Tutorial on Semantic Web

Ivan Herman, W3C
Last update: 2012-04-09
This is an evolving slide set. This means:
- it changes frequently
- there may be bugs, inconsistencies
- it may try to reflect the latest view of technology evolution but that is often a moving target

“Frozen” versions are instantiated for a specific presentation, and those become stable
Introduction
Eric Clapton

Born 30 March 1945.

Biography

Eric Patrick Clapton, CBE (born 30 March 1945) is an English blues-rock guitarist, singer, songwriter and composer. Clapton has been inducted into the Rock and Roll Hall of Fame as a solo performer, as a member of rock bands; the Yardbirds and Cream. Clapton is the only person ever to be inducted three times. Often viewed by critics and fans alike as one of the most important and influential guitarists of all time. Clapton was ranked fourth in Rolling Stone magazine's list of the "100 Greatest Guitarists of All Time" and #53 on their list of the Immortals: 100 Greatest Artists of All Time.
Clapton's chart success was not limited to the blues, with chart-toppers in Delta Blues (Me And Mr. Johnson), pop ("Change the World") and reggae (Bob Marley's "I Shot the Sheriff"). He is often credited for bringing reggae and Bob Marley to the mainstream.) Two of his most successful recordings were the hit love song "Layla", which he played with the band Derek and the Dominos, and Robert Johnson's "Crossroads", which has been his staple song since his days with Cream.

Read more at Wikipedia...

**Wikipedia** This entry is from Wikipedia, the user-contributed encyclopedia. It may not have been reviewed by professional editors and is licensed under the GNU Free Documentation License. If you find the biography content factually incorrect, defamatory or highly offensive you can edit this article at Wikipedia. Find out more about our use of this data.

**Links & Information**

**Links**
- Official homepage at ericclapton.com
- Fanpage at whereseric.com
- Wikipedia article on Eric Clapton
- MySpace at myspace.com/ericclapton
- Last.fm page on Eric Clapton
- MusicBrainz entry on Eric Clapton


Links & information come from MusicBrainz. You can add or edit information about Eric Clapton at musicbrainz.org. Find out more about our use of this data. The BBC is not responsible for the content of external sites.
How to build such a site 1.

- Site editors roam the Web for new facts
  - may discover further links while roaming
- They update the site manually
- And the site gets soon out-of-date 😞
How to build such a site 2.

- Editors roam the Web for new data published on Web sites
- “Scrape” the sites with a program to extract the information
  - Ie, write some code to incorporate the new data
- Easily get out of date again… 😞
How to build such a site 3.

- Editors roam the Web for new data via API-s
- Understand those...
  - input, output arguments, datatypes used, etc
- Write some code to incorporate the new data
- Easily get out of date again... 😞
The choice of the BBC

- Use external, public datasets
  - Wikipedia, MusicBrainz, …
- They are available as data
  - not API-s or hidden on a Web site
  - data can be extracted using, e.g., HTTP requests or standard queries
In short...

- Use the Web of Data as a Content Management System
- Use the community at large as content editors
And this is no secret...
Data on the Web

- There are more and more data on the Web
  - government data, health related data, general knowledge, company information, flight information, restaurants,…
- More and more applications rely on the availability of that data
But: we do not want that!
Imagine...

- A “Web” where
  - documents are available for download on the Internet
  - but there would be no hyperlinks among them
CoCoDat: Collation of Cortical [microcircuitry] Data

CoCoDat is a microcircuitry database that includes published experimental reports. The database contains data tables but also a search board with manual or automatic relaxation of the search criteria. CoCoDat is used for:

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

The database is available for download under the terms of the GNU General Public License. The database is accessible and can be searched for information on:

- Brain region
- Layer
- Neuron type

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

- Equivalent dendrite
- Distal equivalent dendrite
- Middle equivalent dendrite
- Proximal equivalent dendrite
- Soma

Done
We need a proper infrastructure for a real Web of Data

- data is available on the Web
  - accessible via standard Web technologies
- data are interlinked over the Web
- ie, data can be integrated over the Web

This is where Semantic Web technologies come in
This is what we want!
In what follows...

- We will use a simplistic example to introduce the main Semantic Web concepts
The rough structure of data integration

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

The Glass Palace

The magnificent, poignant, fascinating novel of three generations that starts in Mandalay...
A simplified bookstore data (dataset “A”)

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Publisher's name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_qpr</td>
<td>Harper Collins</td>
<td>London</td>
</tr>
</tbody>
</table>
1st: export your data as a set of relations

The Glass Palace : a:title
2000 : a:year
London : a:city
Harper Collins : a:p_name
Ghosh, Amitav : a:name
http://...isbn/000651409X : a:author
http://www.amitavghosh.com : a:homepage
Some notes on the exporting the data

- Relations form a graph
  - the nodes refer to the “real” data or contain some literal
  - how the graph is represented in machine is immaterial for now
Amitav Ghosh

Le Palais des Miroirs
Another bookstore data (dataset “F”)

<table>
<thead>
<tr>
<th></th>
<th>ID</th>
<th>Titre</th>
<th>Traducteur</th>
<th>Original</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ID</th>
<th>Auteur</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ISBN 0–00–6511409–X</td>
<td>$A11$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Nom</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Ghosh, Amitav</td>
</tr>
<tr>
<td>12</td>
<td>Besse, Christianne</td>
</tr>
</tbody>
</table>
2nd: export your second set of data

http://…isbn/000651409X

Le palais des miroirs

http://…isbn/2020386682

Ghosh, Amitav

Besse, Christianne
3rd: start merging your data

The Glass Palace

2000

London

Harper Collins

Ghosh, Amitav

http://www.amitavghosh.com

http://isbn/000651409X

http://isbn/2020386682

Le palais des miroirs

Ghosh, Amitav

Besse, Christianne
3rd: start merging your data (cont)

The Glass Palace
http://...isbn/000651409X

2000

London
Harper Collins

http://www.amitavghosh.com

Ghosh, Amitav

Le palais des miroirs
http://...isbn/2020386682

Besse, Christianne

Ghosh, Amitav

Same URI!
3rd: start merging your data
Start making queries...

- User of data “F” can now ask queries like:
  - “give me the title of the original”
    - well, … « donnes-moi le titre de l’original »

- This information is not in the dataset “F”…

- …but can be retrieved by merging with dataset “A”!
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
3rd revisited: use the extra knowledge
Start making richer queries!

- User of dataset “F” can now query:
  - “donnes-moi la page d’accueil de l’auteur de l’original”
    - well… “give me the home page of the original’s ‘auteur’”
- 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”
Combine with different datasets

- Using, e.g., the “Person”, the dataset can be combined with other sources.

- For example, data in Wikipedia can be extracted using dedicated tools:
  - e.g., the “dbpedia” project can extract the “infobox” information from Wikipedia already…
Merge with Wikipedia data
Is that surprising?

- It may look like it but, in fact, it should not be…
- 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
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
What did we do?

Applications

Data represented in abstract format

Data in various formats

Manipulate

Query

Map,

Expose,

...
What did we do? (alternate view)
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.
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
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…
The Basis: RDF
RDF triples

Let us begin to formalize what we did!

- we “connected” the data...
- but a simple connection is not enough... data should be named somehow
- hence the RDF Triples: a labelled connection between two resources
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, RDFa, Json, …)
RDF triples (cont.)

- RDF triples are also referred to as “triplets”, or “statements”
- The “p” is sometimes referred to as “predicate”
RDF triples (cont.)

- Resources can use *any* URI
  - `http://www.example.org/file.html#home`
  - `http://www.example.org/f.xml#xpath(/q[@a=b])`
  - `http://www.example.org/form?a=b&c=d`

- RDF triples form a directed, labeled graph (the best way to think about them!)
A simple RDF example (in RDF/XML)

http://...isbn/2020386682

- f:titre: Le palais des miroirs
- f:original: http://...isbn/000651409X

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

(Note: namespaces are used to simplify the URI-s)
A simple RDF example (in Turtle)

```turtle
<http://.../isbn/2020386682>
  f:title "Le palais des miroirs"@fr ;
  f:original <http://.../isbn/000651409X> .
```
The book entitled "Le palais des miroirs" is the French translation of the "Glass Palace".
URI-s play a fundamental role

- URIs made the merge possible
- URIs ground RDF into the Web
  - information can be retrieved using existing tools
  - this makes the “Semantic Web”, well… “Semantic Web”
RDF/XML principles

- Encode nodes and edges as elements or literals:

```xml
<http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
  <http://.../isbn/000651409X>
  <http://.../isbn/2020386682>
```
RDF/XML principles (cont.)

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

```xml
<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>
```
Encode the properties (i.e., edges) in their own namespaces:

```xml
<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>
```
Examples of RDF/XML “simplifications”

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

```xml
<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
“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»?

[Diagram showing nodes and relationships with URIs]
One solution: create an extra URI

- The resource will be “visible” on the Web
  - care should be taken to define unique URI-s

```xml
<rdf:Description rdf:about="http://.../isbn/000651409X">
  <a:publisher rdf:resource="urn:uuid:f60ff40-307d-..."/>
</rdf:Description>
<rdf:Description rdf:about="urn:uuid:f60ff40-307d-...">
  <a:p_name>HarpersCollins</a:p_name>
  <a:city>HarpersCollins</a:city>
</rdf:Description>
```
Internal identifier ("blank nodes")

```html
<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".
```

- Internal = these resources are not visible outside
Blank nodes: the system can do it

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

```xml
<rdf:Description rdf:about="http://.../isbn/000651409X">
  <a:publisher>
    <rdf:Description>
      <a:p_name>HarpersCollins</a:p_name>
      ...
    </rdf:Description>
  </a:publisher>
</rdf:Description>
```
Same in Turtle

```
<http://.../isbn/000651409X> a:publisher [
  a:p_name "HarpersCollins";
  ...
].
```
More on blank nodes

- 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 URIs “on-the-fly”

- From a logic point of view, blank nodes represent an “existential” statement
  - “there is a resource such that…”
RDF in programming practice

- For example, using Python+RDFLib:
  - a “Graph” object is created
  - the RDF file is parsed and results stored in the Graph
  - the Graph offers methods to retrieve (or add):
    - triples
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
    - etc.
  - the rest is conventional programming…

- Similar tools exist in Java, PHP, etc.
# create a graph from a file
graph = rdflib.Graph()
graph.parse("filename.rdf", format="rdfxml")
# take subject with a known URI
subject = rdflib.URIRef("URI_of_Subject")
# process all properties and objects for this subject
for (s,p,o) in graph.triples(((subject,None,None))):
    do_something(p,o)
Merge in practice

- Environments merge graphs automatically
  - e.g., in Python+RDFLib, the Graph can load several files
  - the load merges the new statements automatically
One level higher up: RDFS, Datatypes
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...
Think of well known traditional vocabularies:
- 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”, …
Relationships are defined among 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
RDFS defines the meaning of these terms

- (these are all special URI-s, we just use the namespace abbreviation)
Schema example in RDF/XML

- The schema part:

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

- The RDF data on a specific novel:

