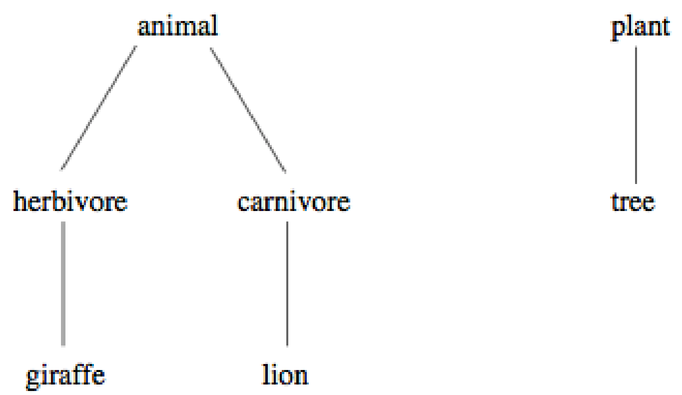
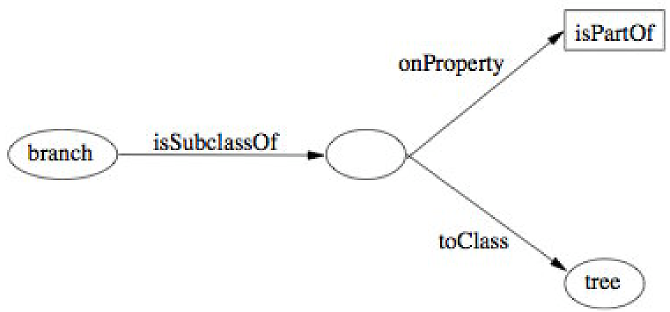
Semi-Automatic Information and Knowledge Systems, Einführung in Semantic Web : Ontology Engineering

Monika Lanzenberger

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors & How to Avoid Them
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook



[Antoniou and van Harmelen, 2004]



[Antoniou and van Harmelen, 2004]

```
<owl:TransitiveProperty rdf:ID="is-part-of"/>

<owl:ObjectProperty rdf:ID="eats">
    <rdfs:domain rdf:resource="#animal"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="eaten-by">
    <owl:inverseOf rdf:resource="#eats"/>
</owl:ObjectProperty>
```



```
<owl:Class rdf:ID="plant">
  <rdfs:comment>Plants are disjoint from animals.
  </rdfs:comment>
  <owl:disjointWith="#animal"/>
</owl:Class>

<owl:Class rdf:ID="tree">
  <rdfs:comment>Trees are a type of plant.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#plant"/>
</owl:Class>
```



```
<owl:Class rdf:ID="branch">
    <rdfs:comment>Branches are parts of trees.</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#is-part-of"/>
          <owl:allValuesFrom rdf:resource="#tree"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```



```
<owl:Class rdf:ID="leaf">
    <rdfs:comment>Leaves are parts of branches. </rdfs:comment>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#is-part-of"/>
         <owl:allValuesFrom rdf:resource="#branch"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

An African Wildlife Ontology – Carnivores 9

```
<owl:Class rdf:ID="carnivore">
    <rdfs:comment>Carnivores are exactly those animals
    that eat also animals.</rdfs:comment>
    <owl:intersectionOf rdf:parsetype="Collection">
        <owl:Class rdf:about="#animal"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#eats"/>
          <owl:someValuesFrom rdf:resource="#animal"/>
       </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

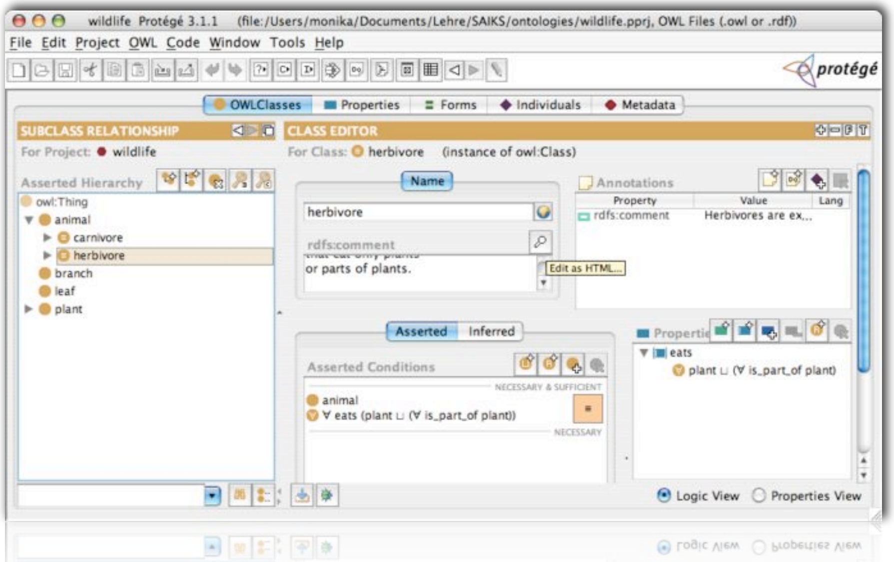

An African Wildlife Ontology – Herbivores 10

