Semi-Automatic Information and Knowledge Systems
:
OWL - Ontology Web Language
&
Ontology Engineering

Monika Lanzenberger

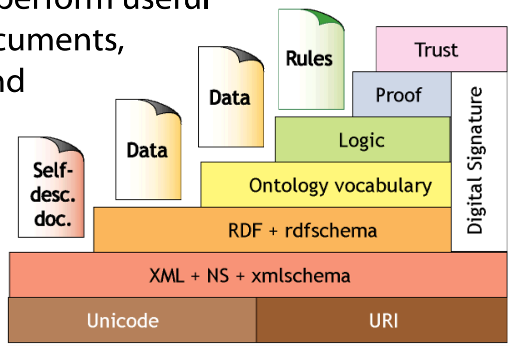
Outline

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

Why OWL?

The Semantic Web is a vision for the future of the Web [...] information is given explicit meaning, [...] machines automatically process and integrate information available on the Web.

If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema.



[W3Ca, Miller]

Requirements for Ontology Languages

Ontology languages allow users to write explicit, formal conceptualizations of domain models.

The main requirements are:

- a well-defined syntax
- efficient reasoning support
- a formal semantics
- sufficient expressive power
- convenience of expression

[Antoniou and van Harmelen, 2004]

- The richer the language is, the more inefficient the reasoning support becomes.
- Sometimes it crosses the border of noncomputability.
- We need a compromise:
  A language supported by reasonably efficient reasoners.
  A language that can express large classes of ontologies and knowledge.

- Class membership
  If x is an instance of a class C,
  and C is a subclass of D,
  then we can infer that x is an instance of D.
- Equivalence of classes
  If class A is equivalent to class B,
  and class B is equivalent to class C,
  then A is equivalent to C, too.

- Consistency
  Consider x being an instance of classes A and B,
  but A and B are disjoint.
  --> Indication of an error in the ontology.
- Classification
  Certain property-value pairs are a sufficient condition for membership in a class A; if an individual x satisfies such conditions, we can conclude that x must be an instance of A.

Reasoning support is important for...
... checking the consistency of the ontology and the knowledge.
... checking for unintended relationships between classes.
... automatically classifying instances in classes.

Checks like the preceding ones are valuable for...
... designing large ontologies, where multiple authors are involved.
... integrating and sharing ontologies from various sources.

Reasoning Support for OWL

- Semantics is a prerequisite for reasoning support
- Formal semantics and reasoning support are usually provided by...
  ... mapping an ontology language to a known logical formalism.
  ... using automated reasoners that already exist for those formalisms.
- OWL is (partially) mapped on a description logic, and makes use of reasoners such as FaCT++, RacerPro, Pellet.
- Description logics are a subset of predicate logic for which efficient reasoning support is possible.

Limitations of Expressive Power of RDF Schema

Local scope of properties

- rdfs:range defines the range of a property (e.g. eats) for all classes .
- In RDF Schema we cannot declare range restrictions that apply to some classes only.
- E.g. , we cannot say that cows eat only plants, while other animals may eat meat, too.

Limitations of Expressive Power of RDF Schema

Disjointness of classes:

- Sometimes we wish to say that classes are disjoint (e.g., child and adult).

Boolean combinations of classes:

- Sometimes we wish to build new classes by combining other classes using union, intersection, and complement.
- E.g., human is the disjoint union of the classes child and adult.

Limitations of Expressive Power of RDF Schema

Cardinality restrictions:

- E.g., a person has exactly two parents, a course is taught by at least one lecturer.

Special characteristics of properties:

- Transitive property (like "greater than")
- Unique property (like "has postcode")
- A property is the inverse of another property (like "eats" and "is eaten by").

Combining OWL with RDF Schema

- Ideally, OWL would extend RDF Schema, consistent with the layered architecture of the Semantic Web.
- But simply extending RDF Schema would work against obtaining expressive power and efficient reasoning:
  - Combining RDF Schema with logic leads to uncontrollable computational properties.
  - Restrictions are required.
- Three Species of OWL defined by the W3C's Web Ontology Working Group.