```xml
<rdf:Description rdf:about="http://.../isbn/000651409X">
</rdf:Description>
```
An aside: typed nodes in RDF/XML

- A frequent simplification rule: instead of

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

  use:

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

  ie:

  ```xml
  <a:Novel rdf:about="http://.../isbn/000651409X">
    ...
  </a:Novel>
  ```
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?)
is not in the original RDF data…
…but can be inferred from the RDFS rules
RDFS environments return that triple, too
Inference: let us be formal...

- The RDF Semantics document has a list of (33) 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 .
Properties

- Property is a special class (rdf:Property)
  - properties are also resources identified by URI-s
- There is also a possibility for a “sub-property”
  - all resources bound by the “sub” are also bound by the other
- Range and domain of properties can be specified
  - i.e., what type of resources serve as object and subject
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:domain C) means:
  - P is a property
  - C is a class
  - when using P, I can infer that the “subject” is of type C
Property specification example
Property specification serialized

- In RDF/XML:

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

- In Turtle:

```turtle
:title
  rdf:type     rdf:Property;
  rdfs:domain :Fiction;
  rdfs:range  rdfs:Literal.
```
Again, new relations can be deduced. Indeed, if

\[
\text{:title} \\
\text{rdf:type} \quad \text{rdf:Property} \\
\text{rdfs:domain} \quad \text{:Fiction} \\
\text{rdfs:range} \quad \text{rdfs:Literal}.
\]

\[
\text{<http://.../isbn/000651409X> :title "The Glass Palace" .}
\]

then the system can infer that:

\[
\text{<http://.../isbn/000651409X> rdf:type :Fiction .}
\]
Literals

- Literals may have a data type
  - floats, integers, booleans, etc., defined in XML Schemas
  - full XML fragments
- (Natural) language can also be specified
Examples for datatypes

```
<rdf:Description rdf:about="http://.../isbn/000651409X">
    <page_number rdf:datatype="http://...#integer">543</page_number>
    <publ_date rdf:datatype="http://...#gYear">2000</publ_date>
    <price rdf:datatype="http://...#float">6.99</price>
</rdf:Description>
```

```
<http://.../isbn/000651409X>
    :page_number  "543"^^xsd:integer ;
    :publ_date    "2000"^^xsd:gYear ;
```
Examples for language tags

```xml
.rdf:Description rdf:about="http://.../isbn/000651409X"
    <title xml:lang="en">The Glass Palace</title>
    <fr:titre xml:lang="fr">Le palais des mirroirs</fr:titre>
</rdf:Description>
```

```xml
<http://.../isbn/000651409X>
    :title      "The Glass Palace"@en ;
    fr:titre    "Le palais des mirroirs"@fr .
```
XML Literals

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

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

- RDFS makes it possible to define vocabularies:
  - collection of properties and classes
  - relationships among those and to terms in other vocabularies

- Some examples:
  - Dublin Core terms: creator, date, …
  - FOAF terms: characterization of persons
  - Good Relations: eCommerce terms
  - Creative Commons: copyright classes, license relations, …
  - schema.org terms: events, organizations, places, reviews, …
  - …
Some predefined structures... (collections, containers)
Predefined classes and properties

- RDF(S) has some predefined classes and properties
- These 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
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
Collections (lists) (cont.)

- Familiar structure for Lisp programmers...
The same in RDF/XML and Turtle

```xml
<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> ...).
```
Sequences

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

- In RDF/XML:

```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:_1>
      </rdf:type>
    </rdf:Description>
  </a:consistsOf>
</rdf:Description>
```

- In Turtle:

```turtle
:Inventory
  a:consistsOf [
    rdf:type <http://.../rdf-syntax-ns#Seq>;
    rdf:_1 <http://.../isbn/000651409X>;
    ...
  ].
```
Sequences (simplified RDF/XML)

```xml
<rdf:Description rdf:about="#Inventory">
    <a:consistsOf>
        <rdf:Seq>
            <rdf:li rdf:resource="http://.../isbn/000651409X">
                ...
            </rdf:li>
        </rdf:Seq>
    </a:consistsOf>
</rdf:Description>
```
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
RDF Data and Web Pages?
(RDFa, microdata, microformats, GRDDL)
RDF with HTML

- Obviously, a huge source of information
- By adding some “meta” information, the same source can be reused for, e.g., data integration, better mashups, etc.
  - typical example: your personal information, like address, should be readable for humans and processable by machines
- Some solutions have emerged:
  - use microformats and convert the content into RDF
  - add extra statements in microdata, that can be converted to RDF
  - add RDF statements directly into XHTML via RDFa
Microformats

- Not a Semantic Web specification per se
  - there is a separate microformat community
- Approach: re-use (X)HTML attributes and elements to add “meta” information
  - typically @abbr, @class, @title, ...
  - different agreements for different applications
- It is not a specification per se, rather a common methodology
  - each vocabulary has its own, independent specification
Microformat example: hCalendar

- Goal: “markup” calendaring information on your HTML page
  - use a community agreement using, eg, :
    - @class for event name
    - abbr element for dates
    - @title for date values
    - etc.
- Automatic procedures (ie, calendaring applications) may then get to the right data
Microformat example: hCalendar

Dan Connolly
Research Scientist, MIT/CSAIL

connolly@w3.org
32 Vassar Street
Room 32-G506
Cambridge, MA, 02139 USA
+1-617-395-0241, DanC

hCard

standards: HTML WG, TAG, GRDDL WG, RDF Calendar, QA, DAWG/SPARQL, Semantic Web IG, OWL, HTML 2, ESW

research: breadcrumbs journal/weblog, cwm, N3, tabulator, PAW, TAMI, microformats open source

Dan Connolly is a research scientist at the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) in the Decentralized Information Group (DIG) and a member of the technical staff of the World Wide Web Consortium (W3C). His research interest is

Mar 7-11, 2008: to Austin, TX for SXSW Interactive

Apr 20-22: in Beijing, China for W3C AC meeting, linked data workshop, trip stuff

May 19-May 22: to Bristol for TAG iff, trip stuff

Sep 23 - 25: in Kansas City TAG meeting

Oct 20 - 25: to NCE for W3C TPAC

Nov 1 - 3: to BOS TAMI meeting

Dec 08 - 11: to Cambridge, MA TAG meeting

Feb 2 to 7: to Denver hoping to go for Web Directions North

Earlier travel/talks/events include Tools of
Behind the scenes...

```html
#31757B; color: #fff; text-decoration: none;" title=""><span
style="background: #000; border-right: 1px solid #000; color:
FFF; padding: 1px 0.75em; margin-right:
0.1em;">&amp;#8250;&amp;#8250;&amp;#8250;</span> hCalendar</a> Events

</h2>

<ol id="travl" class="sched my-events">

<li id="sxsw2008" class="vevent">
  <abbr class="dtstart" title="2008-03-07">Mar 7</abbr> - <abbr class="dtend" title="2008-03-12">11, 2008</abbr>: to <b class="location">Austin, TX</b> for <a class="summary url" href="http://2008.sxsw.com/interactive/">SXSW Interactive</a>
</li>

<li class="vevent" id="_6432">
  <abbr class="dtstart" title="2008-04-20">Apr 20</abbr> -
  <abbr class="dtend" title="2008-04-23">22</abbr>: to <b class="location">Beijing, China</b> for <a href="http://www.w3.org/Member/Meeting/" class="url"
```
To use it on the Semantic Web, microformat data should be converted to RDF

A simple transformation (eg, in XSLT) can be defined, yielding:

```xml
<http://www.w3.org/People/Connolly/#sxsw2008>
  a hcal:Vevent;
  hcal:organizer <http://www.w3.org/People/Connolly/#me>;
  hcal:summary "SXSW Interactive";
  hcal:dtstart "2008-03-07"^^xsd:date;
  hcal:dtend "2008-03-12"^^xsd:date;
  hcal:location "Austin, TX" .
```
So far so good, but...

- The XSLT transformation is hCalendar specific
  - each microformat dialect needs its own
- How does a general processor find the right transformation?
- Enter GRDDL
GRDDL: find the right transformation

- GRDDL defines
  - a few extra attribute values to locate the right transformation
  - a precise processing model on how the transformation should be applied to generate RDF

- Note: we describe GRDDL in terms of XHTML (and microformats) but GRDDL can be used for any XML data
GRDDL: find the right transformation
The GRDDL process: simple case
The GRDDL process: merging case

XHTML/XML

XSLT Script (dc-extract1.xsl)

W3C RDF

@transformation

XSLT Script (dc-extract2.xsl)

W3C RDF

W3C RDF
The GRDDL process: indirect case

- Namespace/profile
  XHTML/XML

- XSLT Script
  (dc-extract.xsl)

- XHTML/XML

- W3C RDF

@namespacetransformation
@profiletransformation
Microformats & GRDDL: pros

Pro: simple to define/add new vocabularies
  - there is a strong microformat community for this
Pro: works with all current browsers, markup validators, etc
Pro: fairly user friendly, easy to understand and use
Microformats & GRDDL: cons

- Cons:
  - does not scale well for complex vocabularies
    - remember: needs a transformation per vocabulary
  - difficult to mix vocabularies within one page
    - what if the usage of an attribute clashes among different vocabularies?
  - the “target” RDF vocabulary is to be defined additionally to the specific microformat specification
  - some of the attributes are meant for other usage
    - eg, the abbr element, the @title attribute, …
A more general solution: microdata and RDFa

- General idea: define a set of extra attributes to add “structured data” to an HTML page
- This extra structured data can
  - be used directly by a 3rd party (e.g., to improve the user interface)
  - extracted, converted into RDF and integrate it with the rest of data out there
- Two “syntaxes” emerged at W3C:
  - RDFa (developed in general for XML languages and for HTML)
  - microdata (part of the HTML5 family of specifications)
HTML+microdata or HTML+RDFa are becoming a major source of data on the Web!
Ivan Herman

Who am I?

I graduated as mathematician at the Eötvös Loránd University of Budapest, Hungary, in 1979. After a brief scholarship at the Université Paris VI I joined the Hungarian research institute in computer science (SZTAKI) where I worked for 6 years (and turned into a computer scientist...). I left Hungary in 1986 and, after a few years in industry in Munich, Germany, I joined the Centre Mathematics and Computer Sciences (CWI) in Amsterdam where I have a tenure position since 1988. I received a PhD degree in Computer Science in 1990 at the University of Leiden, in the Netherlands. I joined the World Wide Web Consortium (W3C) Team as Head of W3C Offices in January 2001 while maintaining my position at CWI. I served as Head of Offices until June 2006, when I was asked to take the Semantic Web Activity Lead position, which is now my principal work at W3C.

Before joining W3C I worked in quite different areas (distributed and dataflow programming, language design, system programming), but I spend most of my research years in computer graphics and information visualization. I also participated in various graphics related ISO standardization activities and software developments. My "professional" home page contains a list of my publications (see also my Mendeley account), my public presentations, and details of the various projects I participated in the past. There is also a dblp entry for my publications generated automatically (although I am not sure it is complete...). (B.t.w., based on my publications, my Erdős number is ≤ 4...)

In my previous life (i.e., before joining W3C...) I was member of the Executive Committee of the Eurographics Association for 15 years, and I was vice-chair of the Association between 2000 and 2002. I was the co-chair of the 6th World Wide Web Conference, in Amsterdam, May 2000; since then, I have also been member of IWWC (International World Wide Conference Committee), responsible for the World Wide Web Conference series. Since autumn 2007 I am also member of SWSA (Semantic Web Science Association), the committee responsible for the International Semantic Web Conferences (better known as "ISWC") series.

Some personal data

- The Hungarian spelling of my full name is Herman Iván. I.e. my name is Ivan (well, spelled properly: Iván) and my surname is Herman (many in the Netherlands and in Germany mix it up, and use "Herman" as my name... this is aggravated by the fact that, uniquely in Europe, the Hungarian custom is to put surname first).
- Nationalities: French and Hungarian
- Gender: male
- Family: I am married and have a son, David.
- Date and city of birth: 24th February, 1955, Budapest, Hungary
- Email addresses: ‘ivan’ on my own ivan-herman.net domain, ‘ivan’ on the w3.org domain, or ‘ivan.herman’ on the cwi.nl domain
- (Mobile) Phone: +31-641044153
- Skype ID: ivan_herman
- I live in Amstelveen (see also geonames), the Netherlands (lat: 52.302063, long: 4.87397). This is a suburb of Amsterdam. The closest airport is Amsterdam Schiphol.
- I am the administrator of the Semantic Web Activity Blog at W3C which can either be accessed directly or via its
Ivan Herman

Who am I?

I graduated at the Eötvös Loránd University of Budapest, Hungary, in 1979. After a brief scholarship at the Université Paris VI I joined the Hungarian research institute in computer science (SZTAKI) where I worked for 6 years (and turned into a computer scientist...). I left Hungary in 1986 and, after a few years in industry in Munich, Germany, I joined the Centre Mathematics and Computer Sciences (CWI) in Amsterdam where I have a tenure position since 1988. I received a PhD degree in Computer Science in 1990 at the University of Leiden, in the Netherlands. I joined the World Wide Web Consortium (W3C) Team as Head of W3C Offices in January 2001 while maintaining my position at CWI. I served as Head of Offices until June 2006, when I was asked to take the Semantic Web Activity Lead position, which is now my principal work at W3C.

Before joining W3C I worked in quite different areas (distributed and dataflow programming, language design, system programming), but I spend most of my research years in computer graphics and information visualization. I also participated in various graphics related ISO standardization activities and software developments. My “professional” home page contains a list of my publications (see also my Mendeley account), my public presentations, and details of the various projects I participated in the past. There is also a dblp entry for my publications generated automatically (although I am not sure it is complete...). (B.t.w., based on my publications, my Erdős number is ≤4...)

In my previous life (i.e., before joining W3C...) I was member of the Executive Committee of the Eurographics Association for 15 years, and I was vice-chair of the Association between 2000 and 2002. I was the co-chair of the 9th World Wide Web Conference, in Amsterdam, May 2000; since then, I have also been member of IW3C2 (International World Wide Web Conference Committee), responsible for the World Wide Web Conference series. Since autumn 2007 I am also member of SWSA (Semantic Web Science Association), the committee responsible for the International Semantic Web Conferences (better known as “ISWC”) series.

Some personal data

- The Hungarian spelling of my full name is Herman Iván, i.e. my name is Ivan (well, spelled properly: Iván) and my surname is Herman (many in the Netherlands and in Germany mix it up, and use “Herman” as my name... this is aggravated by the fact that, uniquely in Europe, the Hungarian custom is to put surname first).
- Nationalities: French and Hungarian
- Gender: male
- Family: I am married and have a son, David.
- Date and city of birth: 24th February, 1955, Budapest, Hungary
- Email addresses: ‘ivan’ on my own ivan-herman.net domain, ‘ivan’ on the w3.org domain, or ‘ivan.herman’ on the cwi.nl domain
- (Mobile) Phone: +31-641044153
- Skype ID: ivan_herman
- I live in Amstelveen (see also geoNames), the Netherlands (lat: 52.302663, long: 4.87397). This is a suburb of Amsterdam. The closest airport is Amsterdam Schiphol
- I am the administrator of the Semantic Web Activity Blog at W3C which can either be accessed directly or via its
Ivan Herman

Who am I?

I graduated as mathematician at the Bővös Loránd University of Budapest, Hungary, in 1979. After a two-year scholarship at the Université Paris VI I joined the Hungarian research institute in computer science (SZTAKI). I later worked for 6 years (and turned into a computer scientist...) in Germany, first in Munich, Germany, I joined the Centre Mathematics and Computer Sciences (CWI) in Amsterdam where I started working as a full professor in 1988. In 1999, I joined the World Wide Web Consortium (W3C) Team as Head of W3C Offices in January 2000 and Head of Offices until June 2006, when I was asked to take up the principal work at W3C.

Before joining W3C I worked in quite different roles, such as information visualization, system architecture, and programming, but I spend a lot of time in various international committees. In the latter role I presented a few keynotes and participated in the various committees of the Eurographics Association, such as the co-chair of the 9th World Wide Web Conference (WW3C2) (International World Wide Web Conferences). Since autumn 2002 I am also member of the International Semantic Web Conference Organizing Committees.

In my previous life (i.e., before joining W3C...).

Some personal data

The Hungarian spelling of my full name is Ivan, and my name is Iñán; I mix it up, and use "Herman" as my name... this is my custom is to put surname as my name... this is the normal spelling in the Netherlands and in Germany mix it up, and use "Herman" as my name... this is the normal spelling in the Netherlands and in Germany.

Nationality: Hungarian
I graduated as mathematician at the Hungarian University of Science and Technology (ELTE). I joined the World Wide Web Consortium (W3C) Team as Head of the SemiWeb Project (now known as the W3C Semantic Web Activity). Before joining W3C I worked in quite different areas (distributed and dataflow programming, language design, systems research). In my previous life (i.e., before joining W3C...) I was member of the Executive Committee of the European Association for Theoretical Computer Science (EATCS) and Secretary of the EATCS Editorial Board. Some personal data:

The Hungarian spelling of my full name is Herman Iván. I.e., my name is Ivan (well, spelled properly: Iván) and my surname is Herman. (many in the Netherlands and in Germany mix it up, and use “Herman” as my name... this is aggravated by the fact that my first name is in the feminine form).
In a slightly more readable format...

<html>
    ....
    <body prefix="schema: http://schema.org/
            foaf: http://xmlns.com/foaf/0.1/"
            
            <div about="http://www.ivan-herman.net/foaf#me" ... >
            ....
            <p>I graduated as mathematician at the
                <a rel="foaf:schoolHomepage schema:alumniOf"
                    href="http://www.elte.hu/">
                    <span property="dc:title">Eötvös Loránd University of
                        Budapest</span>
                </a>, ....
            </p>
    </div>.
</body>
In a slightly more readable format...

```html
<html>
....
<body prefix="schema: http://schema.org/
  foaf: http://xmlns.com/foaf/0.1/"
  
  <div about="http://www.ivan-herman.net/foaf#me" ... >
    ... 
    <p>I graduated as mathematician at the
       <a rel="foaf:schoolHomepage schema:alumniOf"
          href="http://www.elte.hu/">
           <span property="dc:title">Eötvös Loránd University of Budapest</span>
       </a>, ...
    </p>
  ... 
</body>
</html>
```
Yielding...

<http://www.ivan-herman.net/foaf#me>
  schema:alumniOf <http://www.elte.hu> ;
  foaf:schoolHomePage <http://www.elte.hu> ;
  schema:worksFor <http://www.w3.org/W3C#data> ;
...
<http://www.elte.hu>
  dc:title "Eötvös Loránd University of Budapest" .
...
<http://www.w3.org/W3C#data>
  dc:title "World Wide Web Consortium (W3C)"
...
Oscars 2012: The Artist, review

The Artist, an utterly beguiling silent, black-and-white celebration of early Hollywood won Best Picture at the Oscars 2012.
Oscars 2012: The Artist, review

The Artist, an utterly beguiling silent, black-and-white celebration of early Hollywood won Best Picture at the Oscars 2012.
Oscars 2012: The Artist, review - Telegraph

The Artist, an utterly beguiling silent, black-and-white celebration of early Hollywood won Best Picture at the Oscars 2012.
Oscars 2012: The Artist, review

The Artist, an utterly beguiling silent, black-and-white celebration of early Hollywood won Best Picture at the Oscars 2012.

Bérénice Bejo as ri...
In a slightly more readable format...