```
<owl:Class rdf:ID="herbivore">

  <rdfs:comment>Herbivores are exactly those animals that
  eat only plants or parts of plants.</rdfs:comment>

   ...                                                   ?

</owl:Class>
```


An African Wildlife Ontology – Herbivores 11

```
<owl:intersectionOf rdf:parseType="Collection">                    !
  <owl:Class rdf:about="#animal"/>
  <owl:Restriction>
     <owl:onProperty rdf:resource="#eats"/>
     <owl:allValuesFrom>
        <owl:Class>
           <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#plant"/>
              <owl:Restriction>
                <owl:onProperty rdf:resource="#is_part_of"/>
                <owl:allValuesFrom rdf:resource="#plant"/>
              </owl:Restriction>
           </owl:unionOf>
        </owl:Class>
     </owl:allValuesFrom>
  </owl:Restriction>
</owl:intersectionOf>
```


An African Wildlife Ontology – Herbivores 12



An African Wildlife Ontology – Giraffes


```
<owl:Class rdf:ID="giraffe">
    <rdfs:comment>Giraffes are herbivores, and they
    eat only leaves.</rdfs:comment>
    <rdfs:subClassOf rdf:type="#herbivore"/>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#eats"/>
         <owl:allValuesFrom rdf:resource="#leaf"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```


An African Wildlife Ontology – Lions


```
<owl:Class rdf:ID="lion">
    <rdfs:comment>Lions are animals that eat
    herbivores.</rdfs:comment>
    <rdfs:subClassOf rdf:type="#animal"/>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#eats"/>
         <owl:someValuesFrom rdf:resource="#herbivore"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

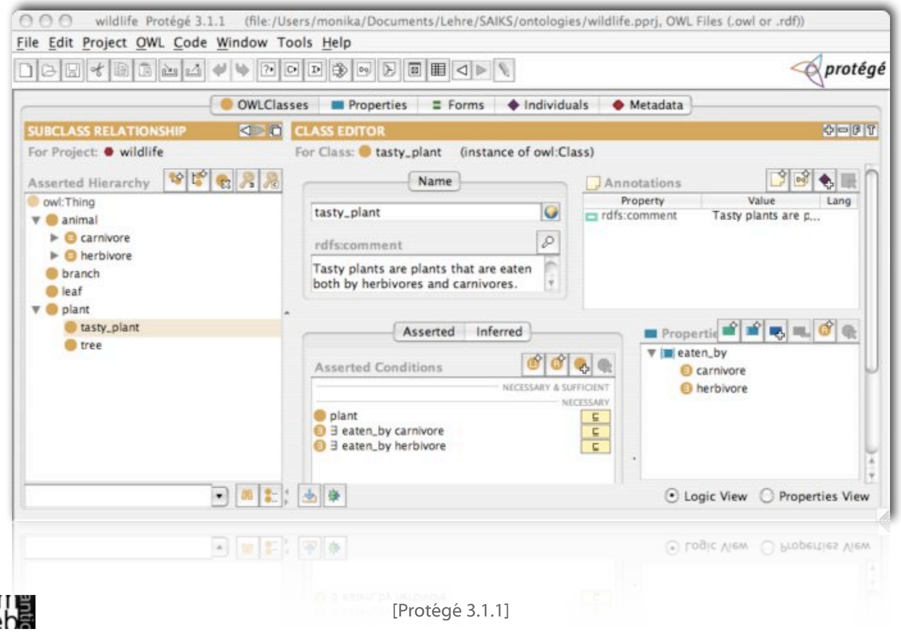

An African Wildlife Ontology – Tasty Plants


