Semantic Web and Multimedia
- Ontologies & Their Languages –

Steffen Staab
http://isweb.uni-koblenz.de
Ontologies
„People can’t share knowledge if they do not speak a common language.“ [Davenport & Prusak, 1998]

Gruber 93:

An Ontology is a

- formal specification ➔ Executable, to discuss
- of a shared ➔ group of stakeholders
- conceptualization ➔ about concepts
- of a domain of interest ➔ between application and single truth
Taxonomy := Segmentation, classification and ordering of elements into a classification system according to their relationships between each other
Thesaurus

Terminology for specific domain
Taxonomy plus fixed relationships (similar, synonym, related to) originate from library science
• Topics (nodes), relationships and occurrences (to documents)
• ISO-Standard
• typically for navigation- and visualisation
• From publishing practice (back of the book index)
Ontology

Representation Languages: Predicate Logic, Datalog, F-Logic
Standards: RDF(S); OWL

ISWeb - Information Systems & Semantic Web
Steffen Staab
staab@uni-koblenz.de
Ontologies - Some Examples

General purpose ontologies:
- DOLCE, http://www.loa-cnr.it/DOLCE.html

Multimedia Ontologies
- Acemedia harmonization effort: http://www.acemedia.org/aceMedia/reference/multimedia_ontology/

Domain and application-specific ontologies:
- Dublin Core, http://dublincore.org/

Semantic Desktop Ontologies
- X-COSIM Ontology, http://isweb.uni-koblenz.de/Research/X-cosim

Web Services Ontologies
- Core ontology of services http://cos.ontoware.org
- OWL-S, http://www.daml.org/services/owl-s/1.0/

Ontologies in a wider sense
Ontologies and Their Relatives (cont’d)

Front-End

- Thesauri
- Topic Maps
- Navigation
- Information Retrieval
- Sharing of Knowledge
- Query Expansion

Ontologies

- Extended ER-Models
- Queries
- Mediation

Back-End

- Semantic Networks
- Consistency Checking
- EAI
- Reasoning
- Predicate Logic
## Ontology Trade-off

<table>
<thead>
<tr>
<th>Very formal (e.g. DOLCE)</th>
<th>Informal (e.g. Gene Ontology)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A lot of reasoning power</td>
<td>Little to no reasoning possible</td>
</tr>
<tr>
<td>Expensive to build</td>
<td>Comparatively inexpensive (total costs for Gene Ontology are not low!)</td>
</tr>
<tr>
<td>Misunderstandings can be corrected by expert developers (costs may be incurred)</td>
<td>Misunderstandings due to ambiguity are hard to correct (very high costs may be incurred!)</td>
</tr>
<tr>
<td>Ontology MUST be (at least partially) hidden from its users</td>
<td>Ontology may appeal to intuition of user</td>
</tr>
</tbody>
</table>
RDF
RDF Data Model

Resources
- A resource is a thing you talk about (can reference)
- Resources have URI’s
- RDF definitions are itself Resources (linkage)

Properties
- slots, defines relationship to other resources or atomic values

Statements
- “Resource has Property with Value”
- (Values can be resources or atomic XML data)

Similar to Frame Systems
**A simple Example**

### Statement
- “Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila”

### Structure
- Resource (subject) http://www.w3.org/Home/Lassila
- Property (predicate) http://www.schema.org/#Creator
- Value (object) "Ora Lassila"

### Directed graph

```
http://www.w3.org/Home/Lassila s:Creator Ora Lassila
```
To add properties to Creator, point through an intermediate Resource.
Collection Containers

Multiple occurrences of the same PropertyType doesn’t establish a relation between the values

- The Millers own a boat, a bike, and a TV set
- The Millers need (a car or a truck)
- (Sarah and Bob) bought a new car

RDF defines three special Resources:

- **Bag** unordered values rdf:Bag
- **Sequence** ordered values rdf:Seq
- **Alternative** single value rdf:Alt

