Untangling the Semantic Web

Alun Preece

http://www.csd.abdn.ac.uk/~apreece

The Semantic Web defined

"An extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

"It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications."

Tim Berners-Lee, James Hendler and Ora Lassila
The Semantic Web, Scientific American, May 2001

The SW: up for grabs

When they look at the SW, different people see different things:

- WWW community: a means to enrich the way information is interlinked on the global network
- Knowledge engineers: the potential to automate problem-solving and information-seeking tasks
- Logicians and ontologists: the challenge of representing meaning and performing inference on a massive scale.

This tutorial aims to untangle these various perspectives on the Semantic Web, while demonstrating that they are essentially compatible.

Roadmap

Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

Roadmap

Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

Starting point: metadata

Metadata is "data about data" – it describes the resources available in an information system.

On the Web, metadata is information about the resources available on the Web.

Metadata makes those resources machine-processable – it facilitates the finding & fusing of information on the Web.

As the Web grows in size, and also grows in importance to business, the value of metadata becomes greater.

Websites using metadata become more useful than those that don't.

The increased use of metadata, and the corresponding improvements in Web services, are now seen as a key feature of the next stage in the Web's evolution.

Metadata in HTML

The original approach to adding metadata to Web resources was the HTML **<meta>** element.

The **<meta>** element is used to add

- keywords
- descriptive summaries

to Web page headers, for use by search engines and Website management tools.

The content of **<meta>** tags is just plain text – processable in limited ways.

Still, some sites (e.g. BBC news) push this technique to its limits:

```
<meta name="Headline" content="Creator of the web turns knight" />
<meta name="Section" content="Technology" />
<meta name="Description" content="Tim Berners-Lee, the &#34;father of the web&#34;, officially receives his knighthood from the Queen." />
```

XML: making <meta> better

Consider searching the Web for "pages created by Tim Berners-Lee".

One page - the BBC news example - contains a <meta> tag:

```
<meta name="Description" content="Tim Berners-Lee, the &#34;father of the web&#34;, officially receives his knighthood from the Queen." />
```

Another page may contain some metadata marked-up with an XML element:

```
<creator>Tim Berners-Lee</creator>
```

Clearly, given a well-defined interpretation for the <creator> element, a search engine that uses the second piece of metadata will return a more accurate result.

We need vocabularies of metadata terms that we can use to mark-up Web resources with XML elements that correspond to these terms….

Metadata example: vCard

vCard is a standard for electronic business cards.

A vCard typically includes:

name	organisation
address	department
email	URI
phone & fax numbers	

The original encoding was as a MIME type for attachment to email messages.

But this is processable only by software programmed to recognise the specific vCard syntax.

More recently, XML encodings of the vCard vocabulary have been defined.

These allow more universal access to the various fields of a vCard, and also allow the vCard data to be embedded in Web pages.

An example vCard

The following (abbreviated) example uses the W3C RDF vCard XML encoding:

```
<rdf:RDF  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:vc = "http://www.w3.org/2001/vcard-rdf/3.0#">

 <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece">
  <vc:FN>Alun Preece</vc:FN>
  <vc:PHOTO>http://www.csd.abdn.ac.uk/~apreece/images/
     apreece.jpg</vc:PHOTO>
  <vc:EMAIL>apreece@csd.abdn.ac.uk</vc:EMAIL>
  <vc:TITLE>Senior Lecturer</vc:TITLE>
 </rdf:Description>

</rdf:RDF>
```

Full version: http://www.csd.abdn.ac.uk/~apreece/apreece.rdf

Metadata example: Dublin Core

The Dublin Core Metadata Initiative originated with the library community, intended to cover the properties of information artefacts in a library (including digital libraries).

The DC element set spans the contents of an electronic "library card":

title	date
creator	type
subject	format
description	language
publisher	etc

Most Web information resources have these common properties, so the Dublin Core element set has wide applicability to Web metadata….

RDF

RDF – the Resource Description Framework – is the W3C's data model for representing metadata about Web resources.

The RDF specification has several components, including:

- the data model
- an XML serialization syntax

The RDF data model is similar to an entity-relational model, using:

- URIs to unambiguously denote entities (objects)
- properties to describe relationships between entities, and also information about entities (literal text)

Metadata about a specific resource (URI) is expressed in RDF as a collection of properties associated with that resource.

To be useful, these properties are usually from a standard source, such as vCard (as we've seen) or Dublin Core….

RDF example (1)

We'd like to say in RDF that the resource
http://www.csd.abdn.ac.uk/~apreece/talks/UntanglingTheSemanticWeb
has the following properties

- it's the AI-2004 tutorial slides (literal text)
- it's created by Alun Preece (an entity)

The Dublin Core metadata standard covers this sort of thing:

- its description property covers the former
- its creator property covers the latter

To use these properties in RDF, we need to refer to them unambiguously as URIs (so they're globally unique):

- http://purl.org/dc/elements/1.1/description
- http://purl.org/dc/elements/1.1/creator

We also need to refer unambiguously to "Alun Preece" by a URI:

- http://www.csd.abdn.ac.uk/~apreece

RDF example (2)

We'll use the standard XML syntax for RDF (though this is not the only syntax). An XML RDF document has the following structure:

```
<?xml version = "1.0"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          <!– any other namespaces go here –> >
  <!– then a sequence of one or more RDF Description elements –>
  <rdf:Description … > … </rdf:Description>
  …
</rdf:RDF>
```

Each RDF Description element:

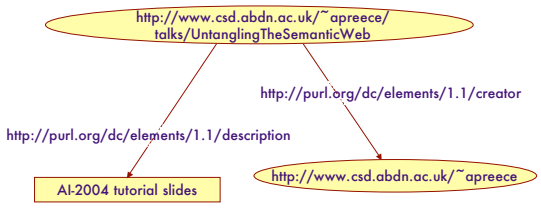
- has an attribute indicating the resource it describes, for example
  <rdf:Description about="http:www.somewhere.com/something">
- contains the property elements for that resource

