Tutorial on the Semantic Web

Ivan Herman, W3C

(Last update: 25 November 2008)

Introduction

Copyright © 2008, W3C

Towards a Semantic Web

- The current Web represents information using
  - natural language (English, Hungarian, Chinese,…)
  - graphics, multimedia, page layout
- Humans can process this easily
  - can deduce facts from partial information
  - can create mental associations
  - are used to various sensory information
    - (well, sort of… people with disabilities may have serious problems on the Web with rich media!)

Copyright © 2008, W3C

Towards a Semantic Web

- Tasks often require to combine data on the Web:
  - hotel and travel information may come from different sites
  - searches in different digital libraries
  - etc.
- Again, humans combine these information easily
  - even if different terminologies are used!

Copyright © 2008, W3C

However…

- However: machines are ignorant!
  - partial information is unusable
  - difficult to make sense from, e.g., an image
  - drawing analogies automatically is difficult
  - difficult to combine information automatically
    - is `<foo:creator>` same as `<bar:author>`?
  - …

Copyright © 2008, W3C

Example: automatic airline reservation

- Your automatic airline reservation
  - knows about your preferences
  - builds up knowledge base using your past
  - can combine the local knowledge with remote services:
    - airline preferences
    - dietary requirements
    - calendaring
    - etc
- It communicates with remote information
  - (M. Dertouzos: The Unfinished Revolution)

Copyright © 2008, W3C

Example: data(base) integration

- Databases are very different in structure, in content
- Lots of applications require managing several databases
  - after company mergers
  - combination of administrative data for e-Government
  - biochemical, genetic, pharmaceutical research
  - etc.
- Most of these data are accessible from the Web (though not necessarily public yet)

Copyright © 2008, W3C

And the problem *is* real…

CoCoDat - Collation of Cortical Data - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.cocom

CoCoMac DATABASES ORT EXAMPLES

CoCoDat: Collation of Cortical [s
microcircuitry] Data

CoCoDat is a microcircuitry database that
published experimental reports. The data
and cellular compartment), as well as the

- Morphology
- Firing properties
- Ionic currents
- Ionic conductances
- Synaptic currents
- Connectivity

The database is available for download u
data tables but also a Search Board with
manual or automatic relaxation of the sea

- Brain region
- Layer
- Neuron type

http://www.cocomac.org/cocodat/catalyzer/inc

Cell Centered Database - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://ccdb.ucsd.edu/sand/main?event=gallery&action=show&dpl=y

red database

Cell Centered Database™
National Center for Microscopy and Imaging Research
Gallery

Data | Search | Gallery | Dictionary | Publications | MyCCDB | Data Download | Contact us | Help

2D image | Reconstruction | Segmentation | Animation

y 23
: 071202tjd
: P1188
t Brain Maps
t Tom Deerinck
t BioRad RTS 2000MP
: Multiphoton
: montage of cerebellar cortex triple labeled for Hoescht stain (blue), IP3 receptor (green) and GFAP (red)

logged out

NeuronDB - Thalamic relay neuron - Overview (A) () - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://senselab.med.yale.edu

senselab

NeuronDB

Back

Thalamic relay neuron

Mode: Overview | Data/Search | plus Connectivity | plus Classical References/Notes | Models

Region: Distal equivalent dendrite | Middle equivalent dendrite | Proximal equivalent dendrite | Soma | Axon hillock | Axon fiber | Axon terminal | All Compartments

Properties: Receptors | Channels | Transmitters | All Properties

Interoperation: Gene and Chromosome | Experimental Data (neurodatabase.org) | Microscopy Data (CCDB)

Neuron type: principal
Organism: Vertebrates

1. Equivalent dendrite — Show other
2. Distal equivalent dendrite — Show other
3. Middle equivalent dendrite — Show other
4. Proximal equivalent dendrite — Show other
5. Soma — Show other

Done

logged out

Example: social networks

- Social sites are everywhere these days (LinkedIn, Facebook, Dopplr, Digg, Plexo, Zyb, …)
- How many times did you have to add your contacts?
- Applications should be able to get to those data via standard means
  - there are, of course, privacy issues…

Copyright © 2008, W3C

Example: digital libraries

- It means catalogs on the Web
  - librarians have known how to do that for centuries
  - goal is to have this on the Web, World-wide
  - extend it to multimedia data, too
- But it is more: software agents should also be librarians!
  - help you in finding the right publications

Copyright © 2008, W3C

Example: semantics of Web Services

- Web services technology is great
- But if services are ubiquitous, searching issue comes up, for example:
  - “find me the best differential equation solver”
  - “check if it can be combined with the XYZ plotter service”
- It is necessary to characterize the service
  - not only in terms of input and output parameters…
  - …but also in terms of its semantics

Copyright © 2008, W3C

What is needed?

- (Some) data should be available for machines for further processing
- Data should be possibly combined, merged on a Web scale
- Sometimes, data may describe other data…
- … but sometimes the data is to be exchanged by itself, like my calendar or my travel preferences
- Machines may also need to *reason* about that data

Copyright © 2008, W3C

In what follows…

- We will use a simplistic example to introduce the main Semantic Web concepts
- We take, as an example area, data integration

Copyright © 2008, W3C

The rough structure of data integration

1. Map the various data onto an abstract data representation
   - make the data independent of its internal representation…
2. Merge the resulting representations
3. Start making queries on the whole!
   - queries not possible on the individual data sets

Copyright © 2008, W3C

A *simplified* bookstore data (dataset "A")

ID	Author	Title	Publisher	Year
ISBN0-00-651409-X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Home Page
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

ID	Publ. Name	City
id_qpr	Harpers Collins	London

Copyright © 2008, W3C

1st: export your data as a set of *relations*

Copyright © 2008, W3C

Some notes on the exporting the data

- Data export does *not* necessarily mean physical conversion of the data
  - relations can be generated on-the-fly at query time
    - via SQL “bridges”
    - scraping HTML pages
    - extracting data from Excel sheets
    - etc.
- One can export *part* of the data

Copyright © 2008, W3C

Another bookstore data (dataset “F”)

	A	B	C	D	E
1	**ID**	**Titre**	**Auteur**	**Traducteur**	**Original**
2	ISBN0 2020386682	Le Palais des miroirs	A7	A8	ISBN-0-00-651409-X
3					
4					
5					
6	**Nom**				
7	Ghosh, Amitav				
8	Besse, Christianne				

Copyright © 2008, W3C

2nd: export your second set of data

Copyright © 2008, W3C

3rd: start merging your data

Copyright © 2008, W3C

3rd: start merging your data (cont.)

Copyright © 2008, W3C

3rd: merge identical resources

Copyright © 2008, W3C

Start making queries…

- User of data “F” can now ask queries like:
  - “give me the title of the original”
- This information is not in the dataset “F”…
- …but can be retrieved by merging with dataset “A”!

Copyright © 2008, W3C

However, more can be achieved…

- We “feel” that **`a:author`** and **`f:auteur`** should be the same
- But an automatic merge does not know that!
- Let us add some extra information to the merged data:
  - **`a:author`** same as **`f:auteur`**
  - both identify a “Person”
  - a term that a community may have already defined:
    - a “Person” is uniquely identified by his/her name and, say, homepage
    - it can be used as a “category” for certain type of resources

Copyright © 2008, W3C

3rd revisited: use the extra knowledge

Start making richer queries!

- User of dataset “F” can now query:
  - “give me the home page of the original’s author”
- The information is not in datasets “F” or “A”…
- …but was made available by:
  - merging datasets “A” and datasets “F”
  - adding three simple extra statements as an extra “glue”

Copyright © 2008, W3C

Combine with different datasets

- Via, e.g., the “Person”, the dataset can be combined with other sources
- For example, data in Wikipedia can be extracted using dedicated tools

Copyright © 2008, W3C

Merge with Wikipedia data

Copyright © 2008, W3C

Merge with Wikipedia data

Copyright © 2008, W3C

Merge with Wikipedia data

Copyright © 2008, W3C

Is that surprising?

- Maybe but, in fact, no…
- What happened via automatic means is done every day by Web users!
- The difference: a bit of extra rigour so that machines could do this, too

Copyright © 2008, W3C

What did we do?