```
<owl:Class rdf:ID="tasty-plant">
  <rdfs:comment>Plants eaten both by herbivores and
  carnivores </rdfs:comment>

      ...                                          ?

</owl:Class>
```


An African Wildlife Ontology – Tasty Plants


```
<rdfs:subClassOf rdf:resource="#plant"/>                     !
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#eaten_by"/>
    <owl:someValuesFrom>
      <owl:Class rdf:about="#herbivore"/>
    </owl:someValuesFrom>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#eaten_by"/>
    <owl:someValuesFrom>
      <owl:Class rdf:about="#carnivore"/>
    </owl:someValuesFrom>
  </owl:Restriction>
</rdfs:subClassOf>
```

An African Wildlife Ontology – Tasty Plants



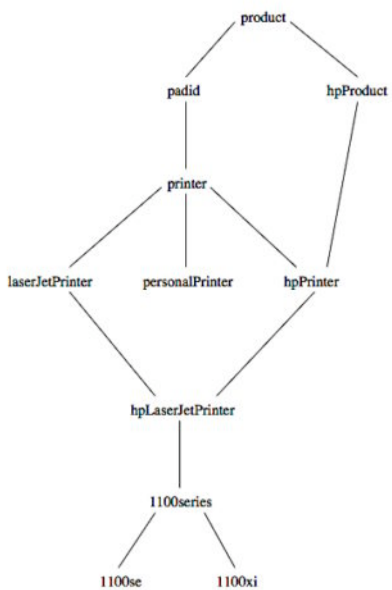
[Protégé 3.1.1]

An African Wildlife Ontology

What problem would emerge if we replace
`owl:someValuesFrom` by
`owl:allValuesFrom`
in the definition of carnivores?

[Antoniou and van Harmelen, 2004]

A Printer Ontology: Class Hierarchy



[Antoniou and van Harmelen, 2004]

A Printer Ontology: Products and Devices


```
<owl:Class rdf:ID="product">
    <rdfs:comment>Products form a class. </rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="padid">
    <rdfs:comment>Printing and digital imaging devices
    form a subclass of products.</rdfs:comment>
    <rdfs:label>Device</rdfs:label>
    <rdfs:subClassOf rdf:resource="#product"/>
</owl:Class>
```

[Antoniou and van Harmelen, 2004]

A Printer Ontology: Properties


```
<owl:DatatypeProperty rdf:ID="manufactured-by">
    <rdfs:domain rdf:resource="#product"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="printingTechnology">
    <rdfs:domain rdf:resource="#printer"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```


A Printer Ontology: HP Products


```
<owl:Class rdf:ID="hpProduct">
    <owl:intersectionOf>
       <owl:Class rdf:about="#product"/>
       <owl:Restriction>
          <owl:onProperty rdf:resource="#manufactured-by"/>
          <owl:hasValue>
            <xsd:string rdf:value="Hewlett Packard"/>
          </owl:hasValue>
       </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

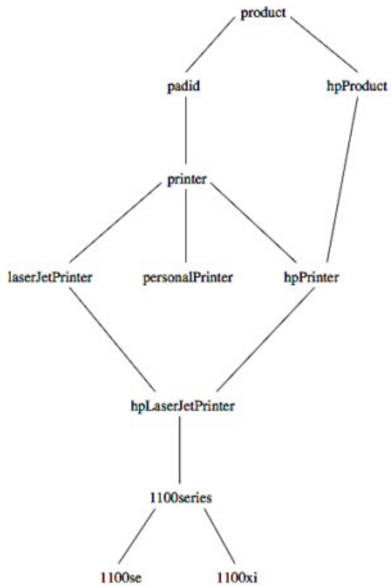

A Printer Ontology: Printers and Personal Printers


