Short introduction to the Semantic Web

$Date: 2006/11/25 13:37:30$

Ivan Herman, W3C
- The current Web represents information using
  - natural language (English, Hungarian, Chinese, …)
  - graphics, multimedia, page layout structure
  - etc
- Humans can process this easily
  - can deduce facts from partial information
  - can create mental associations
  - are used to various sensory information
    - (well, sort of… people with disabilities may have serious problems on the Web with rich media!)
Tasks often require to *combine* data on the Web:
- *hotel and travel information may come from different sites*
- *searches in different digital libraries*
- *etc.*

Again, humans combine these information easily
- *even if different terminologies are used!*
However: machines are ignorant!

- *partial information is unusable*
- *difficult to make sense from, e.g., an image*
- *drawing analogies automatically is difficult*
- *difficult to combine information automatically*
  - is `<foo:creator>` same as `<bar:author>`?
  - how to combine different XML hierarchies?

- ...
The best-known example...

- *Google et al. are great, but there are too many false or missing hits*
  - e.g., if you search in for “yacht racing”, the America’s Cup will *not* be found
- *adding (maybe application specific) descriptions to resources should improve this*
Example: Automatic Airline Reservation

- Your automatic airline reservation
  - *knows about your preferences*
  - *builds up knowledge base using your past*
  - *can combine the local knowledge with remote services:*
    - airline preferences
    - dietary requirements
    - calendaring
    - etc

- It communicates with *remote* information (i.e., on the Web!)
- (M. Dertouzos: The Unfinished Revolution)
Example: Data(base) Integration

- Databases are very different in structure, in content
- Lots of applications require managing *several* databases
  - *after company mergers*
  - *combination of administrative data for e-Government*
  - *biochemical, genetic, pharmaceutical research*
  - *etc.*
- Most of these data are accessible from the Web (though not necessarily public yet)
Example: data integration in life sciences
And the problem *is* real
Example: Digital Libraries

- It is a bit like the search example
- It means catalogs on the Web
  - *librarians have known how to do that for centuries*
  - *goal is to have this on the Web, World-wide*
  - *extend it to multimedia data, too*
- But it is more: software agents should also be librarians!
  - *help you in finding the right publications*
Web services technology is great
But if services are ubiquitous, searching issue comes up, for example:
- “find me the best differential equation solver”
- “check if it can be combined with the XYZ plotter service”

It is necessary to characterize the service
- not only in terms of input and output parameters…
- …but also in terms of its semantics
(Some) data should be available for machines for further processing
Data should be possibly combined, merged on a Web scale
Sometimes, data may describe other data (like the library example, using metadata)…
… but sometimes the data is to be exchanged by itself, like my calendar or my travel preferences
Machines may also need to *reason* about that data
To make data machine processable, we need:

- unambiguous names for resources (that may also bind data to real world objects): URI-s
- a common data model to interchange, connect, describe the resources: RDF
- access to that data: SPARQL
- define common vocabularies: RDFS, OWL, SKOS
- reasoning logics: OWL, Rules

The “Semantic Web” is an extension of the current Web, providing an infrastructure for the integration of data on the Web.
We said “connecting” data…
But a simple connection is not enough… it should be named somehow
- a connection from “me” to my calendar is not the same as the connection from “me” to my CV (even if all of these are on the Web)
- the first connection should somehow say “myCalendar”, the second “myCV”
Hence the RDF Triples: a labelled connection between two resources
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
- conceptually: "p" connects, or relates the "s" and "o"
- note that we use URI-s for naming: i.e., we can use \(http://www.example.org/myCalendar\)
- here is the complete triple:

\[
(http://www.ivan-herman.net, http://…/myCalendar, http://…/calendar)
\]

**RDF**

is a general model for such triples (with machine readable formats like RDF/XML, Turtle, n3, RXR, …)

... *and that's it!* (simple, isn't it? 😊)
RDF Triples are also referred to as “triplets”, or “statement”.

The s, p, o resources are also referred to as “subject”, “predicate”, ”object”, or “subject”, ”property”, ”object”.

Resources can use any URI; i.e., it can denote an element within an XML file on the Web, not only a “full” resource, e.g.:

- http://www.example.org/file.xml#xpointer(id('calendar'))
- http://www.example.org/file.html#calendar
A Simple RDF Example

```xml
<rdf:Description rdf:about="http://www.ivan-herman.net">
  <foaf:name>Ivan</foaf:name>
  <abc:myCalendar rdf:resource="http://.../myCalendar"/>
  <foaf:surname>Herman</foaf:surname>
</rdf:Description>
```
Anybody can create (meta)data on any resource on the Web
- e.g., the same SVG or XHTML file could be annotated through other terms
- semantics is added to existing Web resources via URI-s
- URI-s make it possible to link (via properties) data with one another

URI-s ground RDF into the Web
- information can be retrieved using existing tools
- this makes the “Semantic Web”, well… “Semantic Web”
It becomes easy to *merge* data
- e.g., applications may merge annotations

