Semantic Web Languages
Discussion

- General versus specific Ontology
  - Ontology of everything
  - Many interconnected ontologies
- Task independence/task dependence
- Engineering ontologies
  - Methods, evaluation
  - Ditransitive statements, Multiple inheritance
  - URI’s, Abstract concepts
- Reasoning
  - Owl:nothing
- Ontology Matching / Ontology alignment
Semantic Web Layer ‘Cake’

User Interface & Applications

Trust

Proof

Unifying Logic

Query: SPARQL

Ontology: OWL

Rule: RIF

RDFS

Data interchange: RDF

XML

URI/IRI

Crypto
Bottom layers

- Based on hypertext web technologies
  - URI: unique identification of (semantic web) resources.
  - Unicode: represent and manipulate text in many languages (also natural languages)
  - XML: enables creation of documents composed of structured data.
  - XML Namespaces: way to use markups from more sources; connecting data together
  - Cryptography: digital signature to ensure and verify that statements are coming from trusted source.
Middle layers

- Standardized by W3C to enable building SW applications.
  - Resource Description Framework (RDF):
    - Specified statements as triples.
    - Enables to represent information about resources in the form of graph.
  - RDF Schema (RDFS): the basic vocabulary for RDF
    - e.g. hierarchies of classes and properties.
  - Web Ontology Language (OWL):
    - extends RDFS by adding constructs to describe semantics of statements.
    - Allows stating additional constraints, such as for example cardinality, restrictions of values, or characteristics of properties such as transitivity.
    - It is based on description logic and so brings reasoning power to the SW.
  - SPARQL: RDF query language:
    - it can be used to query any RDF-based data;
    - necessary to retrieve information for semantic web applications.
Top Layers

- Unrealized technologies
  - RIF or SWRL rule languages. Allows describing relations not supported in description logic used in OWL.
  - More cryptography
  - Proof: relying on formal logic during deriving new information.
  - Trust on statements will be supported by Cryptography and Proof
  - User interface: enable humans to use semantic web applications.
Ontology Languages for the Semantic Web

Part 1: What is ontology?
Part 2: SW Languages
Ontology
Ontology

- Philosophy - Aristoteles
  - The metaphysical study of the nature of being and existence
- In Artificial Intelligence
  - “a shared and common understanding of some domain that can be communicated between people and application systems” - Gruber
What is an ontology?

- **Formal, explicit specification of a shared conceptualization**
  - Machine readable
  - Concepts, properties, functions, axioms are explicitly defined
  - Consensual knowledge
  - Abstract model of some phenomena in the world
Ontology defines concepts comprising the vocabulary of a topic area as well as the rules for combining concepts and the relations to define extensions to the vocabulary.

- An ontology specifies how to view the world.
- An ontology specifies in which terms we should speak about the world.

Important:
- how/why/when do we use an ontology?
How to use an ontology?

- To share the common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make the domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge
Benefits

- Communication between people
- Interoperability between software agents
- Reuse of domain knowledge
- Make domain knowledge explicit
- Analyze domain knowledge

But... building an ontology is not a goal in itself.
Requirements for ontology languages

- Well understood
- Formally specified
- (Relatively) easy to use
- Amenable to machine processing
Ontology concepts: linguistic view

- **Statement:**
  
  "Mary drives a Ford".

- **Subject:** Mary
- **Predicate:** drives
- **Object:** Ford

- **But...**
  - Types?
  - Meanings?
  - Order matters?
Ontology concepts: object-oriented view

- **Types/Classes/Concepts**
  - Sets of objects sharing certain characteristics
  - Equivalent to unary predicates in FOL

- **Relations/Properties/Roles**
  - Sets of pairs (tuples) of objects
  - Equivalent to binary predicates in FOL

- **Objects/Instances/Individuals**
  - Elements of the domain of discourse
  - Equivalent to constants in FOL (first order logic)
Classes

- Represent concepts or tasks of the domain, organized in taxonomies
  - Example: student, lecturer, car, course, road,…
- Express regularities
  - All members of a class share certain properties
- Defined in terms of necessary conditions
- Appropriate set of distinctions
Relations

- **Generalization/Specialization**
  - **Is-a:**
    - Car represents a subclass of the more general class Vehicle
    - Poodle, labrador and chihuahua are specializations of class Dog
  - **Inheritance**
    - All sub-classes inherit the characteristics of the super-class
  - *Important:* **is-a is the standard relation between classes in Protégé!!**

- **Aggregation**
  - Part-of: Engine is part of a car

- **Association**
  - Some semantic relation between independent classes; e.g. drive: People drive cars
Other Components