- We combined different datasets that
  - are somewhere on the web
  - are of different formats (mysql, excel sheet, XHTML, etc)
  - have different names for relations
- We could combine the data because some URI-s were identical (the ISBN-s in this case)
- We could add some simple additional information (the “glue”), also using common terminologies that a community has produced
- As a result, new relations could be found and retrieved

Copyright © 2008, W3C

It could become even more powerful

- We could add extra knowledge to the merged datasets
  - e.g., a full classification of various types of library data
  - geographical information
  - etc.
- This is where *ontologies*, extra *rules*, etc, come in
  - ontologies/rule sets can be relatively simple and small, or huge, or anything in between…
- Even more powerful queries can be asked as a result

Copyright © 2008, W3C

What did we do? (cont)

Applications

Query, Manipulate, etc.

Data represented in abstract format

Map, Expose, etc.

Data in various formats

Copyright © 2008, W3C

The abstraction pays off because…

- … the graph representation is independent of the exact structures
- … a change in local database schema's, XHTML structures, etc, do not affect the whole
  - "schema independence"
- … new data, new connections can be added seamlessly

Copyright © 2008, W3C

The network effect

- Through URI-s we can link any data to any data
- The “network effect” is extended to the (Web) data
- “Mashup on steroids” become possible

Copyright © 2008, W3C

So where is the Semantic Web?

- The Semantic Web provides technologies to make such integration possible!
- Hopefully you get a full picture at the end of the tutorial…

Copyright © 2008, W3C

Basic RDF

RDF triples

- Let us begin to formalize what we did!
  - we “connected” the data…
  - but a simple connection is not enough… it should be named somehow
  - hence the RDF Triples: *a labelled connection between two resources*

Copyright © 2008, W3C

RDF triples (cont.)

- An RDF Triple **(s,p,o)** is such that:
  - "**s**", "**p**" are URI-s, ie, resources on the Web; "**o**" is a URI or a literal
    - "**s**", "**p**", and "**o**" stand for "subject", "property", and "object"
  - here is the complete triple:

```
(<http://…isbn…6682>, <http://…/original>, <http://…isbn…409X>)
```

- *RDF* is a general model for such triples (with machine readable formats like RDF/XML, Turtle, N3, RXR, …)

Copyright © 2008, W3C

RDF triples (cont.)

- RDF triples are also referred to as “triplets”, or “statements”
- The “`p`” is also referred to as “predicate” in some cases

Copyright © 2008, W3C

RDF triples (cont.)

- Resources can use any URI; it can denote an element within an XML file on the Web, not only a “full” resource, e.g.:
  - `http://www.example.org/file.xml#element(home)`
  - `http://www.example.org/file.html#home`
  - `http://www.example.org/file2.xml#xpath1(//q[@a=b])`
- RDF triples form a directed, labelled graph (best way to think about them!)

Copyright © 2008, W3C

A simple RDF example (in RDF/XML)

```
<rdf:Description rdf:about="http://…/isbn/2020386682">
    <f:titre xml:lang="fr">Le palais des mirroirs</f:titre>
    <f:original rdf:resource="http://…/isbn/000651409X"/>
</rdf:Description>
```

(Note: namespaces are used to simplify the URI-s)

Copyright © 2008, W3C

A simple RDF example (in Turtle)

```
<http://…/isbn/2020386682>
    f:titre "Le palais des mirroirs"@fr ;
    f:original <http://…/isbn/000651409X> .
```

Copyright © 2008, W3C

URI-s play a fundamental role

- URI-s made the merge possible
- *URI-s ground RDF into the Web*
  - information can be retrieved using existing tools
  - this makes the “Semantic Web”, well… “Semantic *Web*”

Copyright © 2008, W3C

RDF/XML principles

- Encode nodes and edges as XML elements or with literals:

```
«Element for http://…/isbn/2020386682»
  «Element for original»
      «Element for http://…/isbn/000651409X»
  «/Element for original»
«/Element for http://…/isbn/2020386682»
«Element for http://…/isbn/2020386682»
  «Element for titre»
      Le palais des mirroirs
  «/Element for titre»
«/Element for http://…/isbn/2020386682»
```

Copyright © 2008, W3C

RDF/XML principles (cont.)

- Encode the resources (i.e., the nodes):

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://…/isbn/2020386682">
        «Element for original»
            <rdf:Description rdf:about="http://…/isbn/000651409X"/>
        «/Element for f:original»
    </rdf:Description>
<rdf:RDF>
```

Copyright © 2008, W3C

RDF/XML principles (cont.)

- Encode the properties (i.e., edges) in their own namespaces:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:f="http://www.editeur.fr""><
    <rdf:Description rdf:about="http://…/isbn/2020386682">
       <f:original>
            <rdf:Description rdf:about="http://…/isbn/000651409X"/>
       </f:original>
    </rdf:Description>
<rdf:RDF>
```

Copyright © 2008, W3C

Examples of RDF/XML “simplifications”

- Object references can be put into attributes
- Several properties on the same resource

```
<rdf:Description rdf:about="http://…/isbn/2020386682">
   <f:original rdf:resource="http://…/isbn/000651409X"/>
   <f:titre>
       Le palais des mirroirs
   </f:titre>
</rdf:Description>
```

- There are other “simplification rules”, see the “RDF/XML Serialization” document for details

Copyright © 2008, W3C

“Internal” nodes

- Consider the following statement:
  - “the publisher is a «thing» that has a name and an address”
- Until now, nodes were identified with a URI. But…
- …what is the URI of «thing»?

Copyright © 2008, W3C

One solution: create an extra URI

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <a:publisher rdf:resource="urn:uuid:f60ffb40-307d-…"/>
</rdf:Description>
<rdf:Description rdf:about="urn:uuid:f60ffb40-307d-…">
   <a:p_name>HarpersCollins</a:p_name>
   <a:city>HarpersCollins</a:city>
</rdf:Description>
```

- The resource will be “visible” on the Web
  - care should be taken to define *unique* URI-s
- Serializations may give syntactic help to define local URI-s

Copyright © 2008, W3C

Internal identifier (“blank nodes”)

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <a:publisher rdf:nodeID="A234"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A234">
   <a:p_name>HarpersCollins</a:p_name>
   <a:city>HarpersCollins</a:city>
</rdf:Description>
```

```
<http://…/isbn/2020386682> a:publisher _:A234.
_:A234 a:p_name "HarpersCollins".
```

- Syntax is serialization dependent
- A234 is invisible from outside (it is not a “real” URI!); it is an internal identifier for a resource

Copyright © 2008, W3C

Blank nodes: the system can also do it

- Let the system create a “nodeID” internally (you do not really care about the name…)

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
  <a:publisher>
      <rdf:Description>
          <a:p_name>HarpersCollins</a:p_name>
          …
      </rdf:Description>
  </a:publisher>
</rdf:Description>
```

Copyright © 2008, W3C

Same in Turtle

```
<http://…/isbn/000651409X> a:publisher [
    a:p_name "HarpersCollins";
    …
].
```

Copyright © 2008, W3C

Blank nodes: some more remarks

- Blank nodes require attention when merging
  - blanks nodes with identical nodeID-s in *different* graphs are *different*
  - implementations must be careful…
- Many applications prefer not to use blank nodes and define new URI-s “on-the-fly”
  - eg, when triples are in a database
- From a logic point of view, blank nodes represent an “existential” statement
  - “there is a resource such that…”

Copyright © 2008, W3C

RDF in programming practice

- For example, using Java+Jena (HP's Bristol Lab):
  - a "Model" object is created
  - the RDF file is parsed and results stored in the Model
  - the Model offers methods to retrieve:
    - triples
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
    - etc.
  - the rest is conventional programming…
- Similar tools exist in Python, PHP, etc.

Copyright © 2008, W3C

Jena example

```
// create a model
Model model=new ModelMem();
Resource subject=model.createResource("URI_of_Subject")
// 'in' refers to the input file
model.read(new InputStreamReader(in));
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
   st = iter.next();
   p = st.getProperty();
   o = st.getObject();
   do_something(p,o);
}
```

Copyright © 2008, W3C

Merge in practice

- Environments merge graphs automatically
  - e.g., in Jena, the Model can load several files
  - the load merges the new statements automatically

Copyright © 2008, W3C

RDF schemas

Need for RDF schemas

- First step towards the “extra knowledge”:
  - define the terms we can use
  - what restrictions apply
  - what extra relationships are there?
- Officially: “RDF Vocabulary Description Language”
  - the term “Schema” is retained for historical reasons…

Copyright © 2008, W3C

Classes, resources, …

- Think of well known traditional ontologies or taxonomies:
  - use the term “novel”
  - “every novel is a fiction”
  - “«The Glass Palace» is a novel”
  - etc.
- RDFS defines resources and classes:
  - everything in RDF is a “resource”
  - “classes” are also resources, but…
  - …they are also a collection of possible resources (i.e., “individuals”)
    - “fiction”, “novel”, …

Copyright © 2008, W3C

Classes, resources, … (cont.)

- Relationships are defined among classes/resources:
  - "typing": an individual belongs to a specific class
    - "«The Glass Palace» is a novel"
    - to be more precise: "«http://.../**000651409X**» is a novel"
  - "subclassing": *all* instances of one are also the instances of the other ("every novel is a fiction")
- *RDFS formalizes these notions in RDF*

Copyright © 2008, W3C

Classes, resources in RDF(S)

- RDFS defines the meaning of these terms
  - (these are all special URI-s, we just use the namespace abbreviation)

Copyright © 2008, W3C

Schema example in RDF/XML

- The schema part:

```
<rdf:Description rdf:ID="Novel">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

- The RDF data on a specific novel:

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <rdf:type rdf:resource="http://…/bookSchema.rdf#Novel"/>
</rdf:Description>
```

Copyright © 2008, W3C

An aside: typed nodes in RDF/XML

- A frequent simplification rule: instead of

```
<rdf:Description rdf:about="http://...">
    <rdf:type rdf:resource="http://..../something#ClassName>
    ...
</rdf:Description>
```

use:

```
<yourNameSpace:ClassName rdf:about="http://...">
    ...
</yourNameSpace:ClassName>
```

ie:

```
<a:Novel rdf:about="http://…/isbn/000651409X">
    ...
</a:Novel>
```

Copyright © 2008, W3C

Further remarks on types

- A resource may belong to several classes
  - rdf:type is just a property…
  - “«The Glass Palace» is a novel, but «The Glass Palace» is also an «inventory item»…”
- i.e., it is not like a datatype!
- The type information may be very important for applications
  - e.g., it may be used for a categorization of possible nodes
  - probably the most frequently used RDF property…
- (remember the “Person” in our example?)

Copyright © 2008, W3C

Inferred properties

```
(<http://…/isbn/000651409X> rdf:type #Fiction)
```

- is not in the original RDF data…
- …but can be inferred from the RDFS rules
- RDFS environments return that triple, too

Copyright © 2008, W3C

Inference: let us be formal…

- The RDF Semantics document has a list of (44) *entailment rules*:
  - "if such and such triples are in the graph, add this and this"
  - do that recursively until the graph does not change
- The relevant rule for our example:

```
If:
  uuu rdfs:subClassOf xxx .
  vvv rdf:type uuu .
Then add:
  vvv rdf:type xxx .
```

Copyright © 2008, W3C

Properties

- Property is a special class (`rdf:Property`)
  - properties are also resources identified by URI-s
- Properties' range and domain can be specified
  - i.e., what type of resources can serve as object and subject
  - an important purpose: license inferences
    - I can *infer* that, say, the object is of a specific type
- There is also a possibility for a "sub-property"
  - all resources bound by the "sub" are also bound by the other

Copyright © 2008, W3C

Properties (cont.)

- Properties are also resources (named via URI–s)…
- So properties of properties can be expressed as… RDF properties
  - this twists your mind a bit, but you can get used to it
- For example, `(P rdfs:range C)` means:
  - `P` is a property
  - `C` is a class instance
  - when using `P`, the “object” must be an individual in `C`
- This is an RDF statement with subject `P`, object `C`, and property `rdfs:range`

Copyright © 2008, W3C

Property specification example

Copyright © 2008, W3C

Property specification serialized

- In RDF/XML:

```
<rdf:Property rdf:ID="title">
  <rdfs:domain rdf:resource="#Fiction"/>
  <rdfs:range rdf:resource="http://...#Literal"/>
</rdf:Property>
```

- In Turtle:

```
:title
  rdf:type     rdf:Property;
  rdfs:domain  :Fiction;
  rdfs:range   rdfs:Literal.
```

Copyright © 2008, W3C

What does this mean?

- Again, new relations can be deduced. Indeed, if

```
:title
  rdf:type     rdf:Property;
  rdfs:domain  :Fiction;
  rdfs:range   rdfs:Literal.

<http://…/isbn/000651409X> :title "The Glass Palace" .
```

- then the system can *infer* that:

```
<http://…/isbn/000651409X> rdf:type :Fiction .
```

Copyright © 2008, W3C

Literals

- Literals may have a data type
  - floats, integers, booleans, etc, defined in XML Schemas
    - one can also define complex structures and restrictions via regular expressions, …
  - full XML fragments
- (Natural) language can also be specified

Copyright © 2008, W3C

XML literals in RDF/XML

- XML Literals
  - makes it possible to “include” XML vocabularies into RDF:

```
<rdf:Description rdf:about="#Path">
   <axsvg:algorithmUsed rdf:parseType="Literal">
      <math xmlns="...">
        <apply>
          <laplacian/>
          <ci>f</ci>
        </apply>
      </math>
   </axsvg:algorithmUsed>
</rdf:Description/>
```

Copyright © 2008, W3C

A bit of RDFS can take you far…

- Remember the power of merge?
- We could have used, in our example:
  - **`f:auteur`** is a subproperty of **`a:author`** and vice versa (although we will see other ways to do that…)
- Of course, in some cases, more complex knowledge is necessary (see later…)

Copyright © 2008, W3C

Some predefined classes

Predefined classes and properties

- RDF(S) has some predefined classes and properties
- They are not new “concepts” in the RDF Model, just resources with an agreed semantics
- Examples:
  - collections (a.k.a. lists)
  - containers: sequence, bag, alternatives
  - reification
  - **`rdfs:comment`**, **`rdfs:seeAlso`**, **`rdf:value`**

Copyright © 2008, W3C

Collections (lists)

- We could have the following statement:
  - "The book inventory is a «thing» that consists of **`<http://…/isbn/000651409X>`**, **`<http://…/isbn/000XXXX>`**, …"
- But we also want to express the constituents *in this order*
- Using blank nodes is not enough

Copyright © 2008, W3C

Collections (lists) (cont.)

- Familiar structure for Lisp programmers…

Copyright © 2008, W3C

The same in RDF/XML and Turtle

```
<rdf:Description rdf:about="#Inventory">
    <a:consistsOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://.../isbn/000651409X"/>
        <rdf:Description rdf:about="http://.../isbn/XXXX"/>
        <rdf:Description rdf:about="http://.../isbn/YYYY"/>
    </a:consistsOf>
</rdf:Description>
```

```
:Inventory a:consistsOf
     (<http://.../isbn/000651409X> <http://.../isbn/XXXX> …) .
```

Copyright © 2008, W3C

Sequences

- Use the predefined:
  - RDF Schema class **`Seq`**
  - RDF properties **`rdf:_1`**, **`rdf:_2`**, …
- The *agreed* semantics is of a sequential containment

Copyright © 2008, W3C

Sequences serialized

- In RDF/XML:

```
<rdf:Description rdf:about="#Inventory">
  <a:consistsOf>
      <rdf:Description>
          <rdf:type rdf:resource="http:...rdf-syntax-ns#Seq">
          <rdf:_1 rdf:resource="http://.../isbn/000651409X>
          ...
      </rdf:Description>
  </a:consistsOf>
</rdf:Description/>
```

- In Turtle:

```
:Inventory
  a:consistsOf [
    rdf:type <http:...rdf-syntax-ns#Seq>;
    rdf:_1   <http://.../isbn/000651409X>;
          ...
  ].
```

Copyright © 2008, W3C

Sequences (simplified RDF/XML)

```
<rdf:Description rdf:about="#Inventory">
    <a:consistsOf>
        <rdf:Seq>
            <rdf:li rdf:resource="http://.../isbn/000651409X">
            ...
        </rdf:Seq>
    </a:consistsOf>
</rdf:Description/>
```

Copyright © 2008, W3C

Other containers

- Also defined in RDFS
  - rdf:Bag
    - a general bag, no particular semantics attached
  - rdf:Alt
    - *agreed* semantics: only one of the constituents is “valid”
- Note: these containers are incompletely defined semantically; it is better not to use them…
  - use repeated predicates for bags
  - use lists for sequences

Copyright © 2008, W3C

Get to RDF(S) data

Simple approach

- Write RDF/XML or Turtle “manually”
- In some cases that is necessary, but it really does not scale…

Copyright © 2008, W3C

RDF/XML with XHTML

- XHTML is still based on DTD-s
- RDF within XHTML's header does not validate…
- Currently, people use
  - link/meta in the header (using conventions instead of namespaces in metas)
  - put RDF in a comment (e.g., Creative Commons)

Copyright © 2008, W3C

RDF can also be extracted/generated

- Use intelligent “scrapers” or “wrappers” to extract a structure (hence RDF) from a Web pages or XML files…
  - using conventions in, e.g., class names or meta elements
- … and then generate RDF automatically (e.g., via an XSLT script)

Copyright © 2008, W3C

Formalizing the scraper approach: GRDDL

- GRDDL formalizes scrapers:

```
<html xmlns="http://www.w3.org/1999/">
  <head profile="http://www.w3.org/2003/g/data-view">
    <title>Some Document</title>
    <link rel="transformation" href="http:…/dc-extract.xsl"/>
    <meta name="DC.Subject" content="Some subject"/>
    ...
  </head>
  ...
  <span class="date">2006-01-02</span>
  ...
</html>
```

- yields, via **dc-extract.xsl**:

```
<rdf:Description rdf:about="…">
  <dc:subject>Some subject</dc:subject>
  <dc:date>2006-01-02</dc:date>
</rdf:Description>
```

Copyright © 2008, W3C

The GRDDL process: simple case

Copyright © 2008, W3C

The GRDDL process: merging case

Copyright © 2008, W3C

The GRDDL process: indirect case

Copyright © 2008, W3C

GRDDL

- The transformation itself has to be provided for each set of conventions (meta-s, class id-s, etc…)
  - the practical deployment for generally used formats is through the profile document mechanism
- A “bridge” to “microformats”
- A method to get data in other formats to RDF (e.g., XBRL)
  - the “indirect” approach means that, e.g., only the namespace document has to be modified (important for legacy data)

Copyright © 2008, W3C

Another solution: RDFa

- For example:

```
<div about="http://uri.to.newsitem">
  <span property="dc:date">March 23, 2004</span>
  <span property="dc:title">Rollers hit casino for £1.3m</span>
  By <span property="dc:creator">Steve Bird</span>. See
  <a href="http://www.a.b.c/d.avi" rel="dcmtype:MovingImage">
  also video footage</a>…
</div>
```

- yields:

```
<http://uri.to.newsitem>
  dc:date              "March 23, 2004";
  dc:title             "Rollers hit casino for £1.3m;
  dc:creator           "Steve Bird";
  dcmtype:MovingImage <http://www.a.b.c/d.avi>.
```

Copyright © 2008, W3C

RDFa (cont.)

- RDFa extends (X)HTML a bit by:
  - defining general attributes to add data to any elements
  - an almost complete “serialization” of RDF in XHTML
- It is a bit like the microformats/GRDDL approach but with more rigour and fully generic
  - makes it easy to mix different vocabularies, which is not that easy with microformats
  - does not require a new transformation for each new vocabulary

Copyright © 2008, W3C

Bridge to relational databases

- Data on the Web are mostly stored in databases
- “Bridges” are being defined:
  - a layer between RDF and the relational data
    - RDB tables are “mapped” to RDF graphs, possibly on the fly
    - different mapping approaches are being used
  - a number RDB systems offer this facility already (eg, Oracle, OpenLink, …)
- Work for a survey on mapping techniques at W3C

Copyright © 2008, W3C

Linking Open Data Project

- Goal: “expose” open datasets in RDF
- *Set RDF links among the data items* from different datasets
- Set up SPARQL endpoints
- Billions triples, millions of “links”

As of September 2008

Copyright © 2008, W3C

Example data source: DBpedia

- DBpedia is a community effort to
  - extract structured ("infobox") information from Wikipedia
  - provide a SPARQL endpoint to the dataset
  - interlink the DBpedia dataset with other datasets on the Web

Freie Universität Berlin

UNIVERSITÄT LEIPZIG

OPENLINK SOFTWARE

Copyright © 2008, W3C

Extracting structured data from Wikipedia

Amsterdam

The Keizersgracht at dusk

Location of Amsterdam

Coordinates: 52°22'23"N 4°53'32"E

Country	Netherlands
Province	North Holland
Government	
- Type	Municipality
- Mayor	Job Cohen[1] (PvdA)
- Aldermen	Lodewijk Asscher Carolien Gehrels Tjeerd Herrema Maarten van Poelgeest Marijke Vos
- Secretary	Erik Gerritsen
Area [2][3]	
- City	219 km² (84.6 sq mi)
- Land	166 km² (64.1 sq mi)
- Water	53 km² (20.5 sq mi)
- Urban	1,003 km² (387.3 sq mi)
- Metro	1,815 km² (700.8 sq mi)
Elevation [4]	2 m (7 ft)
Population (1 October 2008)[5][6]	
- City	755,269
- Density	4,459/km² (11,548.8/sq mi)
- Urban	1,364,422
- Metro	2,158,372
- Demonym	Amsterdammer
Time zone	CET (UTC+1)
- Summer (DST)	CEST (UTC+2)
Postcodes	1011 – 1109
Area code(s)	020

Website: www.amsterdam.nl

```
@prefix dbpedia <http://dbpedia.org/resource/>.
@prefix dbterm  <http://dbpedia.org/property/>.

dbpedia:Amsterdam
  dbterm:officialName "Amsterdam" ;
  dbterm:longd "4" ;
  dbterm:longm "53" ;
  dbterm:longs "32" ;
  ...
  dbterm:leaderTitle "Mayor" ;
  dbterm:leaderName dbpedia:Job_Cohen ;
  ...
  dbterm:areaTotalKm "219" ;
  ...
dbpedia:ABN_AMRO
  dbterm:location dbpedia:Amsterdam ;
  ...
```

Copyright © 2008, W3C

Automatic links among open datasets

```
<http://dbpedia.org/resource/Amsterdam>
  owl:sameAs <http://rdf.freebase.com/ns/...> ;
  owl:sameAs <http://sws.geonames.org/2759793> ;
  ...
```

```
<http://sws.geonames.org/2759793>
  owl:sameAs <http://dbpedia.org/resource/Amsterdam>
  wgs84_pos:lat "52.3666667" ;
  wgs84_pos:long "4.8833333" ;
  geo:inCountry <http://www.geonames.org/countries/#NL> ;
 ...
```

Processors can switch automatically from one to the other…

Copyright © 2008, W3C

RDF data access, a.k.a. query (SPARQL)

Copyright © 2008, W3C

RDF data access

- How do I *query* the RDF data?
  - e.g., how do I get to the DBpedia data?

Copyright © 2008, W3C

Querying RDF graphs

- Remember the Jena idiom:

```
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty(); o = st.getObject();
    do_something(p,o);
```

- In practice, more complex queries into the RDF data are necessary
  - something like: “give me the (a,b) pair of resources, for which there is an x such that (x parent a) and (b brother x) holds” (ie, return the uncles)
  - these rules may become quite complex
- The goal of SPARQL (Query Language for RDF)

Copyright © 2008, W3C

Analyze the Jena example

```
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty(); o = st.getObject();
    do_something(p,o);
```

- The **`(subject,?p,?o)`** is a *pattern* for what we are looking for (with **`?p`** and **`?o`** as "unknowns")

Copyright © 2008, W3C

General: graph patterns

- The fundamental idea: use graph patterns
  - the pattern contains unbound symbols
  - by binding the symbols, subgraphs of the RDF graph are selected
  - if there is such a selection, the query returns bound resources

Copyright © 2008, W3C

Our Jena example in SPARQL

```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

- The triples in **WHERE** define the graph pattern, with **?p** and **?o** "unbound" symbols
- The query returns *all* **p**,**o** pairs

Copyright © 2008, W3C

Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

- Returns:
  [[<..49X>,33,£], [<..49X>,50,€], [<..6682>,60,€], [<..6682>,78,$]]

Pattern constraints

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == € }
```

- Returns: [[<..409X>,50,€], [<..6682>,60,€]]

Optional pattern

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

- Returns: [[<..49X>,33,£,<…Palace>], … , [<..6682>,78,$,]]

Copyright © 2008, W3C

Optional pattern

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

- Returns: [[<..49X>,33,£,<…Palace>], … , [<..6682>,78,$,]]

Other SPARQL features

- Limit the number of returned results; remove duplicates, sort them, …
- Specify several data sources (via URI-s) within the query (essentially, a merge!)
- *Construct* a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern

Copyright © 2008, W3C

SPARQL usage in practice

- SPARQL is usually used over the network
  - separate documents define the protocol and the result format
    - SPARQL Protocol for RDF with HTTP and SOAP bindings
    - SPARQL results in XML or JSON formats
- Big datasets usually offer “SPARQL endpoints” using this protocol
  - typical example: SPARQL endpoint to DBpedia

Copyright © 2008, W3C

Remote query/reply example