```
<div itemscope itemtype="http://schema.org/Review">
    ...
    <h1 itemprop="name">Oscars 2012: The Artist, review</h1>
    <h2 itemprop="description">The Artist, an utterly beguiling...</h2>
    ...
    <span itemprop="ratingValue" class="hidden">5</span>
    ...
```
Yielding...

[ rdf:type schema:Review ;
  schema:name "Oscars 2012: The Artist, review" ;
  schema:description "The Artist, an utterly beguiling..." ;
  schema:ratingValue "5" ;
...
]
RDFa and microdata: similarities

- Both have similar philosophies:
  - the structured data is expressed *via attributes only* (no specialized elements)
  - both define some special attributes
    - e.g., `itemscope` for microdata, `resource` or `about` for RDFa
  - both reuse *some* HTML core attributes (e.g., `href`)
  - both reuse the textual content of the HTML source, if needed

- RDF data can be extracted from both
RDFa and microdata: differences

- Microdata has been *optimized* for simpler use cases:
  - one vocabulary at a time
  - tree shaped data
  - no datatypes
- RDFa provides a full serialization of RDF in XML or HTML
  - the price is an extra complexity compared to microdata
- RDFa 1.1 Lite is a simplified authoring profile of RDFa, very similar to microdata
Typical usage of structured data

The Artist showtimes for Amsterdam
- Pathe Tuschinski - Reguliersbreestraat 26-32, Amsterdam - Map
  11:50 - 14:05 - 19:10
- Filmtheater "De Uitblik" - Prinsengracht 452, Amsterdam - Map
  12:15 - 19:00 - 21:15
- Filmtheater Radio - Calinuurbaan 338, Amsterdam - Map
  12:45

Show more theaters

The Artist (2011) - IMDb
www.imdb.com/title/tt18655442/
- Silent movie star George Valentin bemoans the coming era of talking ... Still of Jean Dujardin and Missi Pyle in The Artist Still of Bérénice Bajo in The Artist Reem ...

- Full cast and crew - The Artist Trailer (Official) ... - Bérénice Bejo - Jean Dujardin

The Artist (film) - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/The_Artist_(film)
The Artist is a 2011 French romantic comedy drama in the style of a black-and-white silent film written and directed by Michel Hazanavicius, starring Jean ...

- Jean Dujardin - Bérénice Bejo - Uggie - Dieguesis

The Artist Trailer 2011 HD - YouTube
www.youtube.com/watch?v=Q6K9AEcSIOE
25 Aug 2011 - 3 min - Uploaded by TrailersApple.com
I love how George Clooney, and Brad Pitt, lost the Best actor catagogy to this film. It just shows that there is ...

More videos for the artist movie »

Oscar 2012 - The Artist review - Telegraph
www.telegraph.co.uk › Culture › Film › Film reviews

5 stars
Review by Robbie Collin
8 Feb 2012 - The Artist, the final film I saw released in 2011 and also the most heart-wormingly joyous, is a silent movie screened in black and white and ...

The Artist is the perfect film about Hollywood! Harlay Freeman
How to "assign" RDF data to resources?
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
If I know the URI of the resource (photograph, publication, etc), how do I find the relevant RDF data?
The data might be embedded

- Some data formats allow the direct inclusion of (RDF) metadata:
  - SVG (Scalable Vector Graphics)
    - direct inclusion of RDF/XML
    - via RDFa attributes
  - HTML with RDFa
  - JPG files using the comment area and, eg, Adobe’s XMP technology
Simple linkage

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

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

- Similar facilities might exist in other formats (eg, SMIL)
POWDER provides for a more elaborate scenarios:

- 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
- define “description resources” that bind those resources to additional information
- 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), …
A POWDER scenario: copyright for photos

1. GET .... index
2. Return .... descr.xml
3. GET .... descr.xml
4. GET .... http://ex3.org/img/imgXXX.jpg
5. Deduce triplets

http://ex1.org/index
http://ex2.org/descr.xml
cc:license <http://cp...> for resources:
http://ex3.org/img/*

<http://www.ex3.org/img/imgXXX.jpg> cc:license <http://cp...>
The gory details...

- The “description resource” is an XML file:

```xml
<powder xmlns="http://www.w3.org/2007/05/powder#"
    xmlns:cc="http://creativecommons.org/ns#">

    <attribution>
        <issuedby src="http://www.ivan-herman.net/me"/>
    </attribution>

    <dr>
    
        <iriset>
            <includehosts>www.ex2.org</includehosts>
            <includepathstartswith>/img/</includepathstartswith>
        </iriset>

    </dr>

    <descriptorset>
    
        <cc:license rdf:resource="http://..."/>
    </descriptorset>

</powder>
```
The gory details...

- Powder processors may then return
  - direct RDF triples, eg:


- or can transform this XML file into an RDF (OWL) for more generic processors
  - a canonical processing of the XML file is defined by the POWDER specification
POWDER Service

- Online POWDER service can be set up:
  - a Web service with
    - submit a URI and a resource description file
    - return the RDF statements for that URI
  - such service should be set up, eg, at W3C
- A GRDDL transform is also defined
RDF makes a strong separation between:
- URI as an ID for a resource
- URI as a datatype (xsd:anyURI)
- there is no “bridge” between the two

POWDER includes a small extension to the formal semantics of RDF for two properties:
- wdrs:matchregex and wdrs:notmatchregex such that:
  - (R wdrs:matchregex Regex) holds iff the URI of R matches Regex
If you want the OWL version...

<> wdrs:issuedBy <http://www.ivan-herman.net/me> .

_:iriset_1 a owl:Class; owl:intersectionOf ( [ a owl:Restriction;   owl:onProperty wsdrl:matchregex ;   owl:hasValue ".. ugly regex for ex2.org"^^xsd:string ] [ a owl:Restriction;   owl:onProperty wsdrl:matchregex ;   owl:hasValue ".. ugly regex for /img"^^xsd:string ] ) .

_:desc_1 a owl:Restriction;   owl:onProperty cc:license;   owl:hasValue <http://...> .

_:iriset_1 rdfs:subClassOf _:desc_1 .
Consequences of the “hiccup”

- In practice this means that only “POWDER aware” agents can fully handle the description files
  - note that the extension is fairly easy to add, so it is not a big implementation issue...

- Existence of the services to provide the triplets automatically relieve the pain...
Other POWDER features

- 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
  - using tags to identify resources instead of URI patterns
Access to Relational Databases (Direct Mapping, R2RML)
Relational Databases and RDF

- Most of the data on the Web is, in fact, in RDB-s
- Proven technology, huge systems, many vendors…
- Data integration on the Web must provide access to RDB-s
RDF provides a common “view”
What is “export”?

- “Export” does not necessarily mean physical conversion
  - for very large databases a “duplication” would not be an option
  - systems may provide SQL “bridges” to make queries on the fly
- Result of export is a “logical” view of the RDB content
Simple export: Direct Mapping

- A standard RDF “view” of RDB tables
- Valid for all RDB-s, independently of the RDB schema
- Fundamental approach:
  - each row is turned into a series of triples with a common subject
  - column names provide the predicate names
  - cell contents are the objects as literals
  - linked tables are expressed with URI subjects
What DM processor does

- An DM processor has access to:
  - an RDB schema
  - a database governed by the schema
- … and produces an RDF graph
Back to the bookshop example

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

The Glass Palace
2000

http://...isbn/000651409X

Ghosh, Amitav

http://www.amitavghosh.com

a:title
a:year
a:author
a:name
a:homepage
Direct mapping of the bookshop tables

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>
Direct mapping of the bookshop tables

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

- “ISBN” and “ID” are keys
- Book table refers to the author table
Direct mapping of the bookshop tables

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

```
<Book/ISBN-0006511409X>
  <Book#Title>The Glass Palace</Book#Title>
  <Book#Year>2000</Book#Year>
<Book#ref-Author>id_xyz</Book#ref-Author>
```
Direct mapping of the bookshop tables

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>
Result of the Direct Mapping

- What do we have?
  - we have an RDF “view” of the two tables
- What do we miss?
  - an RDF view that is close to our application; a more “natural” view of the book data
Direct graph must be transformed

- Property names should be mapped
- URI-s should be minted
- Literals should be replaced by URI-s
Pros and cons of Direct Mapping

Pros:
- Direct Mapping is simple, does not require any other concepts
- know the Schema ⇒ know the RDF graph structure
- know the RDF graph structure ⇒ good idea of the Schema(!)

Cons:
- relies on other tools to get the “final” RDF graph
Enters R2RML

- Separate vocabulary to control the mapping
- Gets to the final RDF graph with one processing step
- Fundamentals are similar:
  - each row is turned into a series of triples with a common subject
- But:
  - cell contents *may* become a resource (not a literal)
  - there is a control over the generated URI-s
- Direct mapping is a “default” R2RML mapping
What R2RML processor does

- An R2RML processor has access to:
  - an RDB schema
  - an R2RML instance
  - a database governed by the schema
- ... and produces an RDF graph
Back to the bookshop example

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>
Back to the bookshop example

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

The Glass Palace

2000

http://...isbn/000651409X

http://www.amitavghosh.com
Back to the bookshop example

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

The Glass Palace
2000

http://...isbn/000651409X

Ghosh, Amitav
http://www.amitavghosh.com
Back to the bookshop example

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

http://...isbn/000651409X

The Glass Palace

2000

Ghosh, Amitav

http://www.amitavghosh.com

a:title

a:year

a:author

a:name

a:homepage
Step 1: transform “Person Table”

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

```xml
<Person>
  rr:tableName "Person_Table" ;
  rr:subjectMap [ rr:termtype rr:BlankNode ; ] ;
</Person>
```
Step 2: transform “Book Table”

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

```xml
<Book>
  rr:tableName "Book_Table" ;
  rr:subjectMap [ rr:template "http://...isbn/{ISBN}" ; ];
  rr:predicateObjectMap [ rr:predicate a:title ; rr:objectMap [ rr:column "Title" ] ];
  rr:predicateObjectMap [ rr:predicate a:year ; rr:objectMap [ rr:column "Year" ] ];
</Book>
```
Step 3: “bind” the two tables

<table>
<thead>
<tr>
<th>ISBN</th>
<th>Author</th>
<th>Title</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>id_xyz</td>
<td>The Glass Palace</td>
<td>id_qpr</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_xyz</td>
<td>Ghosh, Amitav</td>
<td><a href="http://www.amitavghosh.com">http://www.amitavghosh.com</a></td>
</tr>
</tbody>
</table>

```xml
<Book>
  ...
  rr:refPredicateObjectMap [  
    rr:predicate a:author ;  
    rr:objectMap [  
      rr:parentTriplesMap <Person> ;  
      rr:joinCondition  
        rr:child "Author" ;  
        rr:parent "ID"  
    ]  
  ]  
  
</Book>
```
Further R2RML features

- There are some additional features
  - assign a datatype to a literal object
  - more complicated object assignments (e.g., for a specific column the object is a cell of another column)

```xml
<Book>
  ...
</Book>
```
Further R2RML features: logical table

- Back to our example:

<table>
<thead>
<tr>
<th>ISBN</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>...</td>
</tr>
</tbody>
</table>

```
<Book>
  rr:tableName "Book_Table"
  rr:subjectMap [
    rr:template "http://...isbn/{ISBN}";
  ];
  ...
</Book>
```

http://...isbn/000651409X
An alternative could have been to use SQL
- generate a “logical table”
- all other definitions are on that logical table

Would be an overkill for our example, but can be very powerful for complicated cases!