Merge can be done because statements refer to the *same* URI-s
- nodes with identical URI-s are considered identical

Merging is a *very* powerful feature of RDF
- data linkage, metadata, etc, may be defined by several (independent) parties…
- …and combined by an application
- one of the areas where RDF is much handier than pure XML in many applications
What Merge Can Do...
Need for a Query Language

- Each data model needs its own “query language” to access large amount of data
  - relational databases have SQL, XML has XQuery…
- SPARQL is the query language for RDF
  - queries are expressed in forms of RDF triples with unknown variables
  - the query returns a list possible resources (i.e., URI-s or literal values) or full set of triples (depending on the query type)
- SPARQL is emerging as the primary way to access RDF data
How to Get to RDF Data?

- The simplest approach: write your own RDF data in your preferred syntax
- Using URI-s in RDF binds you automatically to the real resources
- You may add RDF to XML directly (in its own namespace)
  - *e.g., in SVG:*

```xml
<svg ...
 ...  <metadata>
     <rdf:RDF xmlns:rdf="http://../rdf-syntax-ns#">
    ...  
     </rdf:RDF>
</metadata>
 ... 
</svg>
```

- Works in some cases, but not satisfactory for a real deployment!
RDF Can Also Be Extracted/Generated

- Use intelligent “scrapers” or “wrappers” to extract a structure (hence RDF) from a Web page…
  - using conventions in, e.g., class names or header conventions like `meta` elements
- … and then generate RDF automatically (e.g., via an XSLT script)
- This is what the “microformats” are doing
  - they may not extract RDF but use the data directly instead, but that depends on the application
  - other applications may extract it to yield RDF (e.g., RSS1.0)
Most of the data are stored in relational databases

“RDFying” them is an impossible task

“Bridges” are being defined:

- a layer between RDF and the database
- RDB tables are “mapped” to RDF graphs on the fly
- in some cases the mapping is generic (columns represent properties, etc…)
- … in other cases separate mapping files define the details

This is a very important source of RDF data
SPARQL As a Unifying Force
Creating data and using it from a program works, provided the program *knows* what terms to use!

We used terms like:

- `foaf:name`, `abc:myCalendar`, `foaf:surname`, ...
- `etc`

Are they all known? Are they all correct? (it is a bit like defining record types for a database)
Possible Issues to Handle

- What are the possible terms?
  - “is the set of data terms known to the program?”

- Are the properties used correctly?
  - “do they make sense for the resources?”

- Can a program reason about some terms? Eg:
  - “if «A» is left of «B» and «B» is left of «C», is «A» left of «C»?”
  - obviously true for humans, not obvious for a program …
  - … programs should be able to deduce such statements

- If somebody else defines a set of terms: are they the same?
  - clearly an issue in an international context
The Semantic Web needs a support of *ontologies*:

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

We need a *Web Ontologies Language* to define:

- the terminology used in a specific context
- possible constraints on properties
- the logical characteristics of properties
- the equivalence of terms across ontologies
- etc

This is done by RDFS (RDF Schemas) and OWL (Web Ontology Language)
Think of well known in traditional ontologies:
- *use the term “mammal”*
- “every dolphin is a mammal”
- “Flipper is a dolphin”
- 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”)
  - “mammal”, “dolphin”, …
Relationships are defined among classes/resources:

- "typing": an individual belongs to a specific class ("Flipper is a dolphin")
- "subclassing": instance of one is also the instance of the other ("every dolphin is a mammal")

RDFS formalizes these notions in RDF
RDFS defines `rdfs:Resource`, `rdfs:Class` as nodes; `rdf:type`, `rdfs:subClassOf` as properties

* (these are all special URI-s, we just use the namespace abbreviation)
Ivan Herman, W3C

Inferred Properties

- `(Flipper rdf:type Mammal)`
- is *not* in the original RDF data…
- …but can be *inferred* from the RDFS rules
- Better RDF environments return that triplet, too
RDFS and OWL

- RDFS defines the basic principles
- OWL adds more complicated features to RDFS like:
  - constructions of classes using existing ones
  - characterize relationships (e.g., whether they are transitive, symmetric, functional, etc)
Essentially, like a set-theoretical union:
Ontologies may be extremely large:
- their management requires special care
- they may consist of several modules
- come from different places and must be integrated

Ontologies are on the Web. That means:
- applications may use several, different ontologies, or…
- … same ontologies but in different languages
- equivalence of, and relations among terms become an issue

OWL includes possibilities for class/property equivalence, version and deprecation control, etc.
Example: Connecting to Hungarian
However: Ontologies are Hard!