```
<owl:Class rdf:ID="printer">
    <rdfs:comment>Printers are printing and digital imaging
    devices.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#padid"/>
</owl:Class>
<owl:Class rdf:ID="personalPrinter">
    <rdfs:comment>Printers for personal use form
    a subclass of printers.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#printer"/>
</owl:Class>
```


A Printer Ontology: HP LaserJet 1100se Printers


```
<owl:Class rdf:ID="1100se">
    <rdfs:comment>1100se printers belong to the 1100 series
       and cost $450.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#1100series"/>
    <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#price"/>
         <owl:hasValue><xsd:integer rdf:value="450"/>
         </owl:hasValue>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

A Printer Ontology: Class Hierarchy



[Antoniou and van Harmelen, 2004]

Outline

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors & How to Avoid Them
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook

Future Extensions of OWL

... Modules and Imports
... Defaults
... Closed World Assumption
... Unique Names Assumption
... Procedural Attachments
... Rules for Property Chaining

[Antoniou and van Harmelen, 2004]

Modules and Imports

- The importing facility of OWL is very trivial:
  It only allows importing of an entire ontology, not parts of it.
- Modules in programming languages based on information hiding (state functionality, hide implementation details):
  Open question how to define appropriate module mechanism for Web ontology languages.

[Antoniou and van Harmelen, 2004]

Defaults

- Many practical knowledge representation systems allow inherited values to be overridden by more specific classes in the hierarchy.
  (Treat inherited values as defaults.)
- No consensus has been reached on the right formalization for the nonmonotonic behaviour of default values.

Closed World Assumption

- OWL currently adopts the open-world assumption:
  A statement cannot be assumed true on the basis of a failure to prove it. On the huge and only partially knowable WWW, this is a correct assumption.
- Closed-world assumption: a statement is true when its negation cannot be proved:
  tied to the notion of defaults, leads to nonmonotonic behaviour.

Unique Names Assumption

- Typical database applications assume that individuals with different names are indeed different individuals.
- OWL follows the usual logical paradigm where this is not the case. (Plausible on the WWW.)
- One may want to indicate portions of the ontology for which the assumption does or does not hold.

Procedural Attachments

- A common concept in knowledge representation is to define the meaning of a term by attaching a piece of code to be executed for computing the meaning of the term, instead of through explicit definitions in the language.
- Although widely used, this concept does not lend itself very well to integration in a system with a formal semantics, and it has not been included in OWL.

Rules for Property Chaining

- OWL does not allow the composition of properties for reasons of decidability.
- Integration of rule-based knowledge representation and DL-style knowledge representation is currently an active area. (E.g., W3C's Rule Interchange Format Working Group)

Outline

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook

Main Stages in Ontology Development

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- Define properties
- Define constraints
- Create instances
- Check for anomalies

Not a linear process!

Determine Scope

- There is no correct ontology of a specific domain:
  An ontology is an abstraction of a particular domain, and there are always viable alternatives.
- What is included in this abstraction should be determined by ...
  ... the use to which the ontology will be put.
  ... by future extensions that are already anticipated.

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- Define properties
- Define constraints
- Create instances
- Check for anomalies

Determine Scope (2)

Basic questions to be answered at this stage are:

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions should the ontology provide answers?
- Who will use and maintain the ontology?

Consider Reuse

- With the spreading deployment of the Semantic Web, ontologies will become more widely available.
- We rarely have to start from scratch when defining an ontology.
  There is almost always an ontology available from a third party that provides at least a useful starting point for our own ontology.

Enumerate Terms

Write down in an unstructured list all the relevant terms that are expected to appear in the ontology:

- Nouns form the basis for class names.
- Verbs (or verb phrases) form the basis for property names.

Traditional knowledge engineering tools can be used to obtain:

- the set of terms.
- an initial structure for these terms.

Define Classes and a Taxonomy

- Relevant terms must be organized in a taxonomic hierarchy. Opinions differ on whether it is more efficient/reliable to do this in a top-down or a bottom-up fashion.
- Ensure that hierarchy is indeed a taxonomy:
  If `A` is a subclass of `B`,
  then every instance of `A`
  must also be an instance of `B`.