- Core RDF does not enforce ‘set’ semantics amongst values
Example: Bag

The students in course 6.001 are Amy, Tim, John, Mary, and Sue.
Making statements about *statements* requires a process for transforming them into Resources

- **subject** the original referent
- **predicate** the original property type
- **object** the original value
- **type** rdf:Statement

Distinguish:

- The image depicts Henry walking on water
- Henry walks on water
Photo1 depicts that
- http://www.mit.edu/~lieber
- S:WalksOn
- AtlanticOcean
Example: **Reification**

**Photo1 depicts that**
- `http://www.mit.edu/~lieber`
- `S:WalksOn`
- `AtlanticOcean`
Datamodel does not enforce particular syntax
Specification suggests many different syntaxes based on XML
General form:

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
    <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
  </rdf:Description>
</rdf:RDF>
```
Resulting Graph

`<rdf:RDF>`
  `<rdf:Description about="http://www.w3.org/Home/Lassila">
    `<s:Creator>Ora Lassila</s:Creator>`
    `<s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
  </rdf:Description>`
`</rdf:RDF>`
Typing Information

<s:Homepage rdf:about="http://www.w3.org/Home/Lassila"
          s:Creator="Ora Lassila"/>
<s:Title>Ora's Home Page</s:Title>
<s:createdWith>
  <s:HTMLEditor rdf:about="http://www.w3c.org/amaya"/>
</s:createdWith>
</s:Homepage>

Subject (OID)

RDF Syntax II: Syntactic Varieties

Property

RDF

http://www.w3.org/Home/Lassila
rdf:type
s:Homepage

Ora Lassila
s:Creator

http://www.w3.org/amaya
rdf:type
HTMLEditor

s:createdWith

s:Homepage

RDF just defines the datamodel
Need for definition of vocabularies for the datamodel - an Ontology Language!
RDF schemas are Web resources (and have URIs) and can be described using RDF
RDF-Schema: Example

s = rdfs:subClassOf
\[ s = rdfs:subClassOf \]
\[ t = rdfs:subClassOf \]
\[ \text{xyz:PassengerVehicle} \]
\[ s = \text{rdfs:subClassOf} \]
\[ \text{xyz:Van} \]
\[ s = \text{rdfs:subClassOf} \]
\[ \text{xyz:MiniVan} \]
\[ t = \text{rdfs:subClassOf} \]
\[ \text{xyz:myvan} \]
<rdfs:description about="Xyz:Minivan">
  <rdfs:subclassOf about="xyz:Van"/>
</rdfs:description>

<rdfs:description about="myvan">
  <rdf:type about="xyz:MiniVan"/>
</rdfs:description>

**Predicate Logic Consequences:**

For all $X$: \(\text{type}(X, \text{MiniVan}) \rightarrow \text{type}(X, \text{Van})\).

For all $X$: \(\text{subclassOf}(X, \text{MiniVan}) \rightarrow \text{subclassOf}(X, \text{Van})\).
Rdf:property

.rdf:description about="possesses">
  <rdf:type about="property"/>
  <rdfs:domain about="person"/>
  <rdfs:range about="vehicle"/>
</rdf:description>
.rdf:description about="peter">
  <possesses>petersminivan</possesses>
</rdf:description>

Predicate Logic Consequences:
Forall X,Y: possesses (X,Y) -> (type(X,person) & type(Y,vehicle)).
OWL
OWL – General

W3C Recommendation since 2004
More work on OWL2 to come
Semantic fragment of FOL
Four variants:
OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL2
OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL Full

RDFS is fragment of OWL Full
OWL DL is decidable
OWL DL = SHOIN(D) (description logics)
W3C-Documents contain many more details that we cannot talk about here
OWL Overview

OWL – Syntax and semantics

a. Description logics: SHOIN(D)

b. OWL as SHOIN(D)

c. Serializations

d. Knowledge modelling in OWL
General DL Architecture