RDF example (3)

```
<?xml version = "1.0"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece/
        talks/UntanglingTheSemanticWeb">
    <dc:description>AI-2004 tutorial slides</dc:description>
    <dc:creator rdf:resource="http://www.csd.abdn.ac.uk/~apreece"/>
  </rdf:Description>
</rdf:RDF>
```

These 2 properties are associated with the Dublin Core namespace

RDF graphical syntax

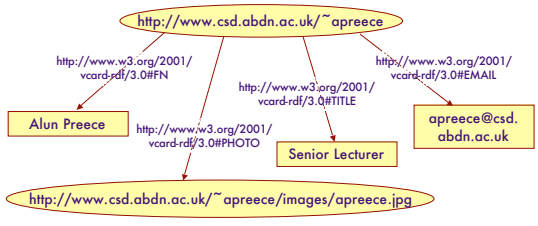


Note: this is a valid alternative syntax for RDF. The meaning of the graph is the same as the previous XML version.

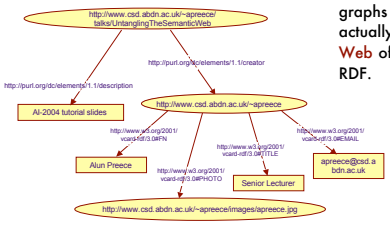
resource
text ("literal")
property

vCard example revisited

The vCard RDF example can also be shown graphically:



A Web of RDF



The two example graphs are actually part of a Web of linked RDF.

Merging the RDF in XML (1)

```
<rdf:RDF  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:vc="http://www.w3.org/2001/vcard-rdf/3.0#">
 <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece/
          talks/UntanglingTheSemanticWeb">
  <dc:description>AI-2004 tutorial slides</dc:description>
  <dc:creator rdf:resource="http://www.csd.abdn.ac.uk/~apreece"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece">
  <vc:FN>Alun Preece</vc:FN>
  <vc:PHOTO>http://www.csd.abdn.ac.uk/~apreece/images/apreece.jpg
     </vc:PHOTO>
  <vc:EMAIL>apreece@csd.abdn.ac.uk</vc:EMAIL>
  <vc:TITLE>Senior Lecturer</vc:TITLE>
 </rdf:Description>
</rdf:RDF>
```

Merging the RDF in XML (2)

```
<rdf:RDF  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:vc="http://www.w3.org/2001/vcard-rdf/3.0#">
 <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece/
          talks/UntanglingTheSemanticWeb">
  <dc:description>AI-2004 tutorial slides</dc:description>
  <dc:creator>
   <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece">
    <vc:FN>Alun Preece</vc:FN>
    <vc:PHOTO>http://www.csd.abdn.ac.uk/~apreece/images/
       apreece.jpg</vc:PHOTO>
    <vc:EMAIL>apreece@csd.abdn.ac.uk</vc:EMAIL>
    <vc:TITLE>Senior Lecturer</vc:TITLE>
   </rdf:Description>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

RDF: global, distributed metadata

In summary, the RDF specifications allow us to:

- Represent relations between URIs (and literals)
- Define those relations globally as URIs
- Express this information in a universal syntax (XML)

These features make RDF a global, distributed metadata format.

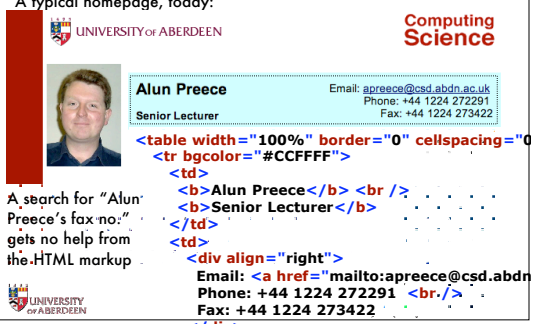
Using RDF, Web authors can state metadata on their pages, which can be used to facilitate:

- More effective web searching (for example, "knowing" someone's email address, or "knowing" who created a page)
- More effective content management (for example, checking that every person's home page includes an email address & job title

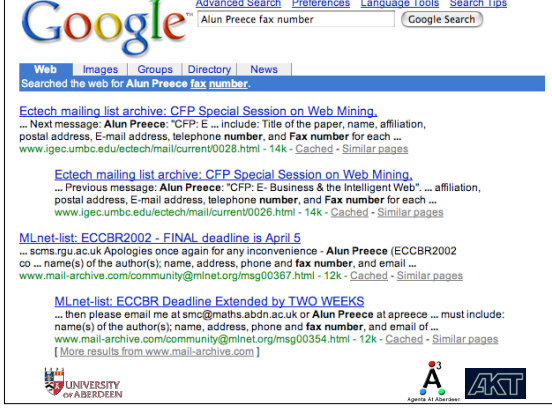
(Also, through the XML syntax, RDF can be styled and presented in a reader-friendly way.)

The syntactic Web 1: HTML

A typical homepage, today:



A search for "Alun Preece's fax no:" gets no help from the HTML markup



The syntactic Web 2: XML

With XML, custom sets of tags can be defined, which can be:

- "styled" into HTML
- processed directly by software - so can be queried...

Alun Preece
Email: apreece@csd.abdn.ac.uk
Phone: +44 1224 272291
Fax: +44 1224 273422

