The Semantic Web

José Luis Ambite

(based on slides by Yolanda Gil, Ian Horrocks, Craig Knoblock, and Tom Russ)

The Semantic Web

W3C's Tim Berners-Lee: “Weaving the Web”:
“I have a dream for the Web... and it has two parts.”

- The first Web enables communication between people
  - The Web shows how computers and networks enable the information space while getting out of the way
- The new Web will bring computers into the action
  - Step 1 -- Describe: putting data on the Web in machine-understandable form -- a Semantic Web
    • RDF (based on XML)
    • Master list of terms used in a document (RDF schema)
    • Each document mixes global standards and local agreed-upon terms (namespaces)
  - Step 2 -- Infer and reason: apply logic inference
    • Operate on partial understanding
    • Answering why
    • Heuristics

Web Semantics

Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Unicode

 Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Unicode

- A character encoding system, like ASCII, designed to help developers who want to create software applications that work in any language in the world
- Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language

URI

Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)
URIs: Uniform Resource Identifiers (aka URLs)

- The Web is an information space. URIs are the points in that space.
- Short strings that identify resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources.
- They make resources addressable in the same simple way. They reduce the tedium of "log in to this server, then issue this magic command ..." down to a single click.

XML and Namespaces

Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Why XML (eXtensible Markup Language)

Problems with HTML

- HTML design:
  - HTML is intended for presentation of information as Web pages.
  - HTML contains a fixed set of markup tags.
- This design is not appropriate for data:
  - Tags don't convey meaning of the data inside the tags.
  - Tags are not extensible.

The Design of XML

- Tags can be used to represent the meaning of data/information – separates syntax (structural representation) from semantics => only syntax is considered in XML
- There is no fixed set of markup tags - new tags can be defined
- Underlying data model is a tree structure
- "XML is the new ASCII" -- Tim Bray

http://www.w3.org/TR/2000/REC-xml-20001006

Simple XML Example

- <Bookstore>
  - <Book ID="101">
    - <Author>John Doe</Author>
    - <Title>Introduction to XML</Title>
    - <Date>12 June 2001</Date>
    - <Publisher>XYZ</Publisher>
  - <Book ID="102">
    - <Author>Foo Bar</Author>
    - <Title>Introduction to XSL</Title>
    - <Date>12 June 2001</Date>
    - <Publisher>ABC</Publisher>
  - </Bookstore>

XML by itself is just hierarchically structured text

An important diversion: Namespaces

- What is a Namespace?
  - The Namespace of an element, is the scope within which, it (and thus it's name) is valid
- Why do we need Namespaces?
  - If elements were defined within a global scope, it becomes a problem when combining elements from multiple documents
  - Modularity: If a markup vocabulary exists which is well understood and for which there is useful software available, it is better to reuse it
- Namespaces in XML:
  - An XML namespace is a collection of names, identified by a URI reference. Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a prefix and a local part. The prefix, which is mapped to a URI reference, selects a namespace

An important diversion: Namespaces

- What is a Namespace?
  - The Namespace of an element, is the scope within which, it (and thus it’s name) is valid
- Why do we need Namespaces?
  - If elements were defined within a global scope, it becomes a problem when combining elements from multiple documents
  - Modularity: If a markup vocabulary exists which is well understood and for which there is useful software available, it is better to reuse it
- Namespaces in XML:
  - An XML namespace is a collection of names, identified by a URI reference. Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a prefix and a local part. The prefix, which is mapped to a URI reference, selects a namespace
XSD: XML Schema Definition

- Written in the same syntax as XML documents (unlike XML DTDs!)
- Elements and attributes
  - Wide range of primitive data types, supporting those found in databases (string, boolean, decimal, integer, date, etc.)
  - Can create your own datatypes (complexType)
- Can derive new type definitions on the basis of old ones (refinement)
- Can have constraints on attributes
  - Examples: maxInclusive, minInclusive, precision, enumeration, etc.

XSD (XML Schema) Example

```xml
<?xml version="1.0"?>
<xs:element name="Bookstore">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Book">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Title" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="Author" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element ref="Date" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Title" type="xsd:string"/>
<xs:element name="Author" type="xsd:string"/>
<xs:element name="Date" type="xsd:Date"/>
<xs:element name="ISBN" type="xsd:integer"/>
<xs:element name="Publisher" type="xsd:string"/>
</xs:element>
</xs:schema>
```

Prefix "xsd" refers to the XMLSchema namespace
"xmlns" refers to the default namespace

Defining the element “Bookstore” as a complex Type
Containing a sequence of 1 or more “Book” elements

When referring to another Element, use “ref”