- **Functions/attributes**
  - Concept properties
  - Example: *Price-of-a-used-car, student-loginname*

- **Axioms/rules**
  - Model sentences that are always true
  - Property restrictions (type, cardinality, domain)
  - Example:
    - \( \text{If register}(st, \text{courseA}) \text{ then passed}(st, \text{courseB}) \)

- **Instances**
  - to represent specific elements
  - Example:
    - *Entity called Peter is an instance of Student class*
Ontology Specification/Visualisation

Many languages for Explicit Specification

- **Graphical notations**
  - Semantic networks
  - Topic Maps (see http://www.topicmaps.org/)
  - UML
  - RDF

- **Logic based**
  - Description Logics (OWL)
  - Rules (e.g., RuleML, LP/Prolog)
  - First Order Logic (e.g., KIF)
  - Conceptual graphs

- **Probabilistic/fuzzy**
Ontology Specification/Visualisation

- **Graphical notations:** Semantic networks
Ontology Specification/Visualisation

- **Graphical notations:** Topic Maps
Ontology Specification/Visualisation

- **Graphical notations:** UML
Ontology Specification/Visualisation

- **Graphical notations:** RDF
Ontology Specification/Visualisation

- Logic based

Every gardener likes the sun.
(Ax) gardener(x) => likes(x, Sun)

You can fool some of the people all of the time.
(Ex)(At) (person(x) ^ time(t)) => can-fool(x, t)

You can fool all of the people some of the time.
(Ax)(Et) (person(x) ^ time(t)) => can-fool(x, t)

All purple mushrooms are poisonous.
(Ax) (mushroom(x) ^ purple(x)) => poisonous(x)

No purple mushroom is poisonous.
¬(Ex) purple(x) ^ mushroom(x) ^ poisonous(x)
(Ax) (mushroom(x) ^ purple(x)) => ¬poisonous(x)

There are exactly two purple mushrooms.
(Ex)(Ey) mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ¬(x=y) ^ (Ax)(mushroom(z) ^ purple(z)) => ((x=z) v (y=z))

Clinton is not tall.
¬tall(Clinton)
Ontology Languages

- **Degree of formality varies widely**
  - Increased formality makes languages more amenable to machine processing (e.g., automated reasoning)
Using Ontologies: Classification

- To **classify** an object, phenomenon or pattern is to identify it as a **member of a class**
- To **categorize** x is to come to think of x as an **instance of a category**

**Why** do we classify?
- Simplify the world
- Generalize knowledge to new instances
- Predict unknown properties (ex: mean dogs)

Knowledge is needed in order to
- match elements of data space to elements of solution space
- select from a predefined set of solutions
Enable machines to classify
Using ontologies: interpretation

World

- Daisy isA Cow
- Cow kindOf Animal
- Mary isA Person
- Person kindOf Animal
- 23-XV-TZ isA Car
- Mary drives 23-XV-TZ

Ontology

- Daisy isA Cow
- Cow kindOf Animal
- Mary isA Person
- Person kindOf Animal
- 23-XV-TZ isA Car
- Mary drives 23-XV-TZ

Interpretation

\[ \{<a,b>, \ldots > \subseteq \Delta \times \Delta \]
**Taxonomy, Thesaurus, Ontology**

- **Taxonomy**: collection of controlled vocabulary terms organized into a hierarchical structure.
  - Each term in a taxonomy is in one or more parent-child relationships to other terms in the taxonomy.

- **Thesaurus**: networked collection of controlled vocabulary terms.
  - Thesaurus uses associative relationships in addition to parent-child relationships.

- **Ontology**: controlled vocabulary expressed in an ontology representation language.
  - Can mean different things, e.g. glossaries & data dictionaries, thesauri & taxonomies, schemas & data models, and formal ontologies & inference.
## Comparison

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Taxonomy</th>
<th>Thesaurus</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>Natural sciences (Linnaeus)</td>
<td>Library sciences</td>
<td>Metaphysics, AI</td>
</tr>
<tr>
<td>Traditional</td>
<td>Classify things within a domain</td>
<td>Describe the content of a collection</td>
<td>automated reasoning</td>
</tr>
<tr>
<td>purpose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationships</td>
<td>Hierarchical</td>
<td>Hierarchical, associational</td>
<td>Hierarchical, associational</td>
</tr>
<tr>
<td>Complexity</td>
<td>Low</td>
<td>Middle</td>
<td>High</td>
</tr>
<tr>
<td>Polyhierarchy</td>
<td>Preferably not</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Standard</td>
<td>None</td>
<td>ISO, ANSI/NISO</td>
<td>None</td>
</tr>
</tbody>
</table>

*(thesis work by K.Schwarz, CKE 2005)*
Semantic Web Languages
Understanding each other on the Web

Lingua franca? OR Babel tower?
Lingua franca for the web?