```
<Xfgpds ID="apreece">
  <ghew>Alun Preece</ghew>
  <ngre resource="mailto:apreece@csd.abdn.ac.uk" />
  <hytcx resource="tel:+44-1224-272291" />
  <hytcx resource="fax:+44-1224-273422" />
</Xfgpds>
```

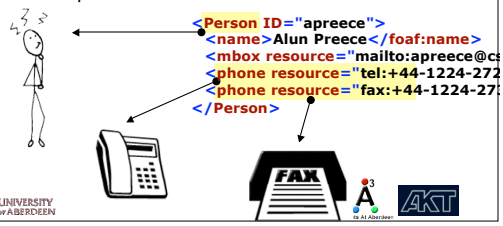
XML is still just syntax - the tags don't mean anything!

The semantic Web

XML alone is not enough - the XML tags need a defined semantics, to make them meaningful.

Formally, to relate the tag symbols to the things they represent in the real world.

So the markup becomes a model of the real world.



Two Webs in parallel

The Semantic Web is not a replacement for the current Web

Semantic markup is designed to exist alongside HTML markup:

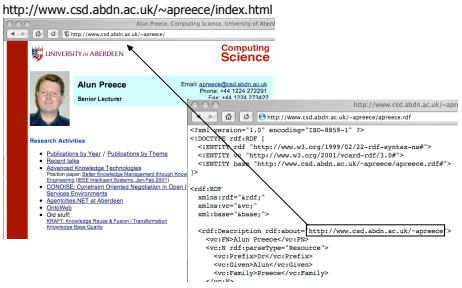
- People will continue to view the HTML
- Software can process the Semantic markup
- Hence the W3C's aim: "better enabling computers and people to work in cooperation"

Because the Semantic markup is in XML, it is possible to generate HTML from it: information readily flows from the SW…

Moreover, information on the SW exists independently of the various ways it can be processed…

Web/Semantic Web example

http://www.csd.abdn.ac.uk/~apreece/index.html



http://www.csd.abdn.ac.uk/~apreece/index.rdf

Roadmap

Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

RDF recap

As we've seen, RDF distinguishes several kinds of thing:

- resources, denoted by URIs
- properties, a kind of resource, also denoted by URIs
- literals – strings of plain text

We saw that an RDF description consists of:

- a resource being described
- a collection of properties of that resource
- a value for each of the properties – either a literal or another resource

We refer to a collection of RDF descriptions as an RDF model.

We also saw that we can express RDF descriptions in several syntaxes, including XML and graphics.

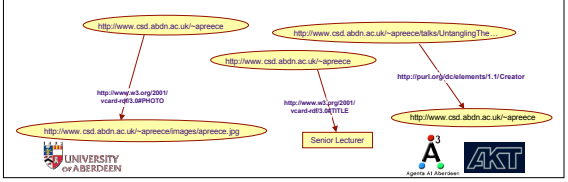
RDF statements

An RDF model is comprised of statements, each of which expresses a binary relationship.

Formally, an RDF statement has 3 parts – we call them triples:

- subject – the source of the relationship (a resource)
- predicate - the relationship itself (a property)
- object - the destination of the relationship (a resource or a literal)

RDF statements/triples are easy to visualise in the graph syntax:



RDF data model

The basic components of RDF – resources, properties, literals – form a simple data model.

(The semantics of this data model are what distinguish RDF fundamentally from XML's tree-based data model, which does not distinguish entities from relations.)

Its simplicity allows the RDF data model to become a "lowest common denominator" for integrating information from disparate data models.

However, the RDF data model alone is not enough to define:

- classes (types) of resources
- which properties make sense for which class of resource

Consequently, we can't use RDF alone to define common terms for specific metadata domains.

In short, we want to define schemas for RDF.

RDF Schema

The RDF Schema (RDFS) specification provides the apparatus for defining:

- classes of RDF resource, and their superclasses (defining class hierarchies)
- properties, including their domain (source) and range (destination)

This is enough to define a simple datatype system, similar to object-oriented and entity-relational systems.

RDFS is an extension of the basic RDF specification. Components of the two specifications have their individual namespace URIs, with conventional prefixes (in XML syntax):

```
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

RDFS data model: main elements

rdfs:Class
- the class of all classes
- all classes have an rdf:type of rdfs:Class

rdfs:Resource
- the class of all resources

rdf:Property
- the class of all properties

rdf:type
- the property relating a resource to a class
- if resource *R* is of rdf:type C, then we say *R* is an instance of class C
- a resource may have more than one rdf:type

rdfs:subClassOf
- the property relating a class to its parent (super)class
- transitive: if C1 is rdfs:subClassOf C2 and C2 is rdf:subClassOf C3, then C1 is rdfs:subClassOf C3

RDFS statements

RDFS statements are expressed in RDF triple form (subject, predicate, object).

As an example, we'll define a simple schema for University staff, starting with some classes:

Subject	Predicate	Object
AcademicPerson	rdf:type	rdfs:Class
Lecturer	rdf:type	rdfs:Class
Lecturer	rdfs:subClassOf	AcademicPerson
TeachingFellow	rdf:type	rdfs:Class
TeachingFellow	rdfs:subClassOf	AcademicPerson

RDFS example in XML syntax: classes

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

<rdf:Description rdf:about="Lecturer">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#AcademicPerson" />
</rdf:Description>

<rdf:Description rdf:about="TeachingFellow">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#AcademicPerson" />
</rdf:Description>
```

Naming RDF resources

The terms we've just defined – AcademicPerson, Lecturer, etc – are relative to the document base. Here, the base is defined to be:

http://www.csd.abdn.ac.uk/teaching/levelfive/CS5008/examples/rdf/people.rdfs#

So when we refer to AcademicPerson within the schema, we're actually referring to a resource with a globally-qualified URI:

http://www.csd.abdn.ac.uk/teaching/levelfive/CS5008/examples/rdf/people.rdfs#AcademicPerson

Finally, note how references to our defined terms within the schema are made using named-anchor notation:

#AcademicPerson

Abbreviated XML rdf:type syntax

When an rdf:Description is given with an rdf:type property, an abbreviated syntax form allows us to substitute the referenced type in place of the rdf:Description element.

So, the previous examples now become more readable:

```
<rdfs:Class rdf:about="AcademicPerson" />

<rdfs:Class rdf:about="Lecturer">
  <rdfs:subClassOf rdf:resource="#AcademicPerson" />
</rdfs:Class>