The Author can be 1 or more

Notice the use of more meaningful data types

Summary of the XML+NS+XSD Layer
The Power of Simplicity

- "When I designed HTML, I chose to avoid giving it more power than it absolutely needed — a "principle of least power", which I have stuck to ever since. I could have used a language like Knuth's TeX but..." — TBL
- Keeps the principles of SGML in place but its spec is thin enough to wave ☺
- To say you are "Using XML." is sort of like saying you are using ASCII
- Using XSD (XML Schema) makes a lot more sense

Limitations of XML & XML Schema

- Tree structure — awkward to do graphs
- More of a syntactic than a semantic specification
- XML has ordered elements — often unnecessary, unnatural in metadata descriptions
- XML allows constructions that mix up some text along with child elements, which are hard to handle.

RDF (Resource Description Framework)

- RDF provides a way of describing resources via metadata (data about data)
  - It restricts the description of resources to triples (subject, predicate, object)
- It provides interoperability between applications that exchange machine understandable information on the Web.
- The original broad goal of RDF was to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain.
- Provides a lightweight ontology system
  - Labeled graph model
  - Subsumption of instance of
  - Property domain and range
- Uses XML as the interchange syntax.
- The formal specification of RDF is available at: [http://www.w3.org/RDF/](http://www.w3.org/RDF/)
RDF Syntax

Subject, Predicate and Object Tuples (Triples)

- Subject: The resource being described.
- Predicate: A property of the resource
- Object: The value of the property

A combination of them is said to be a Statement

Subject, Predicate, and Object: 

http://foo.bar.org/index.html
John Doe

A web page being described
A property of the web page (author)
The value of the predicate (here the author)

RDF Example

<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
   xmlns:s="http://description.org/schema/">
   <rdf:Description about="http://foo.org/index.html">
     <s:Author>John Doe</s:Author>
   </rdf:Description>
</rdf:RDF>

Subject: a resource
Property: a resource
Object: a resource or a literal

In Triples notation:  

Both statements say:  The Author of http://foo.org/index.html is John Doe.

RDF Schema

- A schema defines the terms that will be used in the RDF statements and gives specific meanings to them.
- RDF Schema namespace
- RDF Schema (Triples Notation)
- RDF Schema Example (cont..)

RDF Schema (RDF/XML notation)

RDF Schema Namespace

An rdf:ID attribute names a new resource
(Relationship) is the top level class
PassengerVehicle is a subclass of MotorVehicle

Multiple Inheritance

Domain of a property
Range of a property

Multiple Domains
“Typed Node” Abbreviation

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#
  xml:base="http://example.org/schemas/vehicles">
  <rdfs:Class rdf:ID="MotorVehicle"/>
  <rdfs:Class rdf:ID="PassengerVehicle">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Van">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="MiniVan">
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
  </rdfs:Class>
</rdf:RDF>
```

the rdf:type of MotorVehicle is rdfs:Class (i.e., MotorVehicle is a Class)

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://example.org/schemas/vehicles#"
  xml:base="http://example.org/things">
  <ex:MiniVan rdf:ID="minivan123"/>
</rdf:RDF>
```

the rdf:type of minivan123 is ex:MiniVan (i.e., minivan123 is a MiniVan)

Reification

```
exproducts:triple12345 rdf:subject exproducts:item10245 .
exproducts:triple12345 rdf:object "2.4"^^xsd:decimal .
```

Summary: RDF & RDF Schema layer

- Minimalist model:
  - Class, Property
  - Subclass, Subproperty
  - Domain & Range
- RDF Schema: W3C recommendation, Feb 2004
  - http://www.w3.org/RDF/
- SPARQL: W3C recommendation, Jan 2008
  - http://www.w3.org/TR/rdf-sparql-query/
- Efficient storage and retrieval
  - “Triple store” using database backends

SPARQL Query Language for RDF

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX exproducts: <http://example.org/products#>
PREFIX exterms: <http://example.org/externs#>