Knowledge Base

- Tbox (schema)
  - Man ≡ Human \& Male
  - Happy-Father ≡ Man \& \exists has-child.Female \& ...

- Abox (data)
  - Happy-Father(John)
  - has-child(John, Mary)

Inference System

Interface

- Sometimes: „TBox“ is equated with „Ontology“
- Sometimes: „Knowledge Base“ is equated with Ontology
- My preference: „Ontology“ is everything in KB that is constant in all worlds possible in the given domain → Find out what the other person wants to say
DLs are a Family of logic-based formalism for knowledge representation
Special language characterized by:
- Constructors to define complex concepts and roles based on simpler ones.
- Set of axiom to express facts using concepts, roles and individuals.

ALC is the smallest DL, which is propositionally closed:
- Constructors are noted by $\cap$, $\cup$, $\neg$ (intersection, union, negation)
- Quantors define how roles are to be interpreted:
  Man $\cap \exists$hasChild.Female $\cap \exists$hasChild.Male
  $\cap \forall$hasChild.(Rich $\sqcup$ Happy)
Number restrictions (cardinality constraints) for roles:
≥3 hasChild, ≤1 hasMother
Qualified number restrictions:
≥2 hasChild.Female, ≤1 hasParent.Male
Nominals (definition by extension):
{Italy, France, Spain}
Concrete domains (datatypes): hasAge.(≥21)
Inverse roles: hasChild^− ≡ hasParent
Transitive roles: hasAncestor^* (descendant)
Role composition: hasParent.hasBrother (uncle)
DL Knowledge Bases consist of two parts (in general):

- **TBox**: Axioms, describing the structure of a modelled domain (conceptual schema):
  - HappyFather \(\equiv\) Man \(\sqcap\) \(\exists\)hasChild.Female \(\sqcap\) …
  - Elephant \(\sqsubseteq\) Animal \(\sqcap\) Large \(\sqcap\) Grey
  - transitive(hasAncestor)

- **Abox**: Axioms describing concrete situations (data, facts):
  - HappyFather(John)
  - hasChild(John, Mary)

The distinction between TBox/ABox does not have a deep logical distinction … but it is common useful modelling practice.
OWL – **Syntax and semantics**

a. Description logics: SHOIN(D)
b. **OWL as SHOIN(D)**
c. Serializations
d. Knowledge modelling in OWL
### OWL DL 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>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</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>$\leq nP$</td>
<td>$\leq 1$hasChild</td>
<td>$\exists y.P(x, y) \land y \leq n$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2$hasChild</td>
<td>$\exists y.P(x, y) \land y \geq n$</td>
</tr>
</tbody>
</table>

Nesting of expression is allowed at arbitrary depth:

\[ \text{Person} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \]
### OWL DL 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>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>disjointWith</td>
<td>( C_1 \sqsubseteq \neg C_2 )</td>
<td>Male ( \sqsubseteq \neg ) Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>{x_1} \equiv {x_2}</td>
<td>{President_Bush} \equiv {G_W_Bush}</td>
</tr>
<tr>
<td>differentFrom</td>
<td>{x_1} \sqsubseteq \neg {x_2}</td>
<td>{john} \sqsubseteq \neg {peter}</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>inverseOf</td>
<td>( P_1 \equiv P_2^\dashv )</td>
<td>hasChild ( \equiv ) hasParent^\dashv</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>( P^+ \sqsubseteq P )</td>
<td>ancestor^\dashv ( \sqsubseteq ) ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>( \top \sqsubseteq \sqsubseteq \sqsubseteq \top P )</td>