<rdfs:Class rdf:about="TeachingFellow">
  <rdfs:subClassOf rdf:resource="#AcademicPerson" />
</rdfs:Class>
```

Properties of RDFS properties

In RDFS, there are a number of properties defined on the class rdf:Property:

rdfs:subPropertyOf

- specifies that one property is a specialisation of another – example: hasDog might be rdfs:subPropertyOf hasPet
- transitive: if *P1* is rdfs:subPropertyOf *P2* and *P2* is rdf:subPropertyOf *P3*, then *P1* is rdfs:subPropertyOf *C3*

rdfs:range

- specifies a range constraint on a property: that the value of the property must be an instance of a specified class
- example: hasPet might have rdfs:range class Pet

rdfs:domain

- specifies a domain constraint on a property: that the property can be used only on instances of a specified class
- example: hasPet might have rdfs:range class Person

RDFS example: properties

```
<rdf:Property rdf:about="email">
  <rdfs:domain rdf:resource="#AcademicPerson" />
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
<rdf:Property rdf:about="homepage">
  <rdfs:domain rdf:resource="#AcademicPerson" />
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
<rdf:Property rdf:about="title">
  <rdfs:domain rdf:resource="#AcademicPerson" />
  <rdfs:range rdf:resource=" http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

Note that again we are using the abbreviated XML syntax – these are resources of rdf:type rdf:Property

RDFS & XML schema datatypes

The most recent version of RDF Schema allows us to be more specific in stating the range of resources, by using the primitive XML schema datatypes such as integer, string, etc

Example:

```
<rdf:Property rdf:about="email">
   <rdfs:domain rdf:resource="#AcademicPerson" />
   <rdfs:range
      rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
 </rdf:Property>
```

See the full "academic people" schema at
http://www.csd.abdn.ac.uk/teaching/levelfive/CS5008/examples/rdf/people.rdfs

RDF Schemas & instances

Once an RDF Schema has been defined, we can create instances of the classes in the schema, with the defined properties.

The instances must refer to the schema definitions, by using the fully-qualified URIs of the classes and properties.

This is done using namespaces wherever possible.

As usual in XML, the header of an RDF file containing instance data will typically have several namespaces.

Example:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:csd="http://www.csd.abdn.ac.uk/teaching/levelfive/
         CS5008/examples/rdf/people.rdfs#">
```

Here, the last namespace refers to our "academic people" schema.

Example instance data

```
<csd:Lecturer rdf:about="apreece">
  <csd:firstname>Alun</csd:firstname>
  <csd:surname>Preece</csd:surname>
  <csd:title>Dr</csd:title>
  <csd:email>apreece@csd.abdn.ac.uk</csd:email>
  <csd:telephone>01224 272291</csd:telephone>
  <csd:homepage>http://www.csd.abdn.ac.uk/~apreece</csd:homepage>
  <csd:office>230</csd:office>
</csd:Lecturer>
```

As before:

- the abbreviated syntax is used, so the rdf:type of this resource is implicitly csd:Lecturer
- the resource name – apreece – is relative to the document base:
  http://www.csd.abdn.ac.uk/teaching/levelfive/CS5008/examples/rdf/department.rdf
  so this resource has the fully-qualified URI
  http://www.csd.abdn.ac.uk/teaching/levelfive/CS5008/examples/rdf/department.rdf#apreece

RDF Schema versus XML Schema

It might appear that RDF Schema merely duplicates the functions of XML Schema. This is not the case!

XML Schema defines document structure.
Example: "the element book contains an element author".

RDF Schema defines a data model.
Example: "an instance of the class book has a property hasAuthor that relates the instance of book to an instance of the class person".

RDF Schemas are intended to be models of the world, whereas XML Schemas are intended to be models of XML documents.

Remember that the RDF data model is independent of XML – RDF can just as easily be written graphically, or as triples.

The real relationship between RDF/RDFS and XML Schema

The RDF XML syntax has an XML Schema, which describes the structure of an XML RDF document (but none of its meaning).
In natural language, the basic rules are:

- the outermost element is rdf:RDF
- the rdf:RDF element contains at least one rdf:Description element (the abbreviated syntax of course is more flexible)

In XML Schema syntax, this is basically:

```
<xsd:element name="RDF" type="rdfType"/>

<xsd:complexType name="rdfType">
  <xsd:sequence>
    <xsd:element name="Description" type="descriptionType"
      minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

RDF processing tools

The use of RDF is facilitated by the availability of a range of tools for processing it, including:

- Validators for RDF (W3C's online validator) and RDF Schema (FRODO RDFviz)
- Parsers (for Java, Python, Perl, Prolog, ...)
- Editors (Protégé, OntoEdit, Ontomat, ...)
- Storage & querying (Jena, Sesame, 3store, ...)
- Inference (Jena2, Triple, OntoBroker, ...)

Notably, Hewlett Packard's Jena library includes parsing, validation, storage and querying for RDF within Java.

Querying RDF

Because it has a formally-defined data model, RDF data can be queried. Several RDF query languages exist, the most popular of which is RDQL. The form of an RDQL query is similar to SQL:

SELECT vars
FROM documents
WHERE expressions
USING namespace declarations

Example – select all the (resource ID, email address) pairs from our "academic people":

```
SELECT ?x, ?y
FROM <http://www.csd.abdn.ac.uk/teaching/levelfive/CS5008/
examples/rdf/department.rdf>
WHERE ( ?x, <vcd:email>, ?y )
USING vcd for <http://www.csd.abdn.ac.uk/teaching/levelfive/
CS5008/examples/rdf/people.rdfs#>
```

Working with RDF & RDQL

RDF statements on the open Semantic Web can be parsed from the XML format into an RDF model, for example using Jena toolkit

Jena RDF models can be queried directly using the RDQL query language.

Example, "retrieve the phone number(s) of the person whose name is 'Alun Preece"':

```
SELECT ?y
WHERE ( ?x, <foaf:name>, "Alun Preece" )
AND ( ?x, <foaf:phone>, ?y > )
USING foaf FOR <http://xmlns.com/foaf/0.1/>
```

(RDF can also be queried in XML RDF syntax using University of Aberdeen's RDF Query-By-Example.)