```
GET /qps?&query=SELECT+:…+WHERE:+… HTTP/1.1
User-Agent: my-sparql-client/0.0
Host: my.example

HTTP/1.1 200 OK
Server: my-sparql-server/0.0
Content-Type: application/sparql-results+xml

<?xml version="1.0" encoding="UTF-8"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#>
  <head>
    <variable name="a"/>
    ...
  </head>
  <results>
    <result ordered="false" distinct="false">
      <binding name="a"><uri>http:…</uri></binding>
      ...
    </result>
    <result> ... </result>
  </results>
</sparql>
```

Copyright © 2008, W3C

The power of CONSTRUCT: “chaining” queries

```
CONSTRUCT {
    <http://dbpedia.org/resource/Amitav_Ghosh> ?p1 ?o1.
    ?s2 ?p2 <http://dbpedia.org/resource/Amitav_Ghosh>.
}
WHERE {
    <http://dbpedia.org/resource/Amitav_Ghosh> ?p1 ?o1.
    ?s2 ?p2 <http://dbpedia.org/resource/Amitav_Ghosh>.
}
```

- SPARQL endpoint
- returns RDF/XML

```
SELECT *
FROM <http://dbpedia.org/sparql/?query=CONSTRUCT+%7B++…>
WHERE {
  ?author_of dbpedia:author res:Amitav_Ghosh.
  res:Amitav_Ghosh  dbpedia:reference ?homepage;
                    rdf:type          ?type;
                    foaf:name         ?foaf_name.
  FILTER regex(str(?type),"foaf")
}
```

- Data reused in another query…

Copyright © 2008, W3C

A word of warning on SPARQL…

- Some features are missing
  - control and/or description on the entailment regimes of the triple store (RDFS? OWL-DL? OWL-Lite? …)
  - modify the triple store
  - querying collections or containers may be complicated
  - no functions for sum, average, min, max, …
  - ways of aggregating queries
  - …
- Delayed for a next version…

Copyright © 2008, W3C

SPARQL as a unifying point

Copyright © 2008, W3C

SPARQL-ing DBpedia

Virtuoso SPARQL Query Form - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://dbpedia.org/sparql

World Clock Netvibes Validate! Twines Favikis RDFa it! Semantic Web W3C Mailing lists Bookmarklets Python Diigolet Mobical VeriSign PIP Google Calendar

OpenLink Virtuoso SPARQL Query

This query page is designed to help you test OpenLink Virtuoso SPARQL protocol endpoint.
Consult the Virtuoso Wiki page describing the service or the Online Virtuoso Documentation section RDF Database and SPARQL.

There is also a rich Web based user interface with sample queries. You can access it at: /isparql.

Query

Default Graph URI

http://dbpedia.org

Use only local data (including data retrieved before), but do not retrieve more

Query text

edit

```
select ?city
where {
  ?city <http://dbpedia.org/property/leaderName>
<http://dbpedia.org/resource/Job_Cohen>
}
```

Done

Data Sources Pending: 1

How to “assign” RDF data to resources?

- This is important when the RDF data is used as “metadata”
- Some examples:
  - copyright information for your photographs
  - is a Web page usable on a mobile phone and how?
  - bibliographical data for a publication
  - annotation of the data resulting from a scientific experiment
  - etc
- The issue: if I have the URI of the resource (photograph, publication, etc), how do I find the relevant RDF data?

Copyright © 2008, W3C

The data might be embedded

- Some data formats allow the direct inclusion of (RDF) metadata:
  - SVG (Scalable Vector Graphics)
  - XHTML+RDFa
  - microformats+GRDDL
  - JPG files using the comment area and, eg, Adobe's XMP technology
- That can include all the information, or link to further data

Copyright © 2008, W3C

Simple linkage

- Some formats have special link statements. Eg, in (X)HTML:

```
<html>
  <head>
    <link rel="meta" href="meta.rdf"/>
 ...
```

- Similar facilities might exist in other formats (eg, SMIL)

Copyright © 2008, W3C

POWDER

- POWDER provides for a more elaborate scenarios
- With POWDER:
  1. define a *set* of resources by constraints on the URIs; eg
     - URIs must begin with **`http://www.example.com/bla/`**
     - the port number in the URI-s should be **`XYZW`**
  2. define description resources binding the set of resources to additional information
  3. get such description resources, eg, via a link statements, via HTTP, via SPARQL, …
- Use cases: licensing information, mobileOK (and other) trustmarks, finding accessible Web sites, content labeling (eg, child protection), …

Copyright © 2008, W3C

A POWDER scenario: copyright for photos

POWDER (cont)

- There are a number of additional features:
  - built in authentication: description resources *must* be attributed and this is open for authentication
  - assignments may carry validity dates
  - packaging several resource descriptions in one, possibly control their processing order
  - depositories for a specific site can be built, querying them via SPARQL
    - e.g., “give me the cc:license for this and this URI”
- It is a technology under development at W3C (finalization expected in autumn 2008)

Copyright © 2008, W3C

Ontologies (OWL)

Ontologies

- RDFS is useful, but does not solve all possible requirements
- Complex applications may want more possibilities:
  - characterization of properties
  - identification of objects with different URI-s
  - disjointness or equivalence of classes
  - construct classes, not only name them
  - more complex classification schemes
  - can a program reason about some terms? E.g.:
    - “if «Person» resources «A» and «B» have the same «**`foaf:email`**» property, then «A» and «B» are identical”
  - etc.

Copyright © 2008, W3C

Ontologies (cont.)

- The term *ontologies* is used in this respect:

"defines the concepts and relationships used to describe and represent an area of knowledge"

- Ie, there is a need for Web Ontology Language(s)
  - RDFS can be considered as a simple ontology language
- Languages should be a compromise between
  - rich semantics for meaningful applications
  - feasibility, implementability

Copyright © 2008, W3C

Web Ontology Language = OWL

- OWL is an extra layer, a bit like RDF Schemas
  - own namespace, own terms
  - it relies on RDF Schemas
- It is a separate recommendation
- There is an active W3C Working Group working on *extensions* of the current standards
  - the new version will be called “OWL 2”
  - in what follows, some features will be referred to as “may come in future”, i.e., under consideration by that group

Copyright © 2008, W3C

OWL is complex…

- OWL is a large set of additional terms
- We will not cover the whole thing here…

Copyright © 2008, W3C

First some simple features

Term equivalence

- For classes:
  - **`owl:equivalentClass`**: two classes have the same individuals
  - **`owl:disjointWith`**: no individuals in common
- For properties:
  - **`owl:equivalentProperty`**
    - remember the **`a:author`** vs. **`f:auteur`**?
- For individuals:
  - **`owl:sameAs`**: two URIs refer to the same concept (“individual”)
  - **`owl:differentFrom`**: negation of **`owl:sameAs`**

Copyright © 2008, W3C

Typical usage of `owl:sameAs`

- Linking our example of Kolkata from one data set (DBpedia) to the other (Geonames):

```
<http://dbpedia.org/resource/Kolkata>
  owl.sameAs <http://sws.geonames.org/1275004/>;
```

- This is the main mechanism of “Linking” in the Linking Open Data project

Copyright © 2008, W3C

Other example: connecting to French

Copyright © 2008, W3C

Property characterization

- In OWL, one can characterize the behaviour of properties (symmetric, transitive, functional, inverse functional…)
- OWL also separates *data* and *object* properties
  - “datatype property” means that its range are typed literals

Copyright © 2008, W3C

Characterization example

- "`foaf:email`" is inverse functional (i.e., two different subjects cannot have identical objects)

Copyright © 2008, W3C

What this means is…

- If the following holds in our triples:

```
:email rdf:type owl:InverseFunctionalProperty.
<A> :email "mailto:a@b.c".
<B> :email "mailto:a@b.c".
```

- then the following holds, too:

```
<A> owl:sameAs <B>.
```

- I.e., *new relationships* were discovered again (beyond what RDFS could do)

Copyright © 2008, W3C

Other property characterizations

- Functional property (“**`owl:FunctionalProperty`**”)
- Transitive property (“**`owl:TransitiveProperty`**”)
- Symmetric property (“**`owl:SymmetricProperty`**”)
- Inverse of another property (“**`owl:inverseOf`**”)
- May come in future:
  - reflexive and irreflexive object properties
  - specify that properties are “disjoint”

Copyright © 2008, W3C

May come in future: property chains

- Properties, when applied one after the other, may be subsumed by yet another one:
  - “if a person «P» was born in city «A» and «A» is in country «B» then «P» was born in country «B»”
  - more formally (exact syntax is still to be defined):