- XML provides uniform framework to for interchange of data and metadata between applications
- Semantic Web functionality requires framework but for interchange of *semantics* of data between agents
XML

Informal definition:
- XML is a markup language for representation of documents which contain structured information.

Usages:
- Data exchange (e.g. RSS, SOAP-envelope, ad hoc B2B data exchange)
- Web services
- Data integration
- Content publishing (single source multiple output)
- Multimedia presentations (SMIL)
XML doesn’t do the trick

- XML describes internal structure and *not* the common interpretation of it.

- You don’t want to map DTD’s onto one another, that’s EASY.
- You want to map *CONCEPTS*…

- So you’d have to re-engineer the whole thing!
RDF: Resource Description Language

- RDF is a framework for describing resources on the web
- RDF provides a model for data, and a syntax so that independent parties can exchange and use it
- RDF is written in XML
- RDF is a W3C Recommendation
The combination of a Resource, a Property, and a Property value forms a **Statement** (known as the **subject, predicate and object** triple)
RDF(S): Main Issues

- Language for \textit{expressing} metadata.
  - Information \textit{about} information
  - Structured data \textit{about} data

- Must be:
  - \textit{Universal} (so all can understand)
  - \textit{Flexible} (to incorporate different types)
  - \textit{Extensible} (flexible to custom types)
  - \textit{Simple} (to encourage adoption)
  - \textit{Modular} (so that schemes can be mixed, extended)
The RDF Data Model

- Statements are <subject, predicate, object> triples:

- Is represented as <Ann, hasColleague, Bob>
- Statements describe properties of resources
- A resource is a URI representing a (class of) object(s):
  - a document, a picture, a paragraph on the Web;
  - a book in the library, a real person (?)
  - isbn://5031-4444-3333
  - …
- URI = Uniform Resource Identifier
  - “The generic set of all names/addresses that are short strings that refer to resources“
- Properties themselves are also resources (URIs)
Linking Statements

- The subject of one statement can be the object of another.
- Such collections of statements form a directed, labelled graph.

Note that the object of a triple can also be a “literal” (a string).
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Artist</th>
<th>Country</th>
<th>Company</th>
<th>Price</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>#111</td>
<td>Empire Burlesque</td>
<td>Bob Dylan</td>
<td>USA</td>
<td>Columbia</td>
<td>10.90</td>
<td>1985</td>
</tr>
<tr>
<td>#112</td>
<td>Hide your heart</td>
<td>Bonnie Tyler</td>
<td>UK</td>
<td>CBS Records</td>
<td>9.90</td>
<td>1988</td>
</tr>
</tbody>
</table>
Example (2/2)

```xml
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cd="http://www.recshop.fake/cd#">
    <rdf:Description rdf:about="http://www.recshop.fake/cd/111">
        <cd:title>Empire Burlesque</cd:title>
        <cd:artist>Bob Dylan</cd:artist>
        <cd:country>USA</cd:country>
        <cd:company>Columbia</cd:company>
        <cd:price>10.90</cd:price>
        <cd:year>1985</cd:year>
    </rdf:Description>
    <rdf:Description rdf:about="http://www.recshop.fake/cd/112">
        <cd:title>Hide your heart</cd:title>
        <cd:artist>Bonnie Tyler</cd:artist>
        <cd:country>UK</cd:country>
        <cd:company>CBS Records</cd:company>
        <cd:price>9.90</cd:price>
        <cd:year>1988</cd:year>
    </rdf:Description>
    ...
</rdf:RDF>
```
RDF Reification

- Allows making statements about statements
- Turns statement into a resource
RDF Schema

- An extension to RDF
- Formal language meant to describe semantics of any domain
- Provides fixed primitives to model domains
- RDF Schema is a *primitive* ontology language
RDF(S) vs. XML

- Universal expressive power
  - Anything can be encoded in XML
  - Holds but extensions required
- Support for Syntactic Interoperability
- Support for Semantic Interoperability
  - XML is aiming at the structure of documents
  - RDF structure aims at representing domain model
RDFS Interpretation Example

- If RDFS graph includes triples
  - \(<\text{Species}, \text{type}, \text{Class}>\>
  - \(<\text{Lion}, \text{type}, \text{Species}>\>
  - \(<\text{Leo}, \text{type}, \text{Lion}>\>
  - \(<\text{Lion}, \text{subClassOf}, \text{Mamal}>\>
  - \(<\text{Mamal}, \text{subClassOf}, \text{Animal}>\>

- Interpretation conditions imply existence of triples
  - \(<\text{Lion}, \text{subClassOf}, \text{Animal}>\>
  - \(<\text{Leo}, \text{type}, \text{Mamal}>\>
  - \(<\text{Leo}, \text{type}, \text{Animal}>\>
  - ...

