Semantic Web
- 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
- of a *shared* conceptualization
- of a *domain* of interest

⇒ Executable, to discuss
⇒ group of stakeholders
⇒ about concepts
⇒ between application and single truth
Ontology: Taxonomy

Taxonomy := Segmentation, classification and ordering of elements into a classification system according to their relationships between each other.
Ontology: Thesaurus

- Terminology for specific domain
- Taxonomy plus fixed relationships (similar, synonym, related to)
- Originate from bibliography
• Topics (nodes), relationships and occurrences (to documents)
• ISO-Standard
• typically for navigation- and visualisation

ISWeb - Information Systems & Semantic Web
Ontology in our sense

- Representation Language: Predicate Logic (F-Logic)
- Standards: RDF(S); OWL

Rules

- T described_in D
- T is_about D
- P writes D
- D is_about T
- P knows T

- 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

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

Taxonomies

- Query Expansion

Ontologies

- Queries
- Mediation
- EAI
- Reasoning

Semantic Networks

- Consistency Checking

Extended ER-Models

Back-End

- Predicate Logic
- Extended ER-Models
- EAI

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

http://www.w3.org/Home/Lassila

Person://fi/654645635

Name: Ora Lassila
Email: lassila@w3.org
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
Example: **Alternative**

The source code for X11 may be found at ftp.x.org, ftp.cs.purdue.edu, or ftp.eu.net

```
http://x.org/package/X11
  rdf:type  rdf:Alt
    rdf:_1  ftp.x.org
    rdf:_2  ftp.cs.purdue.edu
    rdf:_3  ftp.eu.net
```
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
Example: Reification

Ralph Swick believes that

- the creator of the resource http://www.w3.org/Home/Lassila is Ora Lassila
RDF Syntax I

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

http://www.w3.org/Home/Lassila

s:Creator

Ora Lassila

s:createdWith

http://www.w3c.org/amaya

 RDF
<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>
RDF Syntax II: Syntactic Varieties

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>

Property

http://www.w3.org/Home/Lassila

rdf:type
s:Homepage

s:Creator
Ora Lassila

s:createdWith
http://www.w3c.org/amaya

rdf:type
HTMLEditor
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

xyz:MotorVehicle

xyz:Truck

xyz:Van

xyz:PassengerVehicle

xyz:MiniVan

rdfs:Resource

rdfs:Class

t = rdf:type
Rdfs:subclassOf

<rdf:Description about="Xyz:Minivan">
  <rdfs:subClassOf about="xyz:Van"/>
</rdfs:Description>

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

Predicate Logic Consequences:

Forall X: type(X,MiniVan) -> type(X,Van).
Forall X: subclassOf(X,MiniVan) -> subclassOf(X,Van).
<rdf:description about="possesses">
    <rdf:type about="...property"/>
    <rdfs:domain about="person"/>
    <rdfs:range about="vehicle"/>
</rdf:description>

<predicate logic consequences:
For all X, Y: possesses (X, Y) -> (type(X, person) & type(Y, vehicle)).
OWL
OWL – General

W3C Recommendation since 2004
More work on OWL1.1 to come
Semantic fragment of FOL
Three variants:
OWL Lite $\mu$ OWL DL $\mu$ 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 model theoretic 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 \( \lor \) Human \( \land \) Male
  - Happy-Father \( \lor \) Man \( \land \) 9 \( \land \) has-child.Female \( \land \) ...

- **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" ins everything 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 u, t, : (intersection, union, negation)
- Quantors define how roles are to be interpreted:
  - Man u 9hasChild.Female u 9hasChild.Male
  - u 8hasChild.(Rich t Happy)
Number restrictions (cardinality constraints) for roles:
  \( \geq 3 \) hasChild, \( \leq 1 \) hasMother

Qualified number restrictions:
  \( \geq 2 \) hasChild.Female, \( \leq 1 \) hasParent.Male

Nominals (definition by extension):
  \{Italy, France, Spain\}

Concrete domains (datatypes):
  hasAge.(\( \geq 21 \))

Inverse roles:
  hasChild\(^{-}\) \(\leq\) 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 ≡ Man u ∃hasChild.Female u …
  - Elephant v Animal u Large u 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.
### Syntax for DLs (ohne concrete domains)

#### Concepts

<table>
<thead>
<tr>
<th>ALC Definition</th>
<th>ALC Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>A, B</td>
</tr>
<tr>
<td>Not</td>
<td>:C</td>
</tr>
<tr>
<td>And</td>
<td>C ∪ D</td>
</tr>
<tr>
<td>Or</td>
<td>C ∩ D</td>
</tr>
<tr>
<td>Exists</td>
<td>∃ R.C</td>
</tr>
<tr>
<td>For all</td>
<td>∀ R.C</td>
</tr>
<tr>
<td>At least</td>
<td>≥n R.C (≥n R)</td>
</tr>
<tr>
<td>At most</td>
<td>≤n R.C (≤n R)</td>
</tr>
<tr>
<td>Nominal</td>
<td>{i_1, ..., i_n}</td>
</tr>
</tbody>
</table>

#### Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic</td>
<td>R</td>
</tr>
<tr>
<td>Inverse</td>
<td>R^-</td>
</tr>
</tbody>
</table>

#### Ontology (=Knowledge Base)

**Concept Axioms (TBox)**
- Subclass: C ∪ D
- Equivalent: C \(\sim\) D

**Role Axioms (RBox)**
- Subrole: R ∪ S
- Transitivity: Trans(S)

**Assertional Axioms (ABox)**
- Instance: C(a)
- Role: R(a, b)
- Same: a = b
- Different: a \(\neq\) b

\[ S = ALC + Transitivity \]

**OWL DL = SHOIN(D) (D: concrete domain)**
Content

OWL – Syntax and model theoretic 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>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \wedge \ldots \wedge C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1(x) \vee \ldots \vee C_n(x)$</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>${x_1} \sqcap \ldots \sqcap {x_n}$</td>
<td>${john} \sqcap {mary}$</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>$\exists y. P(x, y) \wedge C(y)$</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>$\exists \geq n y. P(x, y)$</td>
</tr>
</tbody>
</table>

Nesting of expression is allowed at arbitrary depth:

Person u 8hasChild.(Doctor t 9hasChild.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 (\cap) Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man (\equiv) Human (\cap) 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>$P_1 \equiv P_2^-$</td>
<td>hasChild (\equiv) hasParent^-</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor(^+) (\sqsubseteq) ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$\top \sqsubseteq \leq 1P$</td>
<td>$\top \sqsubseteq 1\text{hasMother}$</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$\top \sqsubseteq \leq 1P^-$</td>
<td>$\top \sqsubseteq 1\text{hasSSN}^-$</td>
</tr>
</tbody>
</table>

**General Class Inclusion (v):**

$$C \ ' D \ \text{IFF} \ (C \sqcup D \ \text{und} \ D \sqcup C)$$

**Obvious equivalances with FOL:**

$$C \ ' D , \ (8x) \ (C(x) \ n D(x))$$

$$C \sqcup D , \ (8x) \ (C(x) \ ! D(x))$$
Terminological Knowledge (TBox):
Human v 9hasParent.Human
Orphan ⊒ Human u :9childOf.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>Concept</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Subset of $M^\mathcal{I}$</td>
</tr>
<tr>
<td>¬C</td>
<td>$M^\mathcal{I} \cap C^\mathcal{I}$</td>
</tr>
<tr>
<td>C ∪ D</td>
<td>{x</td>
</tr>
<tr>
<td>C ∩ D</td>
<td>{x</td>
</tr>
<tr>
<td>∃R.C</td>
<td>{x</td>
</tr>
<tr>
<td>∀R.C</td>
<td>{x</td>
</tr>
<tr>
<td>≥ n R.C</td>
<td>{x</td>
</tr>
<tr>
<td>≤ n R.C</td>
<td>{x</td>
</tr>
<tr>
<td>{i₁, ..., iₙ}</td>
<td>{i₁^\mathcal{I}, ..., iₙ^\mathcal{I}}</td>
</tr>
</tbody>
</table>

### Role expressions

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

### Concept Axioms (TBox)

- $C \sqcup D$ \quad $C^\mathcal{I} \subseteq D^\mathcal{I}$
- $C \equiv D$ \quad $C^\mathcal{I} \equiv D^\mathcal{I}$

### Role Axioms (rarely: RBox)

- $R \sqcup S$ \quad $R^\mathcal{I} \subseteq S^\mathcal{I}$

### Assertional Axioms (ABox)

- $C(a)$ \quad $a^\mathcal{I} \in C^\mathcal{I}$
- $R(a, b)$ \quad $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$
- $a = b$ \quad $a^\mathcal{I} = b^\mathcal{I}$
- $a \neq b$ \quad $a^\mathcal{I} \neq b^\mathcal{I}$
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!)
OWL Lite

Simple fragment, comparatively low complexity
Some constructors may not be used:
- owl:unionOf
- owl:complementOf
- owl:oneOf
- owl:hasValue
- owl:disjointWith

The applicability of some operators is restricted:
- owl:intersectionOf
- owl:minCardinality
- owl:maxCardinality
- owl:cardinality
OWL Full

equals OWL DL union RDFS
RDF is contained in OWL Full, not in OWL DL

Intuition:
- OWL Full allows reification.
- OWL Full is no “nice” fragment of FOL
- OWL Full ist not decidable
Sometimes one might state a proposition about a class name:

Class: father.
(concrete) Roles:
  germanClassName(father, “Vater“)
  frenchClassName(father, „père“)
  englishClassName(father, „father“)