```
<Book>
  rr:sqlQuery """Select
    ("http://...isbn/" || ISBN) AS id,
    Author, Title, Publisher, Year
  from Book_Table """ ;
</Book>
```

```
<table>
<thead>
<tr>
<th>ISBN</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006511409X</td>
<td>...</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://...isbn/0006511409X">http://...isbn/0006511409X</a></td>
<td>...</td>
</tr>
</tbody>
</table>
```

```
<Book>
  ...
  rr:subjectMap [
    ff:column "id"
  ];
</Book>
```
Linked Data
Linked Data “Project”

- Goal: “expose” datasets on the Web
  - remember the importance of data!
- Set links among the data items from different datasets
  - we want to avoid the silo effects
Is your data 5 Star?

Start with:

★ Available on the web (whatever format), but with an open license

★★ Available as machine-readable structured data (e.g., excel instead of an image scan)

★★★ As before, but using a non-proprietary format (e.g., CSV instead of excel)

★★★★ All the above, plus use open standards (RDF & Co.) to identify things, so that people could point at your stuff

★★★★★ All the above, plus link your data to other people’s data to provide context
Example data source: DBpedia

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

@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:website <http://www.amsterdam.nl>;
  dbterm:populationUrban "1364422";
  dbterm:areaTotalKm "219";
...

dbpedia:ABN_AMRO
  dbterm:location dbpedia:Amsterdam;
...
Automatic links among open datasets

Processors can switch automatically from one to the other…
The LOD “cloud”, March 2008
The LOD “cloud”, September 2008

As of September 2008

Courtesy of Richard Cyganiak and Anja Jentzsch
The LOD “cloud”, March 2009

Courtesy of Richard Cyganiak and Anja Jentzsch
The LOD “cloud”, June 2009

As of July 2009

Courtesy of Richard Cyganiak and Anja Jentzsch
The LOD “cloud”, September 2010

Courtesy of Richard Cyganiak and Anja Jentzsch
The LOD “cloud”, September 2011

Courtesy of Richard Cyganiak and Anja Jentzsch
Application specific portions of the cloud

- Eg, “bio” related datasets
  - done, partially, by the “Linking Open Drug Data” task force of the HCLS IG at W3C
The importance of Linked Data

- It provides a core set of data that Semantic Web applications can build on
  - stable references for “things”,
    - e.g., http://dbpedia.org/resource/Amsterdam
  - many many relationships that applications may reuse
    - e.g., the BBC application!
  - a “nucleus” for a larger, semantically enabled Web!

- For many, publishing data may be the first step into the world of Semantic Web
Some things to remember if you publish data

- Publish your data first, care about sexy user interfaces later!
  - the “raw data” can become useful on its own right and others may use it
  - you can add your added value later by providing nice user access

- If possible, publish your data in RDF but if you cannot, others may help you in conversions
  - trust the community...

- Add links to other data. “Just” publishing isn’t enough...
Currently, Linked Data is dominated by publishing data for read-only usage
  ▪ creating/updating the data is done “out of band”

The future to read and write Linked Data
Example: Simple Application Integration via Linked Data

Application #1

Linked Data on Server #1

HTTP GET, HEAD

Application #2

Linked Data on Server #2

HTTP PUT, DELETE
Example: Simple Application Integration via Linked Data

Public Library

Bookshop

HTTP GET, HEAD

HTTP PUT, DELETE

Catalogue in Library of Congress

Catalogue in the Centre Pompidou
Example: Simple Application Integration via Linked Data

Private Email Client

Corporate Email Client

HTTP GET, HEAD

HTTP PUT, DELETE

Private Address Data

Corporate Contact Data
Define an “entry” level, HTTP/RESTful based infrastructure to publish, read, write, or modify linked data

- typical usage: data intensive application in a browser, application integration using shared data...

The infrastructure should be easy to implement and install

- provides an “entry point” for Linked Data applications!

The work has just started to flesh out the details...
Provenance
We should be able to express all sorts of “meta” information on the data

- who played what role in creating the data (author, reviewer, etc.)
- view of the full revision chain of the data
- in case of data integration: which part comes from which original data and under what process
- what vocabularies/ontologies/rules were used to generate some portions of the data
- etc.
...the solution is more complicated

- Requires a complete model describing the various constituents (actors, revisions, etc.)
- The model should be usable with RDF
- Has to find a balance between
  - simple ("scruffy") provenance: easily usable and editable
  - complex ("complete") provenance: allows for a detailed reporting of origins, versions, etc.
- That is the role of the Provenance Working Group (started in 2011)
Query RDF: (SPARQL)
RDF data access

- The HTTP/RESTful approach is great to build up simple applications
- But as data grows, complexity of relationships grows
  - more sophisticated query possibilities are necessary!
- How do I query the RDF data?
  - e.g., how do I get to the DBpedia data?
Querying RDF graphs

- Remember the Python+RDFLib idiom:

```python
for (s,p,o) in graph.triples((subject, None, None)) :
    do_something(p,o);
```
Querying RDF graphs

- 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)
for (s, p, o) in graph.triples((subject, None, None)):
    do_something(p, o);
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 the bound resources
Our Python 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
Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
```
Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
```

Returns: [<http://isbn/000651409X>, 33, £]
Simple SPARQL example

SELECT ?isbn ?price ?currency # note: not ?x!

Returns: [<…409X>,33,:£], [<…409X>,50,:€]
Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
```

Returns: [<http://...isbn/2020386682>,60,:€], [<http://...isbn/000651409X>,50,:€], [<http://...isbn/000651409X>,33,:£]
Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
```

Returns: 

```
[<...409X>,33,:£], [<...409X>,50,:€],
[<...6682>,60,:€], [<...6682>,78,:$]
```
Pattern constraints

SELECT ?isbn ?price ?currency # note: not ?x!
FILTER(?currency == :€) }

Returns: [$\ldots$409X>, 50, :€], [$\ldots$6682>, 60, :€]
Optional pattern

```
  OPTIONAL ?wiki w:isbn ?isbn. }
```
Optional pattern

```
  OPTIONAL ?wiki w:isbn ?isbn. }
```

Returns: 

```
[<..09X>,33,:£,<$...Palace>], ..., [<..6682>,78,:$,
]```
Construct a new graph

    ?isbn a:author ?y. ?y a:name ?name . }
    FILTER(?currency == :€) }

http://...isbn/000651409X

a:price

rdf:value

33

p:currency

:£

http://...isbn/2020386682

a:price

rdf:value

60

p:currency

:€

Ghosh, Amitav

a:name

a:author

a:author

p:currency

:£

50

:€

78

:}$
Construct a new graph

    ?isbn a:author ?y. ?y a:name ?name . }
    FILTER(?currency == :€) }

http://...isbn/000651409X

http://...isbn/2020386682

Ghosh, Amitav

50

78
Other SPARQL features

- Limit the number of returned results; remove duplicates, sort them, …
- Specify several data sources (via URI-s) within the query
- Construct a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern
- Aggregation of the results (min, max, average, etc.)
- Path expressions (a bit like regular expressions)
SPARQL usage in practice

- SPARQL is usually used over the network
  - http request is sent to a SPARQL endpoint
  - result is the result of the SELECT, the CONSTRUCT,…

- 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
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>
SPARQL 1.1 Update

- SPARQL CONSTRUCT returns a new, modified graph
  - the original data remains unchanged!
- SPARQL 1.1 Update modifies the original dataset!
Update: insert

```
INSERT {?isbn rdf:type frbr:Work}
```
Update: insert

```sql
INSERT { ?isbn rdf:type frbr:Work } 
```
Update: delete

DELETE {?x p:currency ?currency}
Update: delete

```
DELETE { ?x p:currency ?currency }
```
SPARQL as a unifying point

- Triple store
- SPARQL Endpoint
- SPARQL Construct
- Application
- SPARQL Processor
- RDF Graph
- NLP Techniques
- RDFa
- GRDDL, RDFa
- SQL→RDF
- Relational Database
- HTML
- Unstructured Text
- XML/XHTML
- Database
SPARQL 1.1 as a unifying point

- SPARQL Endpoint
- SPARQL Construct
- SPARQL Update
- Application
- SPARQL Processor
- Triple store
- Database
- RDF
- RDFa
- NLP Techniques
- GRDDL, RDFa
- SQL-\text{RDF}
- Relational Database
- HTML
- Unstructured Text
- XML/XHTML
Remember the Direct Mapping issue?

CONSTRUCT {
  ?id a:title ?title ;
  a:year ?year ;
  a:author _:x .
  _:x a:name ?name ;
  a:homepage ?hp .
}
WHERE {
  ?book
  <Book#ISBN> ?isbn ;
  <Book#Title> ?title ;
  <Book#Year> ?year ;
  <Book#ref-Author> ?author .
  ?author
  <Person#Name> ?name ;
  <Person#Homepage> ?homepage .
  BIND (IRI(fn:concat("http://...",?isbn)) AS ?id)
  BIND (IRI(?homepage) AS ?hp)
}
Vocabularies

- Data integration needs agreements on
  - terms
    - “translator”, “author”
  - categories used
    - “Person”, “literature”
  - relationships among those
    - “an author is also a Person…”, “historical fiction is a narrower term than fiction”
    - ie, new relationships can be deduced
Vocabularies

- There is a need for “languages” to define such vocabularies
  - to define those vocabularies
  - to assign clear “semantics” on how new relationships can be deduced
But what about RDFS?

- Indeed RDFS *is* such framework:
  - there is typing, subtyping
  - properties can be put in a hierarchy
  - datatypes can be defined

- RDFS is enough for many vocabularies
  - we have seen some of those…

- But not for all!
Three technologies have emerged

- To re-use thesauri, glossaries, etc: **SKOS**
- To define more complex vocabularies with a strong logical underpinning: **OWL**
- Generic framework to define rules on terms and data: **RIF**
Thesauri, glossaries (SKOS)
SKOS

- Represent and share classifications, glossaries, thesauri, etc
  - for example:
    - Dewey Decimal Classification, Art and Architecture Thesaurus, ACM classification of keywords and terms...
    - classification/formalization of Web 2.0 type tags

- Define classes and properties to add those structures to an RDF universe
  - allow for a quick port of this traditional data, combine it with other data
Example: the term “Fiction”, as defined by the Library of Congress
Example: the term “Fiction”, as defined by the Library of Congress
Thesauri have identical structures...