- Hard to implement a full ontology management system
  - *may be superfluous for some applications*
- Hence the “onion” model of increasingly complex specs:
  - *no property expressions or datatypes in RDF Schemas*
  - *not all set operators, restricted cardinality in OWL Lite*
  - *some restrictions, but a computational guarantee in OWL DL*
  - *full expressive power in OWL Full (but no computational guarantee)*
Ontologies are Hard! (cont)

- “Lite” < “DL” < “Full”, but not completely true for RDFS
  - RDFS is “almost” a subcategory
  - not all RDFS statements are valid in DL…
  - …but they are for Full

- Applications may take what they really need!
The Work is Not Over

Rules
  more general logical rules to the Semantic Web infrastructure; also includes the *interchange* of rules among rule based systems

Evolution of the RDF model
  e.g., add time information, probabilities, “measure of fuzziness” to statements (still in research phase)

Evolution of OWL
  additional features, new (eg, even lighter) layers

Trust
  a trust infrastructure for SW (for example: “can I trust the author of this set of assertions?”); on the future stack of W3C…

…
Lots of Tools

- **(Graphical) Editors**
  - IsaViz (Xerox Research/W3C/Inria), RDFAuthor (Univ. of Bristol), Protege 2000 (Stanford Univ.), SWOOP (Univ. of Maryland), Orient (IBM)

- **Programming Environments**
  - Jena (for Java, includes OWL reasoning and SPARQL queries), RDFLib (for Python), Redland (in C, with interfaces to Tcl, Java, PHP, Perl, Python, … and with SPARQL queries), SWI-Prolog, IBM’s Semantic Web Toolkit, …

- **Databases**
  - either based on an internal sql engine or fully triple based
  - Kowari, Gateway, 3Store, Jena’s Joseki, Oracle’s Database 10g, …

- **RDF and OWL validators and reasoners**
  - W3C’s RDF Validator, BBN OWL Validator, Pellet OWL Reasoner, …

- **RDB→RDF layers, converters**
  - D2R Server, SquirrelRDF, SPASQL, R2O, …
Applications patterns emerge

Major companies offer (or will offer) Semantic Web tools or systems using Semantic Web: Adobe, Oracle, IBM, HP, Software AG, webMethods, Northrop Gruman, Altova, …

Some of the names of active participants in W3C SW related groups: ILOG, HP, Agfa, SRI International, Fair Isaac Corp., Oracle, Boeing, IBM, Chevron, Siemens, Nokia, Merck, Pfizer, AstraZeneca, Sun, Citigroup, …

“Corporate Semantic Web” listed as major technology by Gartner

Various application patterns emerge
  * often pioneered by specific communities, eg, life sciences, eGovernment, energy industry, …
Applications are not always very complex...

- Eg: simple semantic annotations of patients’ data greatly enhances communications among doctors
- What is needed: some simple ontologies, an RDFa/microformat type editing environment
- Simple but powerful!
Data integration comes to the fore as one of the SW Application areas.

Very important for large application areas (life sciences, energy sector, eGovernment, financial institutions), as well as everyday applications (e.g., reconciliation of calendar data).

Life sciences example:
- *data in different labs…*
- *data aimed at scientists, managers, clinical trial participants…*
- *large scale public ontologies (genes, proteins, antibodies, …)*
- *different formats (databases, spreadsheets, XML data, XHTML pages)*
- *etc*
Life Sciences (cont.)

R&D Scientist
manipulating the data manually.

LIMS  Bioinformatics  Cheminformatics  Public Data Source

LIMS  Bioinformatics  Cheminformatics  Public Data Source

Chromatin, Roles (OWL, RDFS...)
Query (SPARQL, ...)
Data (LGD,RDF,..)

Ivan Herman, W3C
General approach

1. Map the various data onto RDF
   - “mapping” may mean on-the-fly SPARQL to SQL conversion, “scraping”, etc
2. Merge the resulting RDF graphs (with a possible help of ontologies, rules, etc, to combine the terms)
3. Start making queries on the whole!

- Remember the role of SPARQL?
Example: antibodies demo

- Scenario: find the known antibodies for a protein in a specific species
- Combine (“scrape”) three different data sources
- Use SPARQL as an integration tool (see also demo online)
Portals

- Vodafone's Live Mobile Portal
  - search application (e.g. ringtone, game, picture) using RDF
    - page views per download decreased 50%
    - ringtone up 20% in 2 months

- A number of other portal examples: Sun’s White Paper Collections and System Handbook collections; Nokia’s S60 support portal; Harper’s Online magazine linking items via an internal ontology; Oracle’s virtual press room; Opera’s community site, Yahoo! Food,…
Improved search on top of pubmed.org
- search results are ranked using the specialized ontologies
- extra search terms are generated and terms are highlighted

Importance of *domain specific ontologies* for search improvement
Adobe's XMP

- Adobe’s tool to add RDF-based metadata to *most* of their file formats
  - *used for more effective organization*
  - *supported in Adobe Creative Suite*
  - *support from 30+ major asset management vendors, with separate XMP conferences*

- The tool is available for all!
Thank you for your attention!

These slides are publicly available on:

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

in XHTML and PDF formats; the XHTML version has active links that you can follow