Roadmap

Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

OK, but where do I start?

Most of us "got onto the Web" by writing a home page.

We used HTML to mark-up things like:

- who we are
- a photo
- things we like and do
- links to the home pages of people we know

Now we'll do the same on the Semantic Web.

Motivating example: social networks using **FOAF** (Friend-Of-A-Friend)

http://www.csd.abdn.ac.uk/foaf/

Creating a "Semantic Homepage"

Go to **http://www.csd.abdn.ac.uk/foaf**

Click on Add Your Profile.

Fill out the form; click the OK, Save My Details! Button.

You can browse the data in several ways:

- from your profile, click things you like/dislike to see others with the same…
- on the main page http://www.csd.abdn.ac.uk/foaf/
  - View people on UK map or
  - Browse people by interest or
  - Search by email address or postcode

So far, this is much less than you can get at any **social networking** website.

To see the point, go to your profile page, and click the RDF button:


```
<foaf:Person rdf:nodeID="bNode5">
 <foaf:name>Alun Preece</foaf:name>
 <foaf:firstName>Alun</foaf:firstName>
 <foaf:surname>Preece</foaf:surname>
 <foaf:homepage>http:/www.csd.abdn.ac.uk/~apreece</foaf:homepage>
 <foaf:mbox_sha1sum>1d7e49a9d75b5a9354fc5c22af344848103fcfa5
   </foaf:mbox_sha1sum>
 <foaf:schoolHomepage>http:/www.csd.abdn.ac.uk/</foaf:schoolHomepage>
 <wil:has rdf:resource="http://dmoz.org//Recreation/Pets/Cats/" />
 <wil:has rdf:resource="http://www.whatilike.org/gadget/digital_camera" />
 <wil:likes
   rdf:resource="http://www.dmoz.org/Arts/Literature/Genres/Science_Fiction/" />
 <wil:likes rdf:resource="http://www.dmoz.org/Arts/Literature/Short_Stories/" />
 <wil:likes rdf:resource="http://www.dmoz.org/Arts/Movies/Genres/Comedy/" />
 <wil:likes rdf:resource="http://www.dmoz.org/Arts/Movies/Genres/Cult_Movies/" />
 <wil:likes rdf:resource="http://www.london-eating.co.uk/cuisines/pizza.asp" />
 <wil:likes rdf:resource="http://www.london-eating.co.uk/cuisines/polish.asp" />
 <wil:likes rdf:resource="http://www.whatilike.org/travel/train" />
 <foaf:depiction
   rdf:resource="http://www.csd.abdn.ac.uk/~apreece/images/apreece.jpg" />
</foaf:Person>
```

```
Person
  name  Alun Preece
  firstName  Alun
  surname  Preece
  homepage  http:/www.csd.abdn.ac.uk/~apreece
  mbox_sha1sum  1d7e49a9d75b5a9354fc5c22af344848103fcfa5

  schoolHomepage  http:/www.csd.abdn.ac.uk
has                                  Pets/Cats
has                                  gadget/digital_camera
likes
                                     Literature/Genres/Science_Fiction
likes                                Literature/Short_Stories
likes                                Movies/Genres/Comedy
likes                                Movies/Genres/Cult_Movies
likes                                cuisines/pizza
likes                                cuisines/polish
likes                                travel/train
  depiction
          http://www.csd.abdn.ac.uk/~apreece/images/apreece.jpg
```

What does the code mean?

This code is RDF written in XML

Tags are defined in RDF Schema "vocabularies"

- tags beginning **foaf:** are from the **FOAF** vocab, defined at http://xmlns.com/foaf/0.1/
- tags beginning **wil:** are from the **WhatILike** vocab, defined at http://whatilike.org/ontology

The FOAF vocabulary covers

- entities: people, organisations, projects, documents
- "identifying" details: mbox, homepage, phone, depiction
- relationships between people: who knows who

FOAF vocab excerpt

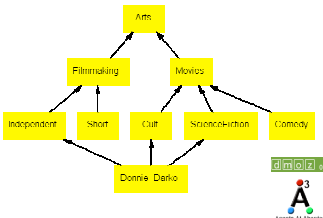
```
<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Person" />

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/name">
  <rdfs:range rdf:resource=
    "http://www.w3.org/2000/01/rdf-schema#Literal" />
</rdf:Property>

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/mbox" />

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/phone" />

<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/knows">
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person" />
  <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Person" />
</rdf:Property>
```

Vocab reuse

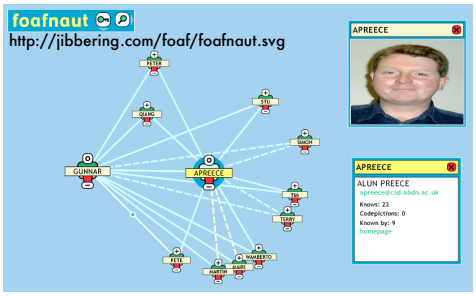
Where possible, URIs are used to name things, rather than plain text:

- literature & movie categories from **dmoz.org**
- cuisine types from **london-eating.co.uk**

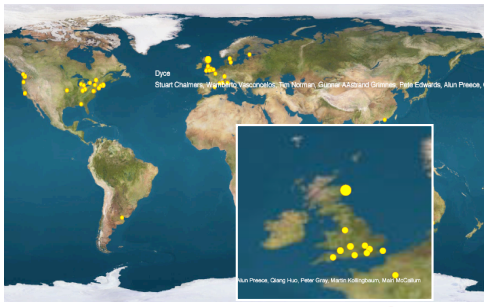
This is to promote standardised descriptions (and thus exact matches).