<td>( \top \sqsubseteq \sqsubseteq \sqsubseteq \top ) hasMother</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>( \top \sqsubseteq \sqsubseteq \sqsubseteq \top P )</td>
<td>( \top \sqsubseteq \sqsubseteq \sqsubseteq \top ) hasSSN^\dashv</td>
</tr>
</tbody>
</table>

**General Class Inclusion (\(\sqsubseteq\)):**

\[ C \equiv D \text{ IFF } ( C \sqsubseteq D \text{ und } D \sqsubseteq C ) \]

**Obvious equivalances with FOL:**

\[ C \equiv D \leftrightarrow (\forall x) \ (C(x) \leftrightarrow D(x)) \]
\[ C \sqsubseteq D \leftrightarrow (\forall x) \ (C(x) \rightarrow D(x)) \]
**OWL/DL Example**

**Terminological Knowledge (TBox):**
Human ⊑ ∃hasParent.Human  
Orphan ≡ Human ⊓ ¬∃childOf.Alive

**Knowledge about Individuals (ABox):**
Orphan(harrypotter)  
ParentOf(jamespotter,harrypotter)

Semantics and logical consequences may be derived by translation to FOL
### Model theoretical Semantics – direct

#### Concept expressions

<table>
<thead>
<tr>
<th></th>
<th>Subset of $\Delta^I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td></td>
</tr>
<tr>
<td>$\neg C$</td>
<td>$\Delta^I \setminus C^I$</td>
</tr>
<tr>
<td>$C \cap D$</td>
<td>${x</td>
</tr>
<tr>
<td>$C \cup D$</td>
<td>${x</td>
</tr>
<tr>
<td>$\exists R.C$</td>
<td>${x</td>
</tr>
<tr>
<td>$\forall R.C$</td>
<td>${x</td>
</tr>
<tr>
<td>$\geq n R.C$</td>
<td>${x</td>
</tr>
<tr>
<td>$\leq n R.C$</td>
<td>${x</td>
</tr>
<tr>
<td>${i_1, \ldots, i_n}$</td>
<td>${i_1^I, \ldots, i_n^I}$</td>
</tr>
</tbody>
</table>

#### Role expressions

<table>
<thead>
<tr>
<th></th>
<th>Subset of $\Delta \times \Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td></td>
</tr>
<tr>
<td>$R^-$</td>
<td>${(y, x)</td>
</tr>
</tbody>
</table>

#### Ontology (=Knowledge Base)

##### Concept Axioms (TBox)

<table>
<thead>
<tr>
<th>$\forall C \subseteq D$</th>
<th>$C^I \subseteq D^I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C = D$</td>
<td>$C^I = D^I$</td>
</tr>
</tbody>
</table>

##### Role Axioms (rarely: RBox)

| $R \subseteq S$       | $R^I \subseteq S^I$ |

##### Assertional Axioms (ABox)

<table>
<thead>
<tr>
<th>$C(a)$</th>
<th>$a^I \in C^I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(a, b)$</td>
<td>$(a^I, b^I) \in R^I$</td>
</tr>
<tr>
<td>$a = b$</td>
<td>$a^I = b^I$</td>
</tr>
<tr>
<td>$a \neq b$</td>
<td>$a^I \neq b^I$</td>
</tr>
</tbody>
</table>
Concrete Domains

Strings and Integers (required by W3C OWL rec)
Further datatypes may be supported.
Restricted to *decidable* predicates over the concrete domain

Each concrete domain must be implemented separately and then included into the reasoner (weak analogy: built-ins – but no procedural semantics!)
Content

OWL – **Syntax and model theoretic semantics**

a. Description logics: SHOIN(D)
b. OWL as SHOIN(D)
c. Serializations
d. Knowledge modelling in OWL
<table>
<thead>
<tr>
<th>Syntax</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL RDF Syntax</td>
<td>W3C recommendation</td>
</tr>
<tr>
<td>OWL Abstract Syntax</td>
<td>W3C recommendation</td>
</tr>
<tr>
<td></td>
<td>See next section</td>
</tr>
<tr>
<td>OWL XML Syntax</td>
<td>W3C document</td>
</tr>
<tr>
<td>DL Notation</td>
<td>widely used in scientific contexts</td>
</tr>
<tr>
<td>FOL Notation</td>
<td>uncommon</td>
</tr>
</tbody>
</table>
Example: RDF Syntax

Person ⊓ ∀hasChild.(Doctor ⊔∃hasChild.Doctor):