Define Properties

- Often interleaved with the previous step.

- The semantics of `subClassOf` demands that whenever `A` is a subclass of `B`, every property statement that holds for instances of `B` must also apply to instances of `A`:
  It makes sense to attach properties to the highest class in the hierarchy to which they apply.

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- **Define properties**
- Define constraints
- Create instances
- Check for anomalies

Define Properties (2)

While attaching properties to classes, it makes sense to immediately provide statements about the domain and range of these properties.

There is a methodological tension here between generality and specificity:

- Flexibility (inheritance to subclasses)
- Detection of inconsistencies and misconceptions

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- **Define properties**
- Define constraints
- Create instances
- Check for anomalies

Define Constraints

Cardinality restrictions

Required values:

- `owl:hasValue`
- `owl:allValuesFrom`
- `owl:someValuesFrom`

Relational characteristics:

- symmetry
- transitivity
- inverse properties
- functional values

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- Define properties
- **Define constraints**
- Create instances
- Check for anomalies

Create Instances

- Filling the ontologies with such instances is a separate step.

- Number of instances >> number of classes

- Thus populating an ontology with instances is not done manually:
  ... retrieved from legacy data sources.
  ... extracted automatically from a text corpus.

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- Define properties
- Define constraints
- **Create instances**
- Check for anomalies

Check for Anomalies

An important advantage of the use of OWL over RDF Schema is the possibility to detect inconsistencies in ontology and instances.

Examples of common inconsistencies:
...incompatible domain and range definitions for transitive, symmetric, or inverse properties;
...cardinality properties;
...requirements on property values can conflict with domain and range restrictions.

- Determine scope
- Consider reuse
- Enumerate terms
- Define classes and a taxonomy
- Define properties
- Define constraints
- Create instances
- Check for anomalies

Outline

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors & How to Avoid Them
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook

Common Errors

- Failure to make all information explicit, assuming that information implicit in names is "represented" and available to the classifier.
- Mistaken use of universal rather than existential restrictions as the default.
- Open world reasoning.
- The effect of range and domain constraints as axioms.

Common Errors (2)

- Trivial satisfiability of universal restrictions – that "only" (`allValuesFrom`) does not imply "some" (`someValuesFrom`).
- The difference between defined and primitive classes and the mechanics of converting one to the other.
- Errors in understanding common logical constructs.
- Expecting classes to be disjoint by default.
- The difficulty of understanding subclass axioms used for implication.

Guidelines

- Always paraphrase a description or definition before encoding it in OWL, and record the paraphrase in the comment area of the interface.
- Make all primitives disjoint - which requires that primitives form trees.
- Use `someValuesFrom` as the default qualifier in restrictions .
- Be careful to make defined classes defined – the default is primitive.

Guidelines (2)

- Remember the open world assumption. Insert closure restrictions if that is what you mean.

- Be careful with domain and range constraints. Check them carefully if classification does not work as expected.

- Be careful about the use of "and" and "or" (`intersectionOf`, `unionOf`).

Guidelines (2)

- To spot trivially satisfiable restrictions early, always have an existential (`someValuesFrom`) restriction corresponding to every universal (`allValuesFrom`) restriction, either in the class or one of its superclasses (unless you specifically intend the class to be trivially satisfiable).

- Run the classifier frequently; spot errors early.

Outline

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors & How to Avoid Them
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook

Existing Domain-Specific Ontologies

- Medical domain:
  Cancer ontology from the
  National Cancer Institute in the United States
  http://www.mindswap.org/2003/CancerOntology

- Geographical domain:
  Getty Thesaurus of Geographic Names (TGN),
  containing over 1 million entries
  http://www.getty.edu/research/conducting_research/vocabularies/tgn

Existing Domain-Specific Ontologies (2)

Cultural domain:

- Art and Architecture Thesaurus (AAT)
  with 125,000 terms in the cultural domain
  http://www.getty.edu/research/vocabulary/aat

- Union List of Artist Names (ULAN)
  with 220,000 entries on artists
  http://www.getty.edu/research/conducting_research/vocabulary/ulan

- Iconclass vocabulary of 28,000 terms for describing images
  http://www.iconclass.nl

Integrated Vocabularies

- Merge independently developed vocabularies
  into a single large resource.