Visualising FOAF data



World Wide FOAF



<CoffeeBreak>

</CoffeeBreak>

Roadmap

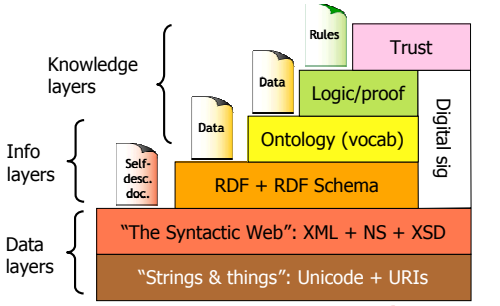
Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

Semantic Web architecture



[Adapted from Semantic Web "layer cake" slide due to Tim Berners-Lee]

Data layers: Unicode, URIs, XML

Like the "traditional" Web (in recent years), Semantic Web data is based on W3C-recommended standards:

Unicode for strings (in all languages)

URIs - Uniform Resource Identifiers - to name "things"

XML as the standard extensible markup language

XML Schema for a variety of primitive datatypes (integer, real number, string, date, URI, ...)

XML namespaces to give global scope for tag names

```
<rdf:RDF xmlns:vc="http://www.w3.org/2001/vcard-rdf/3.0#">
  <rdf:Description rdf:about="http://www.csd.abdn.ac.uk/~apreece">
    <vc:FN>Alun Preece</vc:FN>
    <vc:EMAIL>apreece@csd.abdn.ac.uk</vc:EMAIL> ...
```

Information layer: RDF + RDFS

RDF - Resource Description Framework - is the foundation of the Semantic Web standards

RDF provides:

- a simple semantic data model with classes (entities) & properties (relationships)
- schema definition constructs (RDF Schema) to define simple vocabularies of terms
- an XML syntax for marking-up RDF data

RDF is the best-developed aspect of the Semantic Web:

- many RDFS vocabularies are currently available, including
  - Dublin Core
  - RSS (newsfeeds)
  - FOAF
- a suite of software tools exists to process RDF
- A lot can often be done with small amounts of semantic markup!

The need for ontologies

RDF(S) is designed to be simple.

To define more sophisticated vocabulary, we need to go one layer higher: to the ontology layer.

The Semantic Web ontology language, OWL, extends RDF with some additional functionality.

Concrete examples:

- a Person must have at least one name
- a Person must have exactly one age
- the class Person is the disjoint union of the classes Man and Woman
- A personal email address (mbox) belongs to exactly one Person

This last example is crucial to FOAF...

Roadmap

Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

OWL: Web Ontology Language

Ontology: "explicit specification of a (shared) conceptualisation".

Core of the World Wide Web Consortium's Semantic Web activity in recent years.

In various senses a successor to previous work on "Web-friendly" knowledge modelling languages:

- RDF & RDF Schema
- DAML-ONT
- OIL / DAML+OIL

W3C's Web Ontology Working Group are a "who's who" of the knowledge representation field (a mixed blessing?).

Now a W3C recommendation.

Current activity: "best practices" working group...

XML, RDF & OWL

XML: universal syntax

XML Schema: defines structure of XML docs

RDF: datamodel for resource objects

RDF Schema: basic vocabulary for defining RDF classes & properties, and hierarchies of each

OWL: extended vocab for defining classes & properties, including

- cardinality (e.g. minCardinality 1)
- equality (e.g. equivalentClass)
- relationships between classes (e.g. disjointWith)
- characteristics of properties (e.g. FunctionalProperty)

OWL: why?

Semantic Web apps:

- portal Websites & intranets (information architecture)
- multimedia digital libraries (rich metadata)
- agents & Web services (interoperability, automation)
- design documentation (complex, interlinked)

Capabilities:

- ontology sharing, evolution, interoperability
- inconsistency detection
- expressivity vs scalability
- standards compliance

OWL "species"

OWL Lite

- "RDF-and-a-half"
- Mainly intended for class hierarchies & simple constraints (cardinality 0 or 1, equality, ...)

OWL DL (contains OWL Lite)

- Description Logic theoretical properties
- Intended where completeness & decidability are an issue

OWL Full (contains OWL DL)

- Max expressivity; no computational guarantees
- Supports "Web-scale" & "Web-style" KR&R

OWL sublanguages cont'd

Every legal OWL Lite ontology is a legal OWL DL ontology.
Every legal OWL DL ontology is a legal OWL Full ontology.
Every valid OWL Lite conclusion is a valid OWL DL conclusion.
Every valid OWL DL conclusion is a valid OWL Full conclusion.

The converse in each case does not hold.

OWL Lite: essentials

Schema constructs
Class (i.e. owl:Class)
rdf:Property
rdfs:subClassOf
rdfs:subPropertyOf
rdfs:domain
rdfs:range
Individual

Property characteristics
inverseOf
TransitiveProperty
FunctionalProperty
InverseFunctionalProperty
SymmetricProperty

Equality constructs
equivalentClass
equivalentProperty
sameIndividualAs
differentFrom
allDifferent

Cardinality
minCardinality (0 or 1)
maxCardinality (0 or 1)
Cardinality (0 or 1)

Class intersection
intersectionOf

Headers
imports
priorVersion
backwardCompatibleWith
incompatibleWith

Property type restrictions
allValuesFrom
someValuesFrom

RDF datatyping

OWL DL & OWL Full: essentials

Class axioms
oneOf
disjointWith

Class expressions
equivalentClass
rdfs:subClassOf
unionOf
intersectionOf
complementOf

Property fillers
hasValue

Arbitrary cardinality
minCardinality
maxCardinality
Cardinality

When is a Class not a Class?

Answer: in OWL Lite & OWL DL, when it's an Individual - description logics do not permit Classes to be treated as Individuals.

So, no "Class, an Individual class, being the Class of all Classes" (as in RDF).

So, rdfs:Class cannot be used in OWL Lite or OWL DL.

owl:Class is defined as rdfs:subClassOf rdfs:Class.

(But, in OWL Full, they coincide!)