! Cannot be expressed in OWL DL!
### Complexity (worst-case)

<table>
<thead>
<tr>
<th>OWL variant</th>
<th>Data complexity</th>
<th>Combined complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL Full</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>OWL DL</td>
<td></td>
<td>NExptime</td>
</tr>
<tr>
<td>OWL DL without nominals</td>
<td>NP</td>
<td>Exptime</td>
</tr>
<tr>
<td>OWL Lite</td>
<td>NP</td>
<td>Exptime</td>
</tr>
</tbody>
</table>

Data complexity: assume fixed T-Box, complexity wrt size of ABox

Combined complexity: wrt combined size of ABox and TBox
OWL – Syntax and model theoretic semantics

  a. Description logics: SHOIN(D)
  b. OWL as SHOIN(D)
  c. Serializations
  d. Knowledge modelling in OWL
### Serializations/different Syntaxes

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Standard/Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL RDF Syntax</td>
<td>W3C recommendation</td>
</tr>
<tr>
<td>OWL Abstract Syntax</td>
<td>W3C recommendation, See next section</td>
</tr>
<tr>
<td>OWL XML Syntax</td>
<td>W3C document</td>
</tr>
<tr>
<td>DL Schreibweise</td>
<td>widely used in scientific contexts</td>
</tr>
<tr>
<td>FOL Schreibweise</td>
<td>uncommon</td>
</tr>
</tbody>
</table>

*ISWeb - Information Systems & Semantic Web*

Steffen Staab

staab@uni-koblenz.de
Example: RDF Syntax

Person u 8hasChild.(Doctor t 9hasChild.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: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:unionOf>
         </owl:allValuesFrom>
      </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
Content

OWL – Syntax and model theoretic semantics

a. Description logics: SHOIN(D)
b. OWL as SHOIN(D)
c. Serializations
d. Knowledge modelling in OWL
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))))

`bus_driver ` person u 9drives.bus

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

`driver ` person u 9drives.vehicle

Class(a:bus partial a:vehicle)

`bus v 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))))

\[ \text{driver} \cap \text{person} \cup \text{9drives.vehicle} \]

Class(a:driver partial a:adult)

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

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

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

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: Examples

Class(a:cow partial a:vegetarian)
DisjointClasses(unionOf(restriction(a:part_of someValuesFrom (a:animal)) a:animal)
unionOf(a:plant restriction(a:part_of someValuesFrom (a:plant))))

9 partof.animal t animal ∩ plant t 9partof.plant

Class(a:vegetarian complete intersectionOf( restriction(a:eats allValuesFrom (complementOf(restriction(a:part_of someValuesFrom (a:animal))))
restriction(a:eats allValuesFrom (complementOf(a:animal)) a:animal))
Class(a:mad_cow complete intersectionOf(a:cow restriction(a:eats someValuesFrom (intersectionOf(restriction(a:part_of someValuesFrom (a:sheep)) a:brain))))
Class(a:sheep partial a:animal restriction(a:eats allValuesFrom (a:grass)))

Cows are naturally vegetarians
A mad cow is one that has been eating sheeps brains
Sheep are animals
Thus a mad cow has been eating part of an animal, which is inconsistent with the definition of a vegetarian
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.

<table>
<thead>
<tr>
<th>child(Bill,Bob)</th>
<th>Man(Bob)</th>
<th>≤ 1 child.T(Bill)</th>
</tr>
</thead>
<tbody>
<tr>
<td>? ² ∨ child.Man(Bill)</td>
<td>DL answers</td>
<td>Prolog</td>
</tr>
<tr>
<td>No idea, since we do not know all children of Bill.</td>
<td>don’t know</td>
<td>yes</td>
</tr>
<tr>
<td>If we assume that we know everything about Bill, then all of his children are male.</td>
<td>Now we know everything about Bill’s children.</td>
<td></td>
</tr>
</tbody>
</table>
ObjectProperty(xyz:has_topping
  domain(xyz:Pizza) > v 8has_topping^-..Pizza
  range(xyz:Pizza_topping)) > v 8has_topping.Pizza_topping

Class(xyz:Ice_cream_cone partial
  restriction(xyz:has_topping someValuesFrom (xyz:Ice_cream)))
  Ice_cream_cone v 9has_topping.Ice_cream

If Ice_cream_cone and Pizza not disjunct:
  • Ice_cream_cone is classified as Pizza
  • …but: Ice_cream is not classified as Pizza_topping
  • Consequences: all Ice_cream_cones are Pizzas,
    and some Ice_cream is a Pizza_topping
Knowledge modelling: Some Research Challenges

Concluding with
- uncertainty (fuzzy, probabilistic)
- Inconsistencies (paraconsistent)
- Rules
- Further AI-Paradigms
  (time, space, nonmonotonic reasoning, preferences ...)

Maintenance (updates, infrastructure, etc)
Scalability of reasoning

...
Thank You

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