RDF/RDFS “Liberality”

- No distinction between classes and instances (individuals)
  
  `<Species, type, Class>`
  `<Lion, type, Species>`
  `<Leo, type, Lion>`

- Properties can themselves have properties
  
  `<hasDaughter, subPropertyOf, hasChild>`
  `<hasDaughter, type, familyProperty>`

- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
  
  `<type, range, Class>`
  `<Property, type, Class>`
  `<type, subPropertyOf, subClassOf>`
Other problems

- **No localised range and domain constraints**
  - Can’t say that the range of hasChild is person when applied to persons and elephant when applied to elephants

- **No existence/cardinality constraints**
  - Can’t say that all *instances* of person have a mother that is also a person, or that persons have exactly 2 parents

- **No transitive, inverse or symmetrical properties**
  - Can’t say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
- RDFS too weak to describe resources in sufficient detail
- Difficult to provide reasoning support
  - No “native” reasoners for non-standard semantics
  - May be possible to reason via FOL axiomatisation
Desirable features identified for Web Ontology Language:

- Extends existing Web standards
  - Such as XML, RDF, RDFS
- Easy to understand and use
  - Should be based on familiar KR idioms
- Formally specified
- Of “adequate” expressive power
- Possible to provide automated reasoning support
OWL Language

- **W3C Recommendation** (i.e., a standard like HTML and XML)
- Three species of OWL
  - OWL full is union of OWL syntax and RDF
  - OWL DL restricted to FOL fragment
  - OWL Lite is “easier to implement” subset of OWL DL

- Semantic layering
  - OWL DL \(\approx\) OWL full within DL fragment
  - DL semantics officially definitive

- OWL DL based on Description Logic (DL)
  - Benefits from many years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimised)
OWL in detail: the five conditions

1. Well defined syntax
2. Well defined semantics  
   (Some articles call this: *formal* semantics)
3. Efficient reasoning support
4. Sufficient expressive power
5. Convenience of expression
Why “formal semantics”? 

- Class membership
  \[ A \text{ subcl } B \text{ subcl } C, \quad \rightarrow \quad A \text{ subcl } C \]

- Equivalence of classes
  \[ A \text{ equals } B \text{ equals } C, \quad \rightarrow \quad A \text{ equals } C \]

- Consistency
  Can be automatically checked

- Classification
  If class has properties this and that, it’s probably of class type A
Why “reasoning support”?

- Checking consistency
- Checking for unintended relations
- Automatic classification
  (remember we defined a need for this last week)
## OWL as DL: Class 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>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ 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>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${john} \sqcup {mary}$</td>
<td>$x = x_1 \lor \ldots \lor x = x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$\forall y. P(x, y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\exists y. P(x, y) \land C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1$ hasChild</td>
<td>$\exists y. P(x, y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2$ hasChild</td>
<td>$\exists y. P(x, y)$</td>
</tr>
</tbody>
</table>

- E.g., $\exists$ hasAge. nonNegativeInteger
# OWL as DL: Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td></td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td></td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td></td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>{x_1} $\equiv$ {x_2}</td>
<td></td>
</tr>
<tr>
<td>differentFrom</td>
<td>{x_1} $\sqsubseteq \neg${x_2}</td>
<td></td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td></td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td></td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^-$</td>
<td></td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td></td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$T \sqsubseteq \leq 1P$</td>
<td></td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$T \sqsubseteq \leq 1P^-$</td>
<td></td>
</tr>
</tbody>
</table>

Examples:
- Human $\sqsubseteq$ Animal $\sqcap$ Biped
- Man $\equiv$ Human $\sqcap$ Male
- Male $\sqsubseteq$ $\neg$Female
- \{President\_Bush\} $\equiv$ \{G\_W\_Bush\}
- \{john\} $\sqsubseteq$ $\neg$\{peter\}
- hasDaughter $\sqsubseteq$ hasChild
- cost $\equiv$ price
- hasChild $\equiv$ hasParent$^-$
- ancestor$^+$ $\sqsubseteq$ ancestor
- $T \sqsubseteq \leq 1$hasMother
- $T \sqsubseteq \leq 1$hasSSN$^-$