```
<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>
```

Take home message: avoid RDF serializations – use existing APIs (where possible)
Content

OWL – **Syntax and model theoretic semantics**

a. Description logics: SHOIN(D)
b. OWL as SHOIN(D)
c. Serializations
d. Knowledge modelling in OWL
Example ontology and conclusion from
http://owl.man.ac.uk/2003/why/latest/#2
Also an example for OWL Abstract Syntax.

Namespace(a = <http://cohse.semanticweb.org/ontologies/people#>)
Ontology(
    ObjectProperty(a:drives)
    ObjectProperty(a:eaten_by)
    ObjectProperty(a:eats inverseOf(a:eaten_by) domain(a:animal))
    ...
    Class(a:adult partial annotation(rdfs:comment "Things that are adult."))
    Class(a:animal partial restriction(a:eats someValuesFrom (owl:Thing))))
    Class(a:animal_lover complete intersectionOf(restriction(a:has_pet
        minCardinality(3)) a:person))
    ...
)
Knowledge modelling: examples

Class(a:bus_driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:bus)))))

\[ \text{bus\_driver} \equiv \text{person} \sqcap \exists \text{drives\_bus} \]

Class(a:driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:vehicle)))))

\[ \text{driver} \equiv \text{person} \sqcap \exists \text{drives\_vehicle} \]

Class(a:bus partial a:vehicle)

\[ \text{bus} \sqsubseteq \text{vehicle} \]

A bus driver is a person that drives a bus.
A bus is a vehicle.
A bus driver drives a vehicle, so must be a driver.
The subclass is inferred due to subclasses being used in existential quantification.
Knowledge modelling: examples

Class(a:driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:vehicle))))

\[ \text{driver} \equiv \text{person} \cap \exists \text{drives.vehicle} \]

Class(a:driver partial a:adult)

\[ \text{driver} \sqsubseteq \text{adult} \]

Class(a:grownup complete intersectionOf(a:adult a:person))

\[ \text{grownup} \equiv \text{adult} \cap \text{person} \]

Drivers are defined as persons that drive cars (complete definition)
We also know that drivers are adults (partial definition)
So all drivers must be adult persons (e.g. grownups)

An example of axioms being used to assert additional necessary information about a class. We do not need to know that a driver is an adult in order to recognize one, but once we have recognized a driver, we know that they must be adult.
Knowledge modelling: Example

Individual(a:Walt type(a:person) value(a:has_pet a:Huey) value(a:has_pet a:Louie) value(a:has_pet a:Dewey))
Individual(a:Huey type(a:duck))
Individual(a:Dewey type(a:duck))
Individual(a:Louie type(a:duck))
DifferentIndividuals(a:Huey a:Dewey a:Louie)
Class(a:animal_lover complete intersectionOf(a:person restriction(a:has_pet minCardinality(3))))
ObjectProperty(a:has_pet domain(a:person) range(a:animal))

Walt has pets Huey, Dewey and Louie.
Huey, Dewey and Louie are all distinct individuals.
Walt has at least three pets and is thus an animal lover.

Note that in this case, we don’t actually need to include person in the definition of animal lover (as the domain restriction will allow us to draw this inference).
Knowledge modelling: OWA vs. CWA

OWA: Open World Assumption
The existence of further individuals is possible if it is not explicitly excluded.

OWL uses OWA!

CWA: Closed World Assumption
One assumes that the knowledge base contains all known individuals and all known facts.

Are all children of Bill male?

No idea, since we do not know all children of Bill.

If we assume that we know everything about Bill, then all of his children are male.

<table>
<thead>
<tr>
<th>child(Bill, Bob)</th>
<th>Man(Bob)</th>
<th>$\leq 1$ child.T(Bill)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Rightarrow \forall$ child.Man(Bill)</td>
<td>don’t know</td>
<td>$\Rightarrow \forall$ child.Man(Bill)</td>
</tr>
<tr>
<td>DL answers</td>
<td>Prolog</td>
<td>Now we know everything about Bill’s children.</td>
</tr>
</tbody>
</table>

If we assume that we know everything about Bill, then all of his children are male.
Thank You

Acknowledgements to Pascal Hitzler, York Sure@Karlsruhe for some slides on OWL