- The structure of the LOC page is fairly typical
  - label, alternate label, narrower, broader, ...
  - there is even an ISO standard for these

- SKOS provides a basic structure to create an RDF representation of these
LOC's "Fiction" in SKOS/RDF
Usage of the LOC graph

Fiction

skos:broader

skos:prefLabel

Fiction

skos:broader

The Glass Palace

dc:title

http://.../isbn/...

dc:subject

skos:prefLabel

Historical Fiction

skos:broader

skos:Concept

rdf:type

Fiction

skos:broader

The Glass Palace

dc:title

http://.../isbn/...

dc:subject
Same serialized

<http://.../isbn/000651409X>
dc:title "The Glass Palace"@en;
dc:subject <http://id.loc.gov/authorities/sh85061165#concept>;
...

<http://id.loc.gov/authorities/sh85061165#concept>
a skos:Concept;
skos:prefLabel "Historical Fiction"@en;
skos:broader <http://id.loc.gov/authorities/sh85048050#concept>;
...

<http://id.loc.gov/authorities/sh85048050#concept>
a skos:Concept;
skos:prefLabel "Fiction"@en;
skos:narrower <http://id.loc.gov/authorities/sh85061165#concept>;
...
SKOS terms 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, ...)
Importance of SKOS

- SKOS provides a simple bridge between the “print world” and the (Semantic) Web
- Thesauri, glossaries, etc, from the library community can be made available
  - LOC is a good example
- SKOS can also be used to organize, eg, tags, annotate other vocabularies, …
Importance of SKOS

- Anybody in the World can refer to common concepts
  - they mean the same for everybody
- Applications may exploit the relationships among concepts
  - eg, SPARQL queries may be issued on the library data+LOC
Ontologies (OWL)
SKOS is not enough...

- SKOS may be used to provide simple vocabularies
- But it is not a complete solution
  - it concentrates on the concepts only
  - no characterization of properties in general
  - simple from a logical perspective
    - i.e., only a few inferences are possible
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.
Ontologies (cont.)

- The term ontologies is used in this respect:
  
  "defines the concepts and relationships used to describe and represent an area of knowledge"

- The term ontologies is used in this respect:
  - I.e., there is a need for Web Ontology Languages
  - RDFS can be considered as a simple ontology language

- Languages should be a compromise between
  - rich semantics for meaningful applications
  - feasibility, implementability
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
  - actually... there is a 2004 version of OWL ("OWL 1")
  - and there is an update ("OWL 2") published in 2009
  - this tutorial presupposes OWL 2
OWL is complex...

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

- 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?
  - owl:propertyDisjointWith
Term equivalences

- For individuals:
  - owl:sameAs: two URIs refer to the same concept ("individual")
  - owl:differentFrom: negation of owl:sameAs
Other example: connecting to French

- a:author ➔ owl:equivalentProperty ➔ f:auteur
- a:Novel ➔ owl:equivalentClass ➔ f:Roman
Typical usage of owl:sameAs

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

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

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

- In OWL, one can characterize the behavior of properties (symmetric, transitive, functional, inverse functional, reflexive, irreflexive, …)
- OWL also separates data and object properties
  - “datatype property” means that its range are typed literals
“foaf:email” may be defined as “inverse functional”
- i.e., two different subjects cannot have identical objects
What this means is...

- If the following holds in our triples:

  `:email rdf:type owl:InverseFunctionalProperty`. 
What this means is...

- If the following holds in our triples:

```plaintext
@email rdf:type owl:InverseFunctionalProperty.
<A> :email "mailto:a@b.c".
<B> :email "mailto:a@b.c".
```
What this means is...

- If the following holds in our triples:

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

then, processed through OWL, the following holds, too:

```plaintext
<A> owl:sameAs <B>.
```
Inverse properties

- There may be an inverse relationship among properties, eg:

\[
\text{<somebook> ex:author <somebody>.}
\text{ex:author owl:inverseOf ex:authorOf.}
\]

yields, in OWL:

\[
\text{<somebody> ex:authorOf <somebook>.}
\]
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:

\[
\text{ex:born\_in\_country} \text{ owl:propertyChainAxiom} \left( \text{ex:born\_in\_city} \text{ ex:city\_in\_country} \right).
\]

- More than two constituents can be used
- There are some restrictions to avoid "circular" specifications
Inverse functional properties are important for identification of individuals
  - think of the email examples

But… identification based on one property may not be enough
Keys

“if two persons have the same emails and the same homepages then they are identical”

- Identification is based on the identical values of two properties
- The rule applies to persons only
Previous rule in OWL

:Person rdf:type owl:Class;
  owl:hasKey (:email :homepage) .
What it means is...

If:

```xml
<A> rdf:type :Person ;
   :email "mailto:a@b.c";
   :homepage "http://www.ex.org".

<B> rdf:type :Person ;
   :email "mailto:a@b.c";
   :homepage "http://www.ex.org".
```

then, processed through OWL, the following holds, too:

```xml
<A> owl:sameAs <B>.
```
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 conceptual distinction between classes and individuals

- there is a separate term for owl:Class, to make the difference
- individuals are separated into a special class called owl:Thing

Eg, a precise classification would be:

```ex:Person rdf:type owl:Class.
<uri-for-Amitav-Ghosh>
  rdf:type owl:Thing;
  rdf:type owl:Person .```
OWL classes can be “enumerated”

- The OWL solution, where possible content is explicitly listed:
<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 (:€ :£ :$).
Union of classes

- Essentially, like a set-theoretical union:
Same serialized

```xml
<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>
```

```turtle
:Novel rdf:type owl:Class.
:Short_Story rdf:type owl:Class.
:Poetry rdf:type owl:Class.
:Literature rdf:type owl:Class;
```
For example...

If:

```logical
:Novel    rdf:type owl:Class.
:Short_Story rdf:type owl:Class.
:Poetry   rdf:type owl:Class.
:Literature rdf:type owl:Class;
```

then the following holds, too:

```logical
<myWork> rdf:type :Novel .
```

```logical
<myWork> rdf:type :Literature .
```
If:

:Novel rdf:type owl:Class.
:Short_Story rdf:type owl:Class.
:Poetry rdf:type owl:Class.
:Literature rdf:type owlClass;


<myWork> rdf:type fr:Roman .

then, through the combination of different terms, the following still holds:

<myWork> rdf:type :Literature .
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.
- Many inferred relationship can be found using a traditional rule engine.
Very large vocabularies might require even more complex features

- typical usage example: definition of all concepts in a healthcare environment
- some major issues
  - the way classes (i.e., “concepts”) are defined
  - handling of datatypes

OWL includes those extra features but... the inference engines become (much) more complex 😞
Property value restrictions

- Classes are created by restricting the property values on a (super)class
- 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 (€, £, 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”
Restrictions formally

- Defines a class 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

```rdfs
:Listed_Price rdfs:subClassOf [rdf:type owl:Restriction;
  owl:onProperty p:currency;
  owl:allValuesFrom :Currency.]
```
Possible usage...

If:

:Listed_Price rdfs:subClassOf [  
  rdf:type owl:Restriction;  
  owl:onProperty p:currency;  
  owl:allValuesFrom :Currency.  
].


:Price p:currency <someCurrency> .

then the following holds:

<someCurrency> rdf:type :Currency .
Other restrictions

- owl:allValuesFrom could be replaced by:
  - owl:someValuesFrom
    - e.g., I could have said: there should be a price given in at least one of those currencies
  - owl:hasValue, when restricted to one specific value
  - owl:hasSelf, for local reflexivity
Similar concept: cardinality restriction

- In a property restriction, the goal was to restrict the possible values of a property.
- In a cardinality restriction, the number of relations with that property is restricted.
  - “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).
Cardinality restriction

:Book_on_sale rdfs:subClassOf [  
  rdf:type owl:Restriction;  
  owl:onProperty p:price;  
  owl:minCardinality "1"^^xsd:integer. 
].

- could also be “owl:cardinality” or “owl:maxCardinality”
Qualified Cardinality Restriction

- Combining cardinality and the “all value” restriction
  - “there should be exactly two listed price tags with currency value”

```logic
:Listed_Price rdf:type owl:Class;
  rdfs:subClassOf [ rdf:type owl:Restriction;
    owl:onProperty p:currency;
    owl:onClass :Currency;
    owl:qualifiedCardinality "2"^^xsd:integer.
  ].
```
Datatypes in OWL

- RDF Literals can have a datatypes, OWL adopts those
- But more complex vocabularies require datatypes “restrictions”; eg, numeric intervals
  - “I am interested in a price range between €5 and €15”
Datatype restrictions

- For each datatype, there are possible restriction “facets”: min and max for numeric types, length for strings, etc
- These facets can be used to define new datatypess
Definition of a numeric interval

:AllowedPrice rdf:type rdfs:Datatype;
   owl:onDatatype xsd:float;
   owl:withRestriction (    
      [ xsd:minInclusive 5.0 ]    
      [ xsd:maxExclusive 15.0 ]    
   ) .
i.e.: an affordable book’s price is between 5.0 and 15.0
The combination of class constructions with various restrictions is extremely powerful.

What we have so far follows the same logic as before:
- extend the basic RDF and RDFS possibilities with new features
- define their semantics, i.e., what they “mean” in terms of relationships
- expect to infer new relationships based on those

However... a full inference procedure is hard 😞
- not implementable with simple rule engines, for example
OWL “species” or profiles

- OWL species comes to the fore:
  - restricting which terms can be used and under what circumstances (restrictions)
  - if one abides by those restrictions, then simpler inference engines can be used

- They reflect compromises: expressiveness vs. implementability
OWL Full

- No constraints on any of the 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, apply properties on them
    - etc.
- Extension of RDFS in all respects
- But: an OWL Full ontology may be, eg, inconsistent!
Here is a syntactically valid but inconsistent ontology:

```rdf
: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.
:C rdf:type :A .
```

If \( c \) is of type \( A \) then it must be in \( B \), but then it is in the complement of \( A \), i.e., it is not of type \( A \)…
Nevertheless OWL Full is important
- it gives a generic framework to express many things

Some application just need to express and interchange terms (with possible scruffiness)

Applications may control what terms are used and how
- in fact, they may define their own sub-language via, eg, a vocabulary (eg, SKOS!)
- thereby ensuring a manageable inference procedure
A number of restrictions are defined

- RDFS and OWL terms are reserved
  - no statements on RDFS and OWL resources
- user’s object properties must be among individuals only
- no characterization of datatype properties
- the same symbol, when used for an individual and a class does not mean full identity
- …

But: well known inference algorithms exist!
Examples for restrictions

- Given the statement:

  `<q> rdf:type <A>. # A is a class, q is an individual`

the followings are not “legal” OWL DL:

  `<A> ex:something <B>. # properties are for individuals only`

  `<q> ex:something <s>. # same property cannot be used as object…`

  `<p> ex:something "54". # … and datatype property`
Example for restriction on conceptual identity

- Same symbol may be used both for a class and an instance
- But not all “natural” inferences can be drawn
  - eg, although the following is ok:

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

when using OWL DL, this does **not** yield

```
qu : type B.
```
“DL” stands for “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”
OWL DL and Description Logic

- OWL DL can be interpreted as a variant of Description Logic
  - for connoisseurs: OWL (2) DL $\approx SROIQ(D)$
- Hence the results of this particular area of logic are directly applicable
Description Logic Formalism

- 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)
Description Logic Formalism

- There is also a compact mathematical notation for axioms, assertions, etc:
  - Literature $\equiv$ Novel $\sqsupset$ Short_Story $\sqsubseteq$ Poetry
  - Listed_Price $\sqsubseteq$ $\forall$ currency.Currencies
- You may see these in papers, books…
Abiding to the restrictions means that very large ontologies can be developed that require precise procedures
- eg, in the medical domain, biological research, energy industry, financial services (eg, XBRL), etc
- the number of classes and properties described this way can go up to the many thousands

OWL DL has become a language of choice to define and manage formal ontologies in general
- even if their usage is not necessarily on the Web
OWL also defines “profiles”

- Further restrictions on how terms can be used and what inferences can be expected
Goal: classification and instance queries in polynomial time

Suitable for
- very large number of classes and/or properties
- not require complex expressions
  - e.g.: SNOMED

Some excluded features (beyond those of DL)
- no cardinality restrictions, fewer property restrictions
- no inverse, reflexive, disjoint, symmetric, asymmetric, functional or inverse functional properties
- class disjunction
- ...
OWL profiles: QL

- Goal: conjunctive queries on top of relational databases (essentially: query rewriting to SQL)

- Suitable for
  - lightweight ontologies, but large data

- Some excluded features (beyond those of DL)
  - functional and inverse functional properties, sameAs, keys
  - fewer property restrictions
  - no cardinality restrictions
  - transitive properties, property chains
  - …
Goal: polynomial reasoning on top of rule engines

Suitable for
- relatively lightweight ontologies, but large data

Some excluded features
- fewer property restrictions
- fewer cardinality restrictions (at most 0/1)
- constraints on class expressions (union, intersections, etc) when used in subclass expressions
- no datatype restrictions
- ...
Some more on OWL RL

- Goal: to be implementable through rule engines
- Usage follows a similar approach to RDFS:
  - merge the ontology and the instance data into an RDF graph
  - use the rule engine to add new triples (as long as it is possible)
- This application model is very important for RDF based applications
What can be done in OWL RL?

- Many features are available:
  - identity of classes, instances, properties
  - subproperties, subclasses, domains, ranges
  - union and intersection of classes (though with some restrictions)
  - property characterizations (functional, symmetric, etc)
  - property chains
  - keys
  - some property restrictions (but not all inferences are possible)
- All examples so far could be inferred with OWL RL!
What cannot be done in OWL RL?

- There are restrictions on what can be a sub and superclass. Eg, the following is not manageable:

```
B rdf:type owl:Class;
  owl:unionOf (P Q R).
A rdfs:subClassOf B .
```

- Some features are not available or are restricted:
  - not all datatypes are available
  - no datatype restrictions
  - no minimum or exact cardinality restrictions
  - maximum cardinality only 0 or 1
What cannot be done in OWL RL?

- Some “natural” conclusions cannot be drawn, eg:

\[
\text{A } \text{rdf:type } \text{owl:Class;} \\
\text{owl:intersectionOf } (\text{U V S}).
\]


\text{does not yield:}

\[
\text{A } \text{rdfs:subClassOf U}.
\]
OWL constructs in RDF can be fairly verbose

There are alternative syntaxes to express ontologies
- direct XML encoding of ontologies (OWL/XML)
- “functional” syntax
- “Manchester” syntax

The official exchange syntax is RDF (RDF/XML)
- all other syntaxes are optional for tools
Functional syntax example

:£ rdf:type owl:Thing.
:€ rdf:type owl:Thing.
:$ rdf:type owl:Thing.

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

my:Listed_Price rdf:type owl:Class;
  rdfs:subClassOf [
    rdf:type owl:Restriction;
    owl:onProperty p:currency;
    owl:allValuesFrom my:Currency
  ].

is equal to:

Declaration(NamedIndividual(:£))
Declaration(NamedIndividual(:€))
Declaration(NamedIndividual(:$))

Declaration(Class(:Currency))
EquivalentClasses(:Currency one of (:€ :£ :$))

SubClassOf(my:Listed_Price AllValuesFrom(p:currency my:Currency))
Manchester syntax example

`:£ rdf:type owl:Thing.
:€ rdf:type owl:Thing.
:$ rdf:type owl:Thing.

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

my:Listed_Price rdf:type owl:Class;
   rdfs:subClassOf [
      rdf:type owl:Restriction;
      owl:onProperty p:currency;
      owl:allValuesFrom my:Currency
   ].

is equal to:

Individual: :€
Individual: :£
Individual: :$

Class: my:Currency EquivalentTo { :€ :£ :$}

Class: my:Listed_Price that p:currency only my:Currency
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
Ontology development (cont.)

- Ontologies should be shared and reused
  - can be via the simple namespace mechanisms…
  - …or via explicit imports

- Applications can also be developed with very small ontologies, though
Ontologies examples

- 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
Ginger Lemon Ice Cream

**Rules (RIF)**

1. Prepare #1 and #2 - set aside.
2. Make #3 Ginger Syrup - boil until all dissolved. Keep simmering for a few minutes. Set aside.
3. In a medium saucepan, cook #4 on moderate to high heat until simmering.
4. Remove saucepan from heat, stir in #3 and #2. Return to low heat for a few minutes.
5. Remove saucepan from heat, add in #1. Return to low heat for a few minutes, stir until thickened.
6. Slowly strain into Bowl #5. Add 1/8 tsp kosher salt and whisk for a while.
7. Put in fridge for 20+ minutes until cool.
8. Churn in ice cream maker for 20 minutes or until thickened.
9. In the last few minutes, slowly add #6. Churn for 5 more minutes.
10. Transfer to container. Freeze for a few hours or overnight.
11. ENJOY!
Why rules on the Semantic Web?

- Some conditions may be complicated in ontologies (i.e., OWL)
  - e.g., Horn rules: \((P1 \land P2 \land \ldots) \rightarrow C\)
- In many cases applications just want 2-3 rules to complete integration
- I.e., rules may be an alternative to (OWL based) ontologies
Rules

- There is also a long history of rule languages and rule-based systems
  - e.g., logic programming (Prolog), production rules
- Lots of small and large rule systems (from mail filters to expert systems)
- Hundreds of niche markets
An example from our bookshop integration:
- “I buy a novel with over 500 pages if it costs less than €20”
- something like (in an ad-hoc syntax):