```
[
    rdfs:subPropertyOf ex:born_in_country;
    owl2:propertyChain (ex:born_in_city ex:city_in_country)
]
```

- More than two constituents can be used
- There are some restrictions to avoid “circular” specifications

Copyright © 2008, W3C

Classes in OWL

- In RDFS, you can subclass existing classes… that's all
- In OWL, you can *construct* classes from existing ones:
  - enumerate its content
  - through intersection, union, complement
  - etc
- OWL makes a stronger distinction between *classes* and *individuals*
  - referring to its own **`Class`** and to "**`Thing`**", respectively
    - of course, **`owl:Class`** is a subclass of **`rdfs:Class`**, i.e., it is a refinement

Copyright © 2008, W3C

OWL classes can be “enumerated”

- The OWL solution, where possible content is explicitly listed:

Same serialized

```
<owl:Class  rdf:ID="Currency">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:ID="£"/>
    <owl:Thing rdf:ID="€"/>
    <owl:Thing rdf:ID="$"/>
    …
  </owl:oneOf>
</owl:Class>
```

```
:£ rdf:type owl:Thing.
:€ rdf:type owl:Thing.
:$ rdf:type owl:Thing.
:Currency
    rdf:type owl:Class;
    owl:oneOf (:€ :£ :$).
```

- I.e., the class consists of *exactly* of those individuals

Copyright © 2008, W3C

Union of classes

- Essentially, like a set-theoretical union:

Copyright © 2008, W3C

Same serialized

```
<owl:Class rdf:ID="Literature">
   <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Novel"/>
       <owl:Class rdf:about="#Short_Story"/>
       <owl:Class rdf:about="#Poetry"/>
       …
   </owl:unionOf>
</owl:Class>
```

```
:Novel            rdf:type owl:Class.
:Short_Story      rdf:type owl:Class.
:Poetry           rdf:type owl:Class.
:Literature rdf:type owlClass;
   owl:unionOf (:Novel :Short_Story :Poetry).
```

- Other possibilities: **`complementOf`**, **`intersectionOf`**
- May come in future: **`disjointUnion`**

Copyright © 2008, W3C

What we have so far…

- The OWL features listed so far are already fairly powerful
- E.g., various databases can be linked via **`owl:sameAs`**, functional or inverse functional properties, etc.
- It is still possible to find inferred relationship using a traditional rule engine
  - (more or less… there are some restrictions on details)

Copyright © 2008, W3C

However… that may not be enough

- Very large vocabularies might require even more complex features
  - typical example: definition of all concepts in a health care environment
- One major issue is the way classes (i.e., “concepts”) are defined
- OWL includes those extra features but… the inference engines become (much) more complex ☹

Copyright © 2008, W3C

Property value restrictions

- Classes are created by *restricting* the property values on a superclass
- For example: how would I characterize a “listed price”?
  - it is a price (which may be a general term), but one that is given in one of the “allowed” currencies (say, €, £, or $)
  - more formally:
    - the value of “**`p:currency`**”, when applied to a resource on listed price, *must* take one of those values…
    - …thereby defining the class of “listed price”

Copyright © 2008, W3C

Property restriction example in OWL

- "the value of «`p:currency`» *when applied to a resource on listed price* must take one of those values…":

Restrictions formally

- Defines a node of type `owl:Restriction` with a
  - reference to the property that is constrained
  - definition of the constraint itself
- One can, e.g., subclass from this node when defining a particular class

Copyright © 2008, W3C

Restrictions expressed in OWL/RDF

```
:Listed_Price rdf:type owl:Class;
    rdfs:subClassOf [
      rdf:type              owl:Restriction;
      owl:onProperty        <http://…#currency>;
      owl:allValuesFrom     :Currency.
   ].
```

- "**`allValuesFrom`**" could be replaced by "**`someValuesFrom`**" to express another type of restriction
  - e.g., I could have said: there should be a price given in *at least one* of those currencies
- or "**`hasValue`**", when restricted to *one specific value*

Copyright © 2008, W3C

Similar concept: cardinality restriction

- In a property restriction, the issue was to restrict the possible *values* of a property
- In a cardinality restriction, the *number* of relations with that property is restricted
- Eg: “a book being on offer” could be characterized as having at *least one price property* (i.e., the price of the book has been established)

Copyright © 2008, W3C

Cardinality restriction

```
:Book_on_sale rdf:type owl:Class;
    rdfs:subClassOf [
      rdf:type              owl:Restriction;
      owl:onProperty        <http://…#price>;
      owl:minCardinality    "1"^^xsd:integer.
   ].
```

- could also be "**owl:cardinality**" or "**owl:maxCardinality**"

Copyright © 2008, W3C

May come in future: QCR

- "Qualified cardinality restriction": combining cardinality and value restriction
  - "there should be *exactly* two listed price tags with currency value"

```
:Listed_Price rdf:type owl:Class;
    rdfs:subClassOf [
      rdf:type                    owl:Restriction;
      owl:onProperty              <http://…#currency>;
      owl:onClass                 :Currency;
      owl:qualifiedCardinality    "2"^^xsd:integer.
   ].
```

Copyright © 2008, W3C

But: OWL is hard!

- The combination of class constructions with various restrictions is extremely powerful
- What we have so far is following the same logic as before
  - extend the basic RDF and RDFS possibilities with new features
  - expect to infer new relationships based on those
- However… a full inference procedure is hard ☹
  - not implementable with simple rule engines, for example
  - in some cases, it may even be impossible

Copyright © 2008, W3C

OWL profiles

- The term OWL “profiles” comes to the fore:
  - restricting *which* terms can be used and *under what circumstances (restrictions)*
  - if one abides to those restrictions, then simpler inference engines can be used

Copyright © 2008, W3C

OWL profiles (cont.)

- In the *current* OWL standard, three such “profiles” are defined:
  - OWL Full: no restrictions whatsoever
  - OWL DL (and its “sub profile” OWL Lite): major restrictions to ensure implementability
- The OWL 2 work will add new profiles
  - profiles that are simple enough to be implementable with simple rule engines (like the first few examples we had)
  - profiles that are optimized to a small number of class and property definition but a large amount of data
  - etc.

Copyright © 2008, W3C

OWL Full

- No constraints on the various constructs
  - **`owl:Class`** is equivalent to **`rdfs:Class`**
  - **`owl:Thing`** is equivalent to **`rdfs:Resource`**
  - this means that:
    - Class can also be an individual, a URI can denote a property as well as a Class
      - e.g., it is possible to talk about class of classes, etc.
    - one can make statements on RDFS constructs (e.g., declare **`rdf:type`** to be functional…)
    - etc.
- But: *an OWL Full ontology may be undecidable!*

Copyright © 2008, W3C

Example for a possible OWL Full problem

```
:A rdf:type owl:Class;
   owl:equivalentClass [
      rdf:type           owl:Restriction;
      owl:onProperty     rdf:type;
      owl:allValuesFrom :B.
   ].
:B rdf:type owl:Class;
   owl:complementOf :A.
```

- Is the following true?