- E.g. Unified Medical Language System
  integrating 100 biomedical vocabularies
  The UMLS metathesaurus contains 750,000 concepts,
  with over 10 million links between them.
  http://umlsinfo.nlm.nih.gov

- The semantics of a resource that integrates many
  independently developed vocabularies is rather low.
  But very useful in many applications as starting point.

Upper-Level Ontologies

Some attempts have been made to define very generally applicable ontologies. (Not domain-specific)

- Cyc with 60,000 assertions on 6,000 concepts
  http://www.opencyc.org
- Standard Upperlevel Ontology (SUO)
  http://suo.ieee.org
- Basic Formal Ontology (BFO): series of sub-ontologies
  http://ontology.buffalo.edu/bfo/BFO.html
- Dolce
  http://www.loa-cnr.it/DOLCE.html
- General Formal Ontology (GFO)
  http://www.onto-med.de/en/theories/gfo/index.html

Topic Hierarchies

- Some "ontologies" do not deserve this name: simply sets of terms, loosely organized in a hierarchy.
- This hierarchy is typically not a strict taxonomy but rather mixes different specialization relations (e.g., is-a, part-of, contained-in).
- Such resources often very useful as starting point.
- Example: Open Directory hierarchy, containing more then 400,000 hierarchically organized categories.
  http://dmoz.org

Linguistic Resources

- Some resources were originally built not as abstractions of a particular domain, but rather as linguistic resources.
- These have been shown to be useful as starting places for ontology development.
  E.g. ,WordNet, with over 90,000 word senses.
  http://www.cogsci.princetonedu/~wn

Ontology Libraries

Attempts are currently underway to construct online libraries of online ontologies.

- Rarely existing ontologies can be reused without changes.
- Existing concepts and properties must be refined using `rdfs:subClassOf` and `rdfs:subPropertyOf` .
- Alternative names must be introduced which are better suited to the particular domain using `owl:equivalentClass` and `owl:equivalentProperty`.
- We can exploit the fact that RDF and OWL allow private refinements of classes defined in other ontologies.

Outline

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors & How to Avoid Them
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook

Fundamental Research Challenges 61

Title:	**Fundamental Research Challenges Generated by the Semantic Web**
Author:	**Frank van Harmelen** (More resources of Frank van Harmelen)
General Information	
Provider:	Holger Wache (Vrije Universiteit Amsterdam)
Learning Resource Language:	English
Description Language:	English
Description:	A 1 hour video about the research challenges in Semantic Web
Classification:	Ontologies for the Semantic Web, Knowledge Representation and Reasoning, Knowledge Engineering / Ontology Engineering, Semantic Web Applications
Learning Resource Type:	Educational Material, Presentation

Outline 63

- Some OWL Examples
- Future Extensions
- Constructing Ontologies Manually
- Common Errors & How to Avoid Them
- Reusing Existing Ontologies
- Fundamental Research Challenges
- Outlook

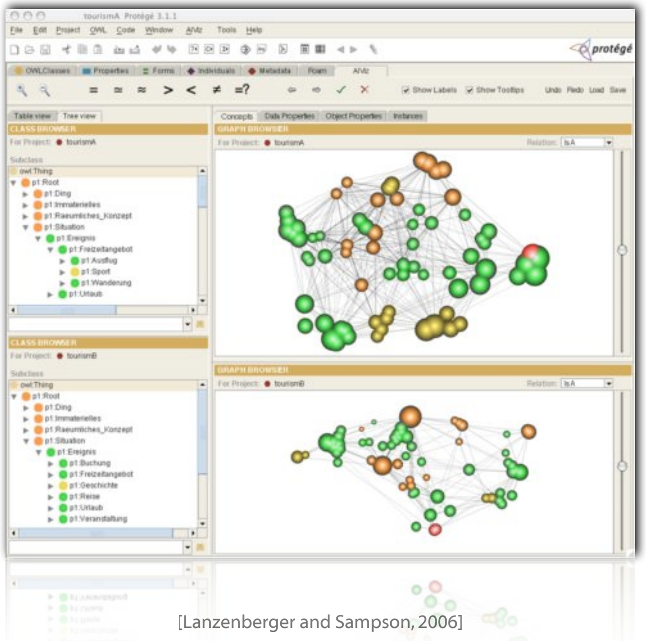
Semi-Automatic Information and Knowledge Systems 64