Note that this means an RDF-processing agent can still use a lot of OWL, because it understands the triple:

owl:Class rdfs:subClassOf rdfs:Class

Defining an owl:Class (I)

By class identifier: Lite/DL/Full

```
<owl:Class rdf:about="Lecturer">
  <rdfs:subClassOf rdf:resource="#Person" />
</owl:Class>
```

By enumeration: DL/Full

```
<owl:Class rdf:about="ComputingOfficer">
  <owl:oneOf rdf:parseType="Collection">
    <Academic rdf:about="#nmurray" />
    <Academic rdf:about="#jmartin" />
    <Academic rdf:about="#mritchie" />
  </owl:oneOf>
</owl:Class>
```

Defining an owl:Class (II)

By property restriction: Lite*/DL/Full

```
<owl:Class rdf:about="Researcher">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#activity" />
      <owl:someValuesFrom rdf:resource="#ResearchArea" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

By intersection/union/complement: DL/Full

```
<owl:Class rdf:about="UniversityStaff">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Lecturer" />
    <owl:Class rdf:about="#Researcher" />
    <owl:Class rdf:about="#ComputingOfficer" />
  </owl:unionOf>
</owl:Class>
```

Properties in OWL

Two types

- ObjectProperty - relations between instances of classes
- DatatypeProperty - relates an instance to an rdfs:Literal or XML Schema datatype

(Both rdfs:subClassOf rdf:Property)

```
<owl:DatatypeProperty rdf:about="name">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema/string" />
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:about="activity">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="ActivityArea" />
</owl: ObjectProperty>
```

Individual axioms ("facts")

OWL is not only a language for defining ontologies - it is used to define their instances (Individuals)

Example:

```
<Lecturer rdf:about="apreece">
  <name>Alun Preece</name>
  <activity rdf:resource="#AgentsResearch" />
  <activity rdf:resource="#WebTeaching" />
</Lecturer>
<ResearchArea rdf:about="AgentsResearch"/>
<TeachingArea rdf:about="WebTeaching"/>
```

(Notice how individual **apreece** follows the definition of Researcher given earlier)

A little ontology goes a long way

FOAF uses OWL to define the mbox property:

```
<rdf:Property rdf:about="http://xmlns.com/foaf/0.1/mbox">
  <rdf:type rdf:resource=
        "http://www.w3.org/2002/07/owl#
        InverseFunctionalProperty" />
</rdf:Property>
```

This definition means: "mbox is a personal mailbox, i.e. an Internet mailbox associated with exactly one owner".

This means, in database terms, the value of mbox acts as a primary key for Persons in the FOAF world - a unique ID.

Ontology mapping

OWL can also be used to map one vocabulary to another.

Example: we might wish to say that the vCard EMAIL property is the same as FOAF's mbox*:

```
<rdf:Property rdf:about=
    "http://www.w3.org/2001/vcard-rdf/3.0#EMAIL">
  <owl:equivalentProperty
    rdf:resource="http://xmlns.com/foaf/0.1/mbox" />
</rdf:Property>
```

An OWL reasoner could use this equivalence to derive a value for some resource's vc:EMAIL if it can find a value for foaf:mbox.

*but we'd be wrong...

OWL: implications for the KBS field

OWL is potentially the most important knowledge representation language we've yet seen.

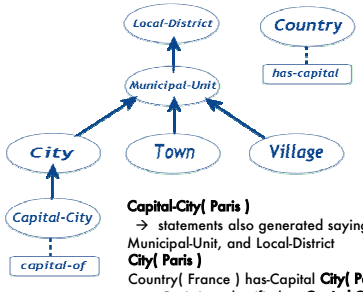
(Jim Hendler claims OWL's predecessor DAML already was, in terms of numbers of statements asserted.)

Could OWL be the "last word" in KR similar to how HTML came to dominate the field of hypertext markup?

Implications:

- If you're doing KR research, you will need to situate yourself in relation to OWL?
- If you're building KBS, OWL will be your first choice of KRL?
- But there are enormous challenges ahead in creating effective OWL reasoners/processors...

Ontological reasoning



Capital-City(Paris)
→ statements also generated saying Paris is a City, Municipal-Unit, and Local-District
City(Paris)
Country(France) has-Capital **City(Paris)**
→ Paris is reclassified as **Capital-City(Paris)**

Semantic Web Rule Language (SWRL)

Allows set of Horn-clause rules to be expressed against RDF/OWL data models:

"If a person has an affiliation with a University and that University has a postal address of a city, then this implies that the person has a base location of the same city where the University is located".

Using informal syntax and adding explicit universal quantifiers, this can be written as:

```
(∀?p,?u,?c) Person(?p) ∧ University(?u) ∧
        has-affiliation(?p,?u) ∧
        has-postal-address(?u,?c) ∧ City(?c)
              ➡ has-base-locn(?p,?c)
```

SWRL in RDF (!)

```
<swrl:Imp>
 <swrl:body rdf:parseType="Collection">
  <swrl:ClassAtom>
   <swrl:classPredicate rdf:resource="&akt;#Person"/>
   <swrl:argument1 rdf:resource="#p"/>
  </swrl:ClassAtom>
  <swrl:ClassAtom>
   <swrl:classPredicate rdf:resource="&akt;#Organization"/>
   <swrl:argument1 rdf:resource="#u"/>
  </swrl:ClassAtom>
  <swrl:IndividualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&akt;#has-affiliation"/>
   <swrl:argument1 rdf:resource="#p"/>
   <swrl:argument2 rdf:resource="#u"/>
  </swrl:IndividualPropertyAtom>
  <swrl:IndividualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&akt;#has-postal-address"/>
   <swrl:argument1 rdf:resource="#u"/>
   <swrl:argument2 rdf:resource="#a"/>
  </swrl:IndividualPropertyAtom>
  <swrl:IndividualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&akt;#address-city"/>
   <swrl:argument1 rdf:resource="#a"/>
   <swrl:argument2 rdf:resource="#c"/>
  </swrl:IndividualPropertyAtom>
 </swrl:body>
 <swrl:head rdf:parseType="Collection">
  <swrl:IndividualPropertyAtom>
   <swrl:propertyPredicate rdf:resource="&akt;#has-base-location"/>
   <swrl:argument1 rdf:resource="#p"/>
   <swrl:argument2 rdf:resource="#c"/>
  </swrl:IndividualPropertyAtom>
 </swrl:head>
</swrl:Imp>
```

Semantic Web Constraints (CIF/SWRL)

Main motivation: explicit quantification

"Any workgroup containing at least five members must contain at least two individuals from different sites (base locations)."