```json
{
  ?x rdf:type p:Novel;
  p:page_number ?n;
  p:price [
    p:currency :€;
    rdf:value ?z
  ].
  ?n > "500"^^xsd:integer.
  ?z < "20.0"^^xsd:double.
}
=>
{ <me> p:buys ?x }
```
Things you may want to express
The goals of the RIF work:
- define simple rule language(s) for the (Semantic) Web
- define interchange formats for rule based systems

RIF defines several “dialects” of languages
RIF Core

- The simplest RIF “dialect”
- A Core document is
  - some directives like import, prefix settings for URI-s, etc
  - a sequence of logical implications
- RIF is not bound to RDF only
  - eg, relationships may involve more than 2 entities
RIF Core example

Document(
    Prefix(cpt http://example.com/concepts#)
    Prefix(ppl http://example.com/people#)
    Prefix(bks http://example.com/books#)

    Group
    (  
        Forall ?Buyer ?Item ?Seller (  
        )
        cpt:sell(ppl:John bks:LeRif ppl:Mary)
    )
)

This infers the following relationship:

cpt:buy(ppl:Mary bks:LeRif ppl:John)
Expressivity of RIF Core

- Formally: definite Horn without function symbols, a.k.a. “Datalog”
  - e.g., \( p(a,b,c) \) is fine, but \( p(f(a),b,c) \) is not
- Includes some extra features
  - built-in datatypes and predicates
  - notion of “local names”, a bit like RDF’s blank nodes
  - …
Expressivity of RIF Core

- There are also “safeness measures”
  - eg, variable in a consequent should be in the antecedent
    - this secures a straightforward implementation strategy (“forward chaining”)

- RIF Core is the simplest rule dialect
  - but covers a large percentage of RDF related use cases...
RIF Syntaxes

- RIF defines
  - a “presentation syntax”
    - a bit like the functional syntax for OWL
  - a standard XML syntax to encode and exchange the rules
  - there is a draft for expressing Core in RDF
    - just like OWL is represented in RDF
What about RDF and RIF?

- Typical scenario:
  - the “data” of the application is available in RDF
  - rules on that data is described using RIF
  - the two sets are “bound” (e.g., RIF “imports” the data)
    - if RIF is encoded in RDF, the two sets may be in the same file
  - a RIF processor produces new relationships
To make RIF/RDF work

- Some technical issues should be settled:
  - RDF facts/triples have to be representable in RIF
  - various constructions (typing, datatypes, lists) should be aligned
  - the formal semantics of the two worlds should be compatible
- There is a separate document that brings these together
Expressing RDF triples in RIF

- RIF has a "frame-based" syntax:
  
  \[ s[p->x \ q->z] \]

- This is used to represent the RDF triples:
  
  \[ s \ p \ x . \]
  \[ s \ q \ z . \]

- The rest is fairly obvious
### RIF/RDF harmonization

<table>
<thead>
<tr>
<th>RDF</th>
<th>↔</th>
<th>RIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>s p o .</td>
<td></td>
<td>s[p-&gt;o]</td>
</tr>
<tr>
<td>s rdf:type A .</td>
<td></td>
<td>A # B</td>
</tr>
<tr>
<td>(A B C)</td>
<td></td>
<td>List(A B C)</td>
</tr>
<tr>
<td>datatypes</td>
<td></td>
<td>datatypes</td>
</tr>
<tr>
<td>_:a</td>
<td></td>
<td>_a (local names)</td>
</tr>
<tr>
<td>(blank nodes)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Remember the what we wanted from Rules?

{?
x rdfs:typeof p:Novel;
   p:page_number ?n;
   p:price [p:currency :€;
            rdfs:value ?z]

?n > "500"^^xsd:integer.
?z < "20.0"^^xsd:double.
}
=>
{ <me> p:buys ?x }

Document {
  Prefix ...
  Group {
    Forall ?x ?n ?z {
      <me>[p:buys->]?x :-
      And(
        ?x rdf:type p:Novel
        ?x[p:page_number->]?n p:price->_abc
        _abc[p:currency->]:€ rdf:value->?z
        External(pred:numeric-greater-than(?n "500"^^xsd:integer))
        External(pred:numeric-less-than(?z "20.0"^^xsd:double))
      )
    }
  }
}
Discovering new relationships...

Forall ?x ?n ?z ( 
  <me>[p:buys->]?x] :- 
  And( 
    ?x # p:Novel 
    ?x[p:page_number->]?n p:price->_abc 
    _abc[p:currency->:€ rdf:value->]?z 
    External( pred:numeric-greater-than(?n "500"^^xsd:integer) ) 
    External( pred:numeric-less-than(?z "20.0"^^xsd:double) ) 
  ) 
)
Discovering new relationships...

For all ?x ?n ?z (  
   <me>[p:buys->]?x] :- 
   And( 
      ?x # p:Novel 
      ?x[p:page_number->]?n p:price->_abc 
      _abc[p:currency->p:€ rdf:value->]?z 
      External( pred:numeric-greater-than(?n "500"^^xsd:integer) ) 
      External( pred:numeric-less-than(?z "20.0"^^xsd:double) ) 
   )
)

combined with:

<http://.../isbn/...> a p:Novel; 
   p:page_number "600"^^xsd:integer ; 
Discovering new relationships...

For all ?x, ?n, ?z:
   <me>[p:buys->?x] :-
   And(
      ?x # p:Novel
      ?x[p:page_number->?n p:price->_abc]
      _abc[p:currency->p:€ rdf:value->?z]
      External( pred:numeric-greater-than(?n "500"^^xsd:integer) )
      External( pred:numeric-less-than(?z "20.0"^^xsd:double) )
   )

combined with:

<http://.../isbn/...> a p:Novel;
   p:page_number "600"^^xsd:integer;

yields:

<me> p:buys <http://.../isbn/...> .
A word on the syntax

- The RIF Presentation syntax is... only syntax
- It can express more than what RDF needs
- Hopefully, a syntax will emerge with
  - close to one of the RDF syntaxes with a better integration of rules
  - can be mapped on Core implementations
RIF vs. OWL?

- OWL concentrates on “taxonomic reasoning”
  - i.e., if you have large knowledge bases, ontologies, use OWL
- Rules concentrate on reasoning problems within the data
  - i.e., if your knowledge base is simple but lots of data, use rules
- But these are thumb rules only…
At the end of the day...

- Using rules vs. ontologies may largely depend on
  - available tools
  - personal technical experience and expertise
  - taste...
What about OWL RL?

- OWL RL stands for “Rule Language”…
- OWL RL is in the intersection of RIF Core and OWL
  - inferences in OWL RL can be expressed with rules
    - the rules are precisely described in the OWL specification
  - there are OWL RL implementations that are based on RIF
Why other RIF dialects?

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)
In an ideal world

Universal RULE exchange format and language

Rule system #1
Rule system #2
Rule system #3
Rule system #4
Rule system #5
Rule system #6
Rule system #7
Rule system #8
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”
A partial interchange...

```
Rule system #1

“Core”

Rule system #3

Rule system #4

Rule system #2
```
RIF “dialects”…

- Possible dialects: F-logic, fuzzy logic, …
The role of dialects
The role of dialects

Diagram showing relationships between dialects and rule systems.
The role of dialects

Rule system #1
Rule system #2
Rule system #3
Rule system #4
Rule system #5
Rule system #6
Rule system #7
Rule system #8
Rule system #9
Rule system #10

“Core”

Dialect #1
Dialect #2
Dialect #3
The role of dialects
The role of dialects
However...

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

- 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”
- The “Core”: shared subset of major language
Hierarchy of cores

RIF Core

RIF PLD

RIF BLD

Rule system #1
Rule system #2
Rule system #3
Rule system #4
Rule system #5
Rule system #6
Rule system #7
Rule system #8
Rule system #9
Rule system #10
A rough division of features...

- **Objects**
- **Print**
- **New**
- **Retract**
- **Actions**

- **XML Data**
- **iNeg**

- **Default Negation**
- **Open lists**
- **Builtins**
- **Lists**
- **Annotations**
- **Datalog Rules**
- **IRI Terms**

- **Classical Negation**
- **Equality**
- **Horn Rules (functions)**

- **Core**
- **Import**
- **XML Datatypes**
- **Frames (Triples)**

Courtesy of Sandro Hawke, W3C
Remember this?

CONSTRUCT {
    ?id a:title ?title ;
    a:year ?year ;
    a:author _:x .
    _:x a:name ?name ;
    a:homepage ?hp .
}
WHERE {
    ?book
    <Book#ISBN> ?isbn ;  
    <Book#Title> ?title ;  
    <Book#Year> ?year ;  
    <Book#Author> ?author .
    ?author
    <Person#Name> ?name ;
    <Person#Homepage> ?homepage .
    BIND (IRI(fn:concat("http://...",?isbn)) AS ?id)
    BIND (IRI(?homepage) AS ?hp)
}
Same with RIF Core
Question: how does SPARQL queries and vocabularies work together?
- RDFS, OWL, and RIF produce new relationships
- on what data do we query?

Answer: in current SPARQL, that is not defined 😞
But, in SPARQL 1.1 it is… 😊
SPARQL 1.1 and RDFS/OWL/RIF

- RDF Data
- RDFS/OWL/RIF data
- SPARQL Pattern

SPARQL Engine with entailment

entailment

pattern matching

RDF Data with extra triples

SPARQL Pattern

Query result
SPARQL 1.1 as a unifying point

SPARQL Processor

- SPARQL Construct
- SPARQL Update

Application

- SPARQL Construct
- SPARQL Update

Triple store

Inferencing

RDF Graph

HTML

Unstructured Text

XML/XHTML

SPARQL Endpoint

Database

NLP Techniques

RDFa

GRDDL, RDFa

SQL→RDF

Inferencing

Relational Database
What have we achieved?
(putting all together)
Remember the integration example?

Data in various formats

Data represented in abstract format

Applications

Manipulate Query ...

Map, Expose, ...

http://www.example.com

http://www.example.com

Applications

http://www.example.com
Data in various formats

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

Applications

RESTful API
SPARQL, Inferences
…

R2RML, RDFa,
…

Same with what we learned
Remember the integration example?
The same with what we learned
What else may be on the horizon?
Quality constraints on graphs
  “may I be sure that certain patterns are present in a graph?”
Quality control of the content of the graphs
  are links really what they should be?
User interfaces for exploring large datasets
Access control issues on a Web of Data
Some items, issues, questions, ...

- Ontology merging, alignment, term equivalences, versioning, development, ...

- Scaling:
  - what does it mean to “reason” and “infer” on graphs with millions of triples?
  - how can one systematically “explore” a huge Web of Data
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.
A different aspect of naming: what is the URI for a specific entity (regardless of the technical details)

- what is the unique URI for, e.g., Bach’s Well-Tempered Clavier?
  - obviously important for music ontologies and 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
Some items, issues, questions, ...

- 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 has issued a report on the current status, possibilities, directions, etc.
  - possible RIF dialect for fuzzy logic, for example?
Available documents, resources
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; RIF Primer is on its way
- The W3C Semantic Web Activity Wiki has links to all the specifications
“Core” vocabularies

- There are also a number “core vocabularies”
  - 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!
Some books

- T. Heath and C. Bizer: *Linked Data: Evolving the Web into a Global Data Space*, 2011
- M. Watson: *Practical Semantic Web and Linked data Applications*, 2010
- …

See the separate [Wiki page collecting book references](http://example.com/wikireferences)
Further information

- Planet RDF aggregates a number of SW blogs:
  - [http://planetrdf.com/](http://planetrdf.com/)

- Semantic Web Interest Group
  - a forum developers with a publicly archived mailing list, and a constant IRC presence on freenode.net#swig
  - anybody can sign up on the list
    - [http://www.w3.org/2001/sw/interest/](http://www.w3.org/2001/sw/interest/)

- Linked Data mailing list
  - a forum concentrating on linked data with a public archive
  - anybody can sign up on the list
    - [http://lists.w3.org/Archives/Public/public-lod/](http://lists.w3.org/Archives/Public/public-lod/)
Great community...

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.
Lots of Tools (not an exhaustive list!)

- Some names:
  - Jena, AllegroGraph, Mulgara, Sesame, flickurl, 4Store, Stardog, ...
  - TopBraid Suite, Virtuoso environment, Falcon, Drupal 7, Redland, Pellet, ...
  - Disco, Oracle 11g, RacerPro, IODT, Ontobroker, OWLIM, Talis Platform, ...
  - RDF Gateway, RDFLib, Open Anzo, DartGrid, Zitgist, Ontotext, Protégé, ...
  - Thetus publisher, SemanticWorks, SWI-Prolog, RDFStore...

- Categories:
  - Triple Stores
  - Inference engines
  - Converters
  - Search engines
  - Middleware
  - CMS
  - Semantic Web browsers
  - Development environments
  - Semantic Wikis
  - ...

More on http://www.w3.org/2001/sw/wiki/Tools
Deployment, applications
See the separate slide set...
Conclusions
To remember…

- *Data* on the Web is a major challenge
  - technologies are needed to use them, to interact with them, to integrate them
- Semantic Web technologies, Linked Data principles and practices, etc., should play a major role in publishing and using Data on the Web
Lot remain to be done...

- Lots of issues to be solved
- But... W3C needs experts!
  - consider joining W3C, as well as the work done there!
Thank you for your attention

These slides are also available on the Web:

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