13.12.2006 Ontology Merging & Integration
15.12.2006 Ontology Mapping & Alignment
10.01.2007 Ontology Re-use: Lessons Learned & Current Challenges
12.01.2007 Hierarchical Data Visualization Techniques
17.01.2007 Ontology Visualization & Semi-automatic Alignment

19.01.2007 Referate (6 x je 15 Min)
24.01.2007 Referate (6 x je 15 Min)

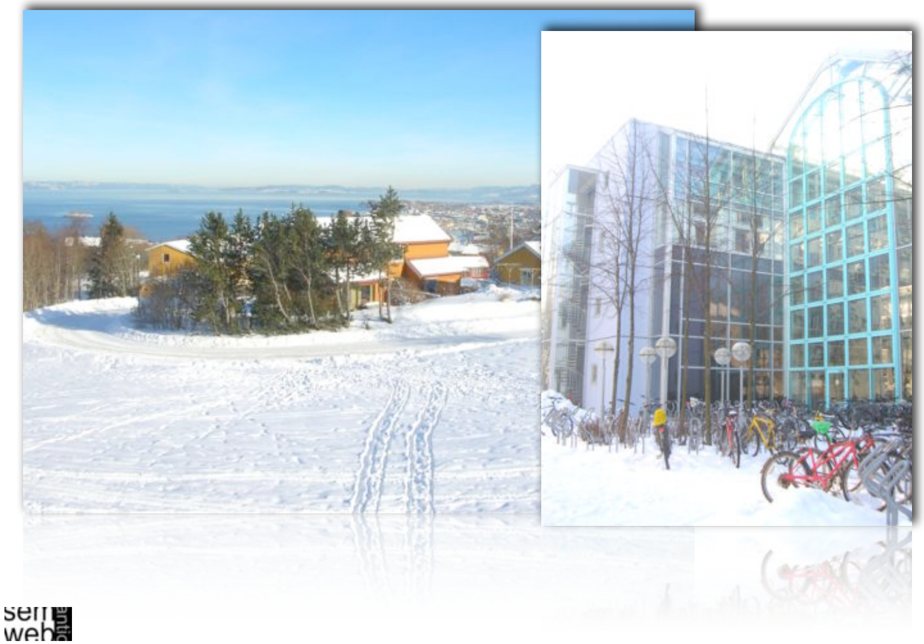
26.01.2007 Prüfung

Current InfoVis Research Activities: AlViz



[Lanzenberger and Sampson, 2006]

AlViz & Trondheim



References & Resources

[Antoniou and van Harmelen, 2004] Grigoris Antoniou and Frank van Harmelen, A Semantic Web Primer, MIT Press, Massachusetts, 2004.

[Noy and McGuinness] Ontology Development 101: A Guide to Creating Your First Ontology Natalya. F. Noy and Deborah L. McGuinness, http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html

[REASE] REASE- the EASE repository for learning units, http://ubp.l3s.uni-hannover.de/ubp/baseapp@home (checked online 8. Nov. 2006).

[Lanzenberger and Sampson, 2006] Monika Lanzenberger and Jennifer Sampson, AlViz - A Tool for Visual Ontology Alignment, In: Proceedings of the IV06, 10th International Conference on Information Visualisation, July 5-7, 2006, London, UK, IEEE Computer Science Society, 2006.

[Protégé 3.1.1] Stanford Medical Informatics, Protégé-OWL, http://protege.stanford.edu/download/download.html (checked online 8. Nov. 2006).

[Rector, et al., 2004] Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe, OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns, http://www.co-ode.org/resources/papers/ekaw2004.pdf (checked online 8. Nov. 2006).

Thanks to ...

... Grigoris Antoniou and
... Frank van Harmelen

for making nice slides of their presentations available.