SELECT * WHERE {
  ?item a ex:MiniVan .
  ?item exproducts:weight "2.4"^^xsd:decimal .
}
```

Limitations of RDF

- Cannot define properties of properties (unique, transitive)
- No equivalence, disjointness, etc.
- No mechanism of specifying necessary and sufficient conditions for class membership.
- Example: If it is given that ‘XYZ’ has a ‘car’ which is ‘7ft high’, has ‘wide wheels’ and ‘loading space is 4 cub. m’, then we should be able to reason that ‘XYZ’ has an ‘SUV’, as given by the necessary and sufficient conditions for being an ‘SUV’: height > 4ft & wide wheels & loading space > 2 m²

Ontology Vocabulary

Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)
**OWL: Web Ontology Language**

- W3C Recommendation (Feb 10, 2004)
- Description logic substrate
  - Extension of RDF schema
  - Concepts, roles, instances
- Concept constructors
- OWL comes in three flavors
  - OWL Lite (SHIF description logic)
  - OWL DL (SHOIN description logic)
  - OWL Full
- OWL Web Ontology Language Overview
  - [http://www.w3.org/TR/owl-features/](http://www.w3.org/TR/owl-features/)
- Full details at:
  - [http://www.w3.org/2004/OWL/#specs](http://www.w3.org/2004/OWL/#specs)

**Description Logic Basics**

- **Concepts**: unary predicates/formulae with one free variable p(x)
  - E.g., Person, Doctor, HappyParent, (Doctor & Lawyer)
- **Roles**: binary predicates/formulae with two free variables r(x,y)
  - E.g., hasChild, loves, (hasBrother & hasDaughter)
- **Individuals**: constants
  - E.g., John, Mary, Italy
- **Concept/Role constructors** restricted so that:
  - Satisfiability/subsumption is decidable and, if possible, of low complexity
  - No need for explicit use of variables
  - Restricted form of $\exists$ and $\forall$
  - Features such as counting can be succinctly expressed

**Description Logic Semantics**

Semantics given by standard first-order model:

- **Interpretation function $I$**
- **Interpretation domain $\Delta$**

**The Description Logic Family**

- Many description logics: depending on choice of concept/role constructors
- Smallest propositionally closed DL is ALC
  - Concepts constructed using boolean operators: $\land$ (and), $\lor$ (or), $\neg$ (complement)
  - plus restricted quantifiers $\forall$ (some), $\exists$ (all)
  - Only atomic roles
- Example: Person all of whose children are either Doctors or have a child who is a Doctor:
  
  Person $\cup \exists$ hasChild.(Doctor $\lor \exists$ hasChild.Doctor)

**OWL RDF/XML Exchange Syntax**

E.g., Person $\cup \exists$ hasChild.(Doctor $\lor \exists$ hasChild.Doctor):

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

**Class/Concept Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\cap$ Male</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Doctor $\cup$ Lawyer</td>
<td>$C_1(x) \lor \ldots \lor C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \cup \ldots \cup {x_n}$</td>
<td>(John) $\cup$ (many)</td>
<td>$x_1 \lor \ldots \lor x_n \equiv x_1 \land \ldots \land x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>hasChild.Doctor</td>
<td>$\forall y.P(x,y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>hasChild.Lawyer</td>
<td>$\exists y.P(x,y) \land C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\geq n$</td>
<td>$\geq 1$</td>
<td>$\exists y.P(x,y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n$</td>
<td>$\geq 2$</td>
<td>$\exists y.P(x,y)$</td>
</tr>
</tbody>
</table>

C is a concept (class); P is a role (property); $x$ is an individual name

OWL has XMLS datatypes as well as classes in 8P.C and 9P.C
Ontology Axioms

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \subseteq P$</td>
<td>ancestor$^+$ $\subseteq$ ancestor</td>
</tr>
</tbody>
</table>

OWL Syntax | DL Syntax | Example |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>$a : C$</td>
<td>John : Happy-Father</td>
</tr>
<tr>
<td>property</td>
<td>$(a, b) : R$</td>
<td>(John, Mary) : has-child</td>
</tr>
</tbody>
</table>

The Description Logic Family (2)

- S often used for ALC extended with transitive roles
- Additional letters indicate other extensions, e.g.:
  - H for role hierarchy (e.g., hasDaughter $\lor$ hasChild)
  - O for nominals/singleton classes (e.g., {Italy})
  - I for inverse roles (e.g., isChildOf $\leftrightarrow$ hasChild)
  - N for number restrictions (e.g., $\geq 2$ hasChild, $\leq 3$ hasChild)
  - Q for qualified number restrictions (e.g., $\geq 2$ hasChild.Doctor)
  - F for functional number restrictions (Functional(hasMother))
- $S + \text{role hierarchy (H) + inverse (I) + QNR (Q)} = \text{SHIQ}$
- SHIQ is the basis for OWL
  - OWL Lite $\equiv$ SHIQ with functional restrictions (i.e., SHIF)