OWL Sublanguages: Lite & DL

OWL Lite ...

... for classification hierarchies with simple constraints,

... supports cardinality constraints, (only 0 or 1),

... simpler to provide tool support,

... provides a quick migration path for thesauri and other taxonomies,

... has a lower formal complexity than OWL DL.

... restricted: excludes for instance disjointness statements and enumerated classes.

OWL DL ...

... offers maximum expressiveness while retaining computational completeness and decidability.

... includes all OWL language constructs, used under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).

OWL Sublanguages: Full

OWL Full ...

... offers maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual.

... allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary.

... is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

... is fully compatible with RDF (syntactially and semantically) and can be viewed as an extension of RDF, while OWL Lite and OWL DL can be seen as extensions of a restricted view of RDF: Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be a legal OWL Lite or OWL DL document.

OWL Sublanguages

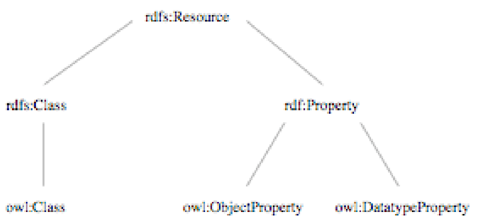
Each of these sublanguages is an extension of its predecessor, both in what can be legally expressed and in what can be validly concluded.

The following set of relations hold:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.
- Their inverses do not!

OWL Compatibility with RDF Schema

- All varieties of OWL use RDF for their syntax
- Instances are declared as in RDF, using RDF descriptions
- and typing information OWL constructors are specialisations of their RDF counterparts



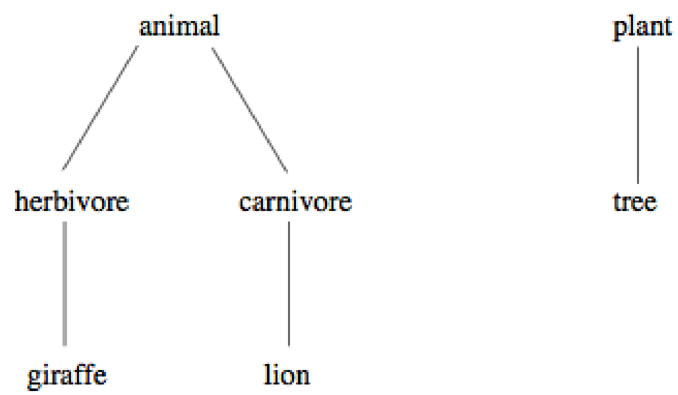
Summary: Why OWL?

- XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.
- XML Schema is a language for restricting the structure of XML documents and also extends XML with data types.
- RDF is a data model for objects ("resources") and relations between them, provides a simple semantics for this data model, and these data models can be represented in an XML syntax.
- RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.
- OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

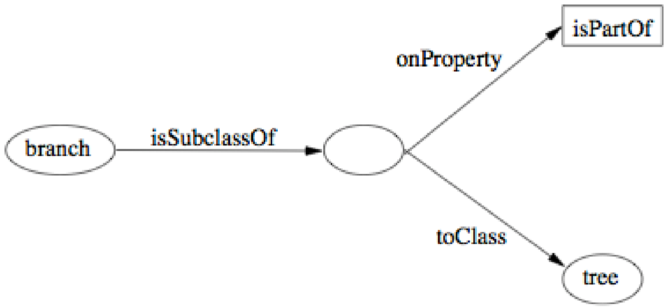
Outline

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

An African Wildlife Ontology – Class Hierarchy



An African Wildlife Ontology – Schematic Representation



[Antoniou and van Harmelen, 2004]

An African Wildlife Ontology – Properties


```
<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>
```

[Antoniou and van Harmelen, 2004]

An African Wildlife Ontology – Plants and Trees


```
<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>
```

[Antoniou and van Harmelen, 2004]

An African Wildlife Ontology – Branches


```
<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>
```

[Antoniou and van Harmelen, 2004]

An African Wildlife Ontology – Leaves


```
<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