```
(∀ ?g ∈ Workgroup)
  has-size(?g,?s) ∧ greaterThanOrEqual(?s,5) ➡
  (∃ ?p1,?p2 ∈ Person)
    has-member(?g,?p1) ∧ has-base-locn(?p1,?b1) ∧
    has-member(?g,?p2) ∧ has-base-locn(?p2,?b2) ∧
    notEqual(?b1,?b2)
```

Roadmap

Part 1 "Untangling the Semantic ***Web***"

- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the ***Semantic*** Web"

- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

Semantic Web applications desiderata

1. Data and/or program nodes should be distributed. If the application is only executed on, and uses data from, a single node, a relational database is likely to be quicker to implement as well as more efficient
2. Following 1, the distributed data will have diverse ownership and will therefore be heterogeneous; properties used and level of detail will vary; different ontologies may be in use, even for a single domain.
3. The heterogeneity of the data should be described in a machine readable format using an ontology language like RDFS or OWL. Different ontologies may be tied together by relating them to common ontologies like Dublin Core or FOAF, or to simple taxonomies like WordNet.
4. Real world data should be included. It is too easy to use artificial toy examples, which will not illustrate the problems that are raised by a real distributed, heterogeneous application.

CS AKTive Space

AKT project: Aberdeen, Edinburgh, Sheffield, Southampton, Open University - see http://www.aktors.org/technologies/

[Slides clipped from "CS AKTiveSpace: Building a Semantic Web Application", presented by High Glaser (Soton) @ ESWC 2004]

Context (3-4 years ago - still holds now):

- Knowledge Life-cycle - all aspects (including maintenance, scale)
- Tools & theories development
- Not much RDF out there
- Scalable tools
- Technology intercepts
- Emergent applications
- Interdisciplinary - need very wide buy in

-ologies

Technologies

- Component details not unique to us
- Scruffy
- Technology Integration Experiment
- Harvesting using DOME, Perl, sed, ...

Ontologies

- Developed own!
  - good exercise
  - our domain
- Then adopted
  - AKT Reference Ontology
- Any sufficiently expressive
- But performance (fit for purpose)

Harvesting

Wherever appropriate

- No cooperation (expected)
- Effort/value balance
- Some UK CS departments
- Funding agencies
- Geographical (ISO 3166)
- General web

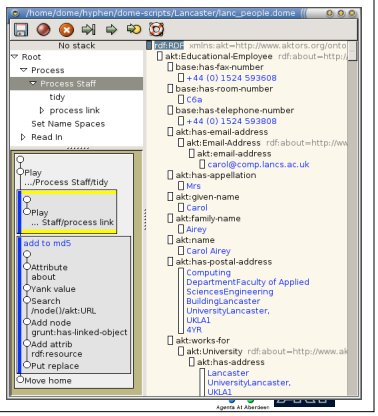
Where to put it?

- 430MB, 10 million triples, 800,000 instances
- 3store (cache!)
  - RDQL or OKBC
  - Linearish performance (10 => 15 => 25M triples)
  - Knows source (provenance)

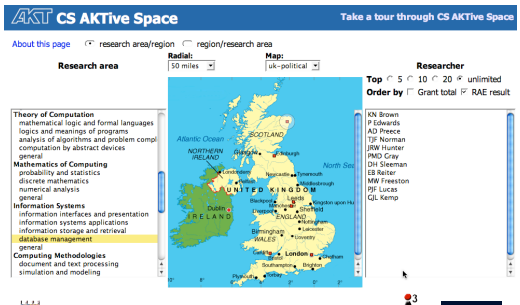
Push and Pull

- Who will push - please?

Dome



CAS UI



SemanticOrganizer

- http://ic.arc.nasa.gov/sciencedesk/
- Investigators
  - Dr. Richard Keller, NASA Ames
    - (keller@email.arc.nasa.gov)
  - Dr. Dan Berrios, NASA Ames
    - (berrios@email.arc.nasa.gov)

Roadmap

Part 1 "Untangling the Semantic *Web*"
- Semantic Web origins: Metadata & RDF
- RDF Schema & data modelling
- Creating a homepage on the Semantic Web

Part 2 "Untangling the *Semantic* Web"
- The Semantic Web architecture stack
- Ontologies, OWL, & reasoning
- Illustrative Semantic Web applications

Wrap-up (1)

At its foundations, the Semantic Web provides for the creation of standard, open information schemas - founded on a formal semantics.

New schemas can extend existing ones, freely but in a prncipled way.

Semantic Web resources can refer to each other, and to "traditional" Web resources.

Information and applications are kept separate - deployers of information on the SW can't (and don't need to) anticipate how the info will be used.

RDF processing software is widely available; OWL is catching-up.

Compelling applications are already there…

Wrap-up (2)

The Semantic Web is exciting from several perspectives:
- as a piece of computing science / AI technology
- as a "new generation" for the Web
- as a platform for diverse kinds of applications

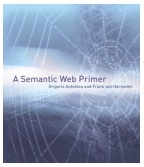
It's still the Web we know and love:
- it co-exists with all our messy HTML, etc data
- it's a global system: URIs are universal!
- it's extremely open
- it's not too hard to get started…

IT'S WORTH GETTING INVOLVED!

Semantic Web primers

http://www.w3.org/2001/sw/

Practical RDF, O'Reilly (2003)



Semantic Web Primer, MIT Press (2004)