```
:c rdf:type owl:Thing; rdf:type :A.
```

- Well… if **c** is of type **A** then it must be in **B**, but then it is in the complement of **A**, ie, it is not of type **A**…

Copyright © 2008, W3C

OWL Description Logic (DL)

- A number of restrictions are defined
  - Classes, individuals, properties strictly separated: a class *cannot* be an individual of another class
  - strict separation of the user's and the reserved (RDFS, OWL) terms
    - no statements on RDFS and OWL resources, for example
  - the values of user's object properties must be individuals
    - i.e., they are used to create relationships *between individuals*
  - no characterization of *datatype* properties
  - …
- But: *well known inference algorithms exist!*

Copyright © 2008, W3C

“OWL 2 DL”: metamodeling

- OWL 2 may remove one of the OWL DL restrictions:
  - same symbol may be used both for a class and an instance
  - but not all “natural” inferences can be drawn in OWL 2 DL. E.g.,

```
q rdf:type A.          # A is a class, q is an individual
A owl:sameAs B.        # A and B are equals as individuals
```

  - does *not* yield

```
q rdf:type B.
```

Copyright © 2008, W3C

Note on OWL profiles

- OWL profiles are defined to reflect compromises:
  - expressibility vs. implementability
- Some application just need to express and interchange terms (with possible scruffiness): OWL Full is fine
  - they may build application-specific reasoning instead of using a general one
- Some applications need rigour, but only a simple set of statements: a rule engine based profile might be o.k.
- Some applications need rigour and complex term classification; then OWL DL/Lite might be the good choice

Copyright © 2008, W3C

New profiles in OWL 2 (DL)

- OWL 2 retains the duality of OWL 2 DL and OWL 2 Full
- Classification and instance queries in polynomial time: *OWL-EL*
- Smaller vocabularies over a possibly very large set of RDF data: *OWL-QL* and *OWL-RL*
  - OWL-QL implementable via query rewriting on RDB systems
  - OWL-RL can be implemented with rule engines, too

Copyright © 2008, W3C

“Description Logic”

- An area in knowledge representation
  - a special type of “structured” First Order Logic (logic with safety guards…)
  - formalism based on “concepts” (i.e., classes), “roles” (i.e., properties), and “individuals”
  - based on model theoretic semantics (like RDF, RDFS, and OWL!)
- There are several variants of Description Logic
  - i.e., OWL DL and Lite are embodiments of *distinct* Description Logics
    - for connoisseurs: OWL DL ≈ $\mathcal{SHOIN}$(D), OWL Lite ≈ $\mathcal{SHIF}$(D)
  - some major differences: *usage of URI-s*, reference to XML Schema datatypes, built-in annotation, …

Copyright © 2008, W3C

“Description Logic” (cont.)

- Traditional DL has its own terminology:
  - named objects or concepts ⇔ definition of classes, relationships among classes
  - roles ⇔ properties
  - (terminological) axioms ⇔ subclass and subproperty relationships
  - facts or assertions ⇔ statements on individuals (`owl:Thing`-s)
- There is also a compact mathematical notation for axioms, assertions, etc:
  - $\mathtt{Literature} \equiv \mathtt{Novel} \sqcup \mathtt{Short_Story} \sqcup \mathtt{Poetry}$
  - $\mathtt{Listed_Price} \sqsubseteq \forall \mathtt{currency.Currencies}$
- You may see these in papers, books…

Copyright © 2008, W3C

OWL-DL “Abstract Syntax”

- There is also a non-XML based notation for OWL DL (and Lite)
  - it makes writing ontologies with DL restrictions easier…
  - OWL 2 may come with a renewed “functional syntax” and a more readable “Manchester syntax”

```
Class(Novel)
Class(Short_Story)
Class(Poetry)
Class(Literature) (
  unionOf(Novel Poetry Short_Story …)
)
```

Copyright © 2008, W3C

Ontology development

- The hard work is to *create* the ontologies
  - requires a good knowledge of the area to be described
  - some communities have good expertise already (e.g., librarians)
  - *OWL is just a tool to formalize ontologies*
- Large scale ontologies are often developed in a community process
- Ontologies should be *shared* and *reused*
  - can be via the simple namespace mechanisms…
  - …or via explicit inclusions
- Applications can also be developed with very small ontologies, though

Copyright © 2008, W3C

Ontologies examples

- International Country List
  - example for an OWL Lite ontology
- Large ontologies are being developed
  - eClassOwl: eBusiness ontology for products and services, 75,000 classes and 5,500 properties
  - National Cancer Institute's ontology: about 58,000 classes
  - Open Biomedical Ontologies Foundry: a collection of ontologies, including the Gene Ontology to describe gene and gene product attributes in any organism or protein sequence and annotation terminology and data (UniProt)
  - BioPAX: for biological pathway data

Copyright © 2008, W3C

Simple Knowledge Organization System

Simple Knowledge Organization System

- Goal: represent and share classifications, glossaries, thesauri, etc, as developed in the “Print World”.
  - for example:
    - Dewey Decimal Classification, Art and Architecture Thesaurus, ACM classification of keywords and terms…
    - DMOZ categories (a.k.a. Open Directory Project)
  - allow for a quick port of this traditional data, combine it with other data
- This is where *SKOS* comes in: define classes, properties, to add those structures to a RDF universe

Copyright © 2008, W3C

Example: entries in a glossary

Assertion

(i) Any expression which is claimed to be true.
(ii) The act of claiming something to be true.

Class

A general concept, category or classification. Something used primarily to classify or categorize other things.

Resource

(i) An entity; anything in the universe.
(ii) As a class name: the class of everything; the most inclusive category possible.

(from the RDF Semantics Glossary)

Copyright © 2008, W3C

Example: entries in a glossary in SKOS

Copyright © 2008, W3C

The structure can be translated into RDF

Copyright © 2008, W3C

Same in turtle

```
<http://lcsh.info/sh85048050#concept>
    a    skos:Concept;
    dcterms:created
       "1986-02-11";
    skos:altLabel
       "Novels",
       "Stories"
       ...;
    skos:broader
       <http://lcsh.info/sh85077507>,...;
    skos:narrower
       <http://lcsh.info/sh2002009675>,
       <http://lcsh.info/sh2002011186>,
       <http://lcsh.info/sh2003003087>,
           ...;
    skos:prefLabel
       "Fiction";
    skos:related
       <http://lcsh.info/sh85092863>
    ...
    ...
.
```

Copyright © 2008, W3C

SKOS Reference overview

- Classes and Properties:
  - Basic description (**`Concept`**, **`ConceptScheme`**,…)
  - Labeling (**`prefLabel`**, **`altLabel`**,…)
  - Documentation (**`definition`**, **`historyNote`**,…)
  - Semantic relations (**`broader`**, **`narrower`**, **`related`**,…)
  - Collections (**`Collection`**, **`OrderedCollection`**,…)
  - Concept mappings (**`broadMatch`**, **`narrowMatch`**,…)

Copyright © 2008, W3C

SKOS and OWL

- SKOS is geared towards some specific (though large) use cases, like
  - taxonomies, glossaries, …
  - annotations of complex structures (including ontologies)
- SKOS is a based on a very simple usage of OWL
  - roughly on the rule based level
- But SKOS also adds some of its own inference rules (currently not in OWL)

Copyright © 2008, W3C

Rules

Rules

- There is a long history of rule languages and rule-based systems
  - eg: logic programming (Prolog), production rules
- Lots of small and large rule systems (from mail filters to expert systems)
- Hundreds of niche markets

Copyright © 2008, W3C

Why rules on the Semantic Web?

- There are conditions that ontologies (ie, OWL) cannot express
  - a well known example is Horn rules: $(P1 \land P2 \land \ldots) \rightarrow C$
    - (the property chains of OWL 2 covers *some* cases)
- But there are conditions that are complicated in rules and ontologies are better
- A different way of thinking — people may feel more familiar in one or the other

Copyright © 2008, W3C

Things you may want to express

- An example from our bookshop integration:
  - “if two Persons have the same name and the same email, or the same name and the same home page, then they are identical”
  - something like (in an ad-hoc syntax):

```
If { ?x rdf:type foaf:Person.
     ?y rdf:type foaf:Person.
     ?x foaf:name ?n.
     ?x foaf:homepage ?h.
     ?y foaf:name ?n.
     ?y foaf:homepage ?h. }
then { ?x owl:sameAs ?y }