  - OWL DL $\equiv$ SHIQ extended with nominals (i.e., SHOIQ)

OWL-Lite

- RDF Schema Features:
  - rdfs:subClassOf
  - rdf:Property
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
- Class Intersection:
  - intersectionOf
- (In)Equality:
  - equivalentClass
  - equivalentProperty
  - sameAs
  - differentFrom
  - AllDifferent
  - distinctMembers
- Property Characteristics:
  - ObjectProperty
  - DatatypeProperty
  - inverseOf
  - TransitiveProperty
  - SymmetricProperty
  - FunctionalProperty
  - InverseFunctionalProperty
- Property Restrictions:
  - allValuesFrom
  - someValuesFrom
- Restricted Cardinality:
  - minCardinality
  - maxCardinality
  - cardinality
- Filler Information:
  - hasValue

OWL-DL

- OWL-Lite +
- Class Axioms:
  - oneOf, dataRange
  - equivalentClass (applied to class expressions)
  - rdf:subClassOf (applied to class expressions)
- Boolean Combinations of Class Expressions:
  - unionOf
  - complementOf
  - intersectionOf
- Arbitrary Cardinality:
  - minCardinality
  - maxCardinality
  - cardinality
- Filler Information:
  - hasValue

Description Logics

- Classes are defined in terms of other classes/relations
- Powerful inference algorithms:
  - Subsumption: is classA a subclass of classB given their definitions?
  - Recognition: is instanceA of classA?
  - Classification: automatic reorganization of class hierarchy based on definitions of classes
- Logical proofs
**Classification: Defining an Ontology**

- **animal**
  - **mammal**
  - **sick animal**
  - **dog**
  - **disease**
  - "A dog is a mammal"
  - "A sick animal has a disease"

- **rabies**
  - **has**

**Classification: Defining a “rabid dog”**

- **animal**
  - **mammal**
  - **sick animal**
  - **dog**
  - **disease**
  - **rabies**
  - **has**

**Classification: Classifier Infers “sick animal”**

- **animal**
  - **mammal**
  - **sick animal**
  - **dog**
  - **disease**
  - **rabies**
  - **has**

**Classification: Defining “rabid animal”**

- **animal**
  - **mammal**
  - **sick animal**
  - **dog**
  - **disease**
  - **rabies**
  - **has**

**Classification: Concept Placed in Hierarchy**

- **animal**
  - **mammal**
  - **sick animal**
  - **dog**
  - **disease**
  - **rabies**
  - **has**
  - **rabid animal**
  - **has**

**Complexity of reasoning in Description Logic**

- **$\text{ALC}$**

- **Concept satisfiability**: Polynomial time
- **Model checking**: $\text{coNP}$-hard
Resources for OWL and DL

- Description Logic Handbook, Cambridge University Press
  - http://books.cambridge.org/0521781760.htm
- Description Logic: http://dl.kr.org/
  - complexity: http://www.cs.man.ac.uk/~ezolin/dl
- Web Ontology Language (OWL): http://www.w3.org/2004/OWL/
- Reasoners:
  - Pellet (open source): http://pellet.owldl.com/
  - FaCT++ (open source): http://owl.man.ac.uk/factplusplus/
  - Racer (commercial): http://www.racer-systems.com/
  - (Loom and Powerloom: http://www.isi.edu/isd/LOOM/)
- Ontology Editors:
  - Protégé: http://protege.stanford.edu/
  - Ian Horrocks has great slides on description logics and OWL:
    - http://www.oucl.ox.ac.uk/oucl/work/ian.horrocks/

Proofs: Logical Derivations

- Use the logic to prove things given the set of facts provided
- The derivation of the proof provides the support for the derived facts
- Easier to verify a proof than it is to find one
### Can We Trust the Result

- Need a mechanism to determine who to trust
- Exploit digital signatures to verify that information comes from a trusted source
- Define a “Web of Trust”
  -- You tell the system who you want to trust

### W3C’s Semantic Web Principles

- Everything identifiable is in the Semantic Web (URIs!)
- Partial information
  -- Anyone can say anything about anything
- Web of trust
  -- All statements on the Web occur in some context
- Evolution
  -- Allow combining independent work done by different communities
- Minimalist design
  -- Make the simple things simple, and the complex things possible
  -- Standardize no more than is necessary

### Hypertext: Then and Now

- SOTA circa 1990: Dynatext’s electronic book
  -- A book had to be compiled (like a program) in order to be displayed efficiently
  -- A central link database, to make sure there were no broken links
  -- Text that was fixed and consistent (a whole book)
- WWW:
  -- Links can be added and used at any time
  -- Distributed (must live with broken links!)
  -- Decentralized

### Knowledge Representation: Now and Tomorrow

“To webize KR in general is, in many ways, the same as to webize hypertext. Replace identifiers with URIs. Remove any requirement for global consistency. Put any significant effort into getting critical mass. Sit back.”

-- TBL