTaughtBy rdf:resource="949318"/>
</rdf:Description>
<rdf:Description rdf:about="949318">
   <rdf:type rdf:resource="&uni:Lecturer"/>
   <uni:name>David Billington</uni:name>
   <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

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

RDF types, another syntax

```
<rdf:Description rdf:ID="CIT1111">
   <rdf:type rdf:resource="http://example.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://example.org/uni-ns#lecturer"/>
   <uni:name>David Billington</uni:name>
   <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

RDF types, yet another Syntax

```
<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://example.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://example.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

```
<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>
```

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>
```

Yuck!

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

Wikipedia: reification (computer science)

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.
- …

Reification Example

```
<rdf:Description rdf:about="#949352">
  <uni:name>Grigoris Antoniou</uni:name>
</rdf:Description>
```

reifies as

```
<rdf:Statement rdf:ID="StatementAbout949352">
  <rdf:subject rdf:resource="#949352"/>
  <rdf:predicate rdf:resource="http://example.org/uni-
     ns#name"/>
  <rdf:object>Grigoris Antoniou</rdf:object>
</rdf:Statement>
```

Reification

- **rdf:subject, rdf:predicate** and **rdf:object** allow us to access the parts of a statement
- The **ID** of the statement can be used to refer to it, as can be done for any description
- We write an **rdf:Description** if we don’t want to talk about a statement further
- We write an **rdf:Statement** if we wish to refer to a statement

RDF Critique: Properties

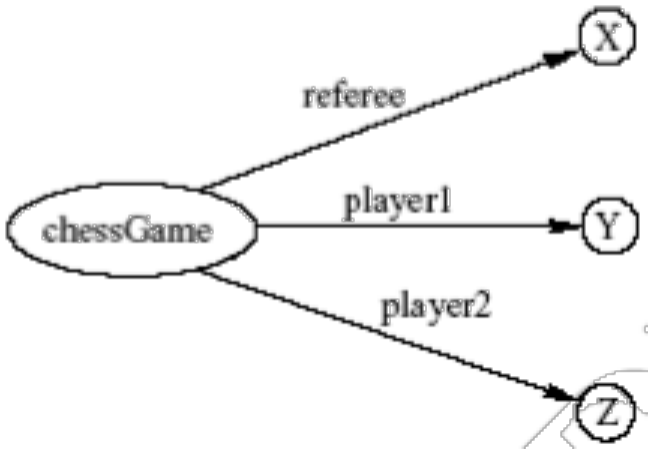
- Properties are special kinds of resources
  - Properties can be used as the object in an object-attribute-value triple (statement)
  - They are defined **independent** of resources
- This possibility offers flexibility
- But it is unusual for modelling languages and OO programming languages
- It can be confusing for modellers

RDF Critique: Binary Predicates

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

RDF Critique: Binary Predicates

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



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: 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 *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