If { ?x rdf:type foaf:Person.
     ?y rdf:type foaf:Person.
     ?x foaf:name ?n.
     ?x foaf:mailbox ?h.
     ?y foaf:name ?n.
     ?y foaf:mailbox ?m. }
then { ?x owl:sameAs ?y }
```

Copyright © 2008, W3C

A new requirement: *exchange* of rules

- Applications may want to exchange their rules:
  - negotiate eBusiness contracts across platforms: supply vendor-neutral representation of your business rules so that others may find you
  - describe privacy requirements and policies, and let clients "merge" those (e.g., when paying with a credit card)
- Hence the name of the working group: *Rule Interchange Format*
  - a language that
    - express the rules a bit like a rule language with, eg, RDF
    - can be used to exchange rules among engines

Copyright © 2008, W3C

In an ideal World

Copyright © 2008, W3C

In the real World…

- Rule based systems can be very different
  - different rule semantics (based on various type of model theories, on proof systems, etc)
  - production rule systems, with procedural references, state transitions, etc
- Such universal exchange format is not feasible
- The idea is to define “cores” for a family of languages with “variants”

Copyright © 2008, W3C

RIF “core”: only partial interchange

Copyright © 2008, W3C

RIF "dialects"

- Possible dialects: F-logic, production rules, fuzzy logic systems, …

Copyright © 2008, W3C

Role of dialects

Copyright © 2008, W3C

Role of dialect

Role of dialects

Copyright © 2008, W3C

Role of dialects

Copyright © 2008, W3C

However…

- Even this model does not completely work
- The gap between production rules and “traditional” logic systems seems to be large
- A hierarchy of cores is necessary:
  - a *Basic Logic Dialect* and *Production Rule Dialect* as “cores” for families of languages
  - a common *RIF Core* binding these two

Copyright © 2008, W3C

Schematically…

- The “Core”: shared subset of major languages
  - technically: positive Horn without function terms, with some simple datatypes
- The “BLD (Basic Logic Dialect)” is of the form:
  - “**`if condition true then this is true`**”
  - conditions may include functions, hierarchies
- The “PLD (Production Logic Dialect)” is of the form:
  - “**`if condition is true then do something`**”

Copyright © 2008, W3C

Hierarchy of cores

Copyright © 2008, W3C

Current status

- There is a draft for the BLD
  - it has an XML syntax and an extension mechanism for dialects
  - the language can be used
    - with or without RDF data and/or OWL
    - as a rule *language* or a rule *interchange format*
- The plan is to have BLD as a recommendation in 2009
- The work on the PLD Core has begun
- The Core will come as an abstraction from BLD and PLD

Copyright © 2008, W3C

What have we achieved?

Copyright © 2008, W3C

Remember the integration example?

Applications

Query, Manipulate, etc.

Data represented in abstract format

Map, Expose, etc.

Data in various formats

Copyright © 2008, W3C

Same with what we learned

Applications

SPARQL,
OWL inferences,
etc.

Data represented in RDF, possibly with extra knowledge (RDFS, OWL, SKOS, Rules, …)

SQL <=> RDF,
GRDDL, RDFa
etc.

Data in various formats

Copyright © 2008, W3C

What is coming?

Revision of the RDF model?

- Some restrictions in RDF may be unnecessary
- Issue of “named graph”: possibility to give a URI to a set of triples and make statements on those
- Syntax issues in RDF/XML (eg, QNames in properties)
- Alternative XML serializations?
- Add a time tag to statements?
- Internationalization issues with literals (how do I set “bidi” writing?)
- …

Copyright © 2008, W3C

Other items…

- Security, trust, provenance
  - combining cryptographic techniques with the RDF model, sign a portion of the graph, etc
  - trust models
- Access control on statements or groups of statements
- Quality constraints on graphs
  - “may I be sure that certain patterns are present in a graph?”
- Ontology merging, alignment, term equivalences, versioning, development, …
- etc

Copyright © 2008, W3C

Other items: uncertainty

- Fuzzy logic
  - look at alternatives of Description Logic based on fuzzy logic
  - alternatively, extend RDF(S) with fuzzy notions
- Probabilistic statements
  - have an OWL class membership with a specific probability
  - combine reasoners with Bayesian networks
- A W3C Incubator Group ("Uncertainty Reasoning on the World Wide Web") has issued a report on the current status, possibilities, directions, etc
- Possible RIF dialect for fuzzy logic, for example?

Copyright © 2008, W3C

Other items: naming

- The SW infrastructure relies on unique naming of “things” via URI-s
- Lots of discussions are happening that touch upon general Web architecture, too
  - http URI-s or other URN-s?
  - URI-s for “informational resources” and “non informational resources”
  - how to ensure that URI-s used on the SW are dereferencable
  - etc

Copyright © 2008, W3C

Other items: naming (cont)

- A different aspect of naming: *what* is the URI for a specific entity (regardless of the technical details)
  - what is the unique URI for, eg, Bach's Well-Tempered Clavier?
    - obviously important for, eg, music ontologies an data
  - who has the authority or the means to define and maintain such URI-s?
  - should we define characterizing properties for these and use **`owl:sameAs`** instead of a URI?
- The traditional library community may be of a big help in this area

Copyright © 2008, W3C

Another view of the R&D trends

Courtesy of Miles Davis, © Project10X

Copyright © 2008, W3C

Available documents, tools

Available specifications: Primers, Guides

- The “RDF Primer” and the “OWL Guide” give a formal introduction to RDF(S) and OWL
- SKOS has its separate “SKOS Primer”
- GRDDL Primer and RDFa Primer have been published
- The W3C Semantic Web Activity Homepage has links to all the specifications

Copyright © 2008, W3C

“Core” vocabularies

- There are also a number “core vocabularies” (not necessarily OWL based)
  - Dublin Core: about information resources, digital libraries, with extensions for rights, permissions, digital right management
  - FOAF: about people and their organizations
  - DOAP: on the descriptions of software projects
  - SIOC: Semantically-Interlinked Online Communities
  - vCard in RDF
  - …
- One should never forget: ontologies/vocabularies must be shared and reused!

Copyright © 2008, W3C

Some books

- P. Mika: Social Networks and the Semantic Web, 2007
- L. Yu: Semantic Web and Semantic Web Services, 2007
- G. Antoniu and F. van Harmelen: Semantic Web Primer, 2nd edition in 2008
- D. Allemang and J. Hendler: Semantic Web for the Working Ontologist, 2008
- …

See the separate Wiki page collecting book references

Copyright © 2008, W3C

Further information

- Dave Beckett's Resources at Bristol University
  - *huge* list of documents, publications, tools, …
- Planet RDF aggregates a number of SW blogs
- Semantic Web Interest Group
  - a forum developers with archived (and public) mailing list, and a constant IRC presence on freenode.net#swig
  - anybody can sign up on the list

Copyright © 2008, W3C

Great community…

computas

Some good lessons

- New standards (e.g. SPARQL), proposals for standardization (e.g. SPARUL), new tools (e.g. Jena), open source (e.g. Tomcat, Apache), lack of good documentation all say **high risk!!!!**
- **However, the support and maintenance from the W3C community and open source developers (e.g. Jena team) has been impressive, the support through IRC channels, mailing lists etc has been invaluable for the project.**

Slide 19

Reproduction prohibited without authorization by Computas AS ©

From a presentation given by David Norheim, Computas AS, at the ESTC2008 Conference, Vienna, Austria

SWBP Working Group documents

- Documents for ontology engineering
  - "Best Practice Recipes for Publishing RDF Vocabularies"
  - "Defining N-ary relations"
  - "Representing Classes as Property Values";
  - "XML Schema Datatypes in RDF and OWL"
  - etc
- See the Group's homepage for further links

Copyright © 2008, W3C

Further information (cont)

- Description Logic links:
  - online course by Enrico Franconi,
  - teaching material and links by Ian Horrocks
- "Ontology Development 101"
- OWL Reasoning Examples
- Lots of papers at WWW2003-WWW2008; see also the ISWC200X conference proceedings (online since ISWC2006) as well as their European and Asian local variants

Copyright © 2008, W3C

Lots of Tools (*not* an exhaustive list!)

- Categories:
  - Triple Stores
  - Inference engines
  - Converters
  - Search engines
  - Middleware
  - CMS
  - Semantic Web browsers
  - Development environments
  - Semantic Wikis
  - …
- Some names:
  - Jena, AllegroGraph, Mulgara, Sesame, flickurl, …
  - TopBraid Suite, Virtuoso environment, Falcon, Drupal 7, Redland, Pellet, …
  - Disco, Oracle 11g, RacerPro, IODT, Ontobroker, OWLIM, Tallis Platform, …
  - RDF Gateway, RDFLib, Open Anzo, DartGrid, Zitgist, Ontotext, Protégé, …
  - Thetus publisher, SemanticWorks, SWI-Prolog, RDFStore…
  - …

Copyright © 2008, W3C

Tools

- Worth noting: major companies offer (or will offer) Semantic Web tools or systems using Semantic Web: Adobe, Oracle, IBM, HP, Software AG, webMethods, Northrop Gruman, Altova, Dow Jones, BBN, …
- See also the W3C Wiki page on tools

Copyright © 2008, W3C

Deployment, application examples

- See the separate slide set…

Copyright © 2008, W3C

Conclusions

- The Semantic Web is there to integrate data on the Web
- The goal is the creation of a *Web of Data*

Copyright © 2008, W3C

Thank you for your attention!

- These slides are publicly available on:

http://www.w3.org/People/Ivan/CorePresentations/SWTutorial/

Copyright © 2008, W3C

Appendix

- There is a separate slide set on the formal semantics of RDF(S) and OWL…

Copyright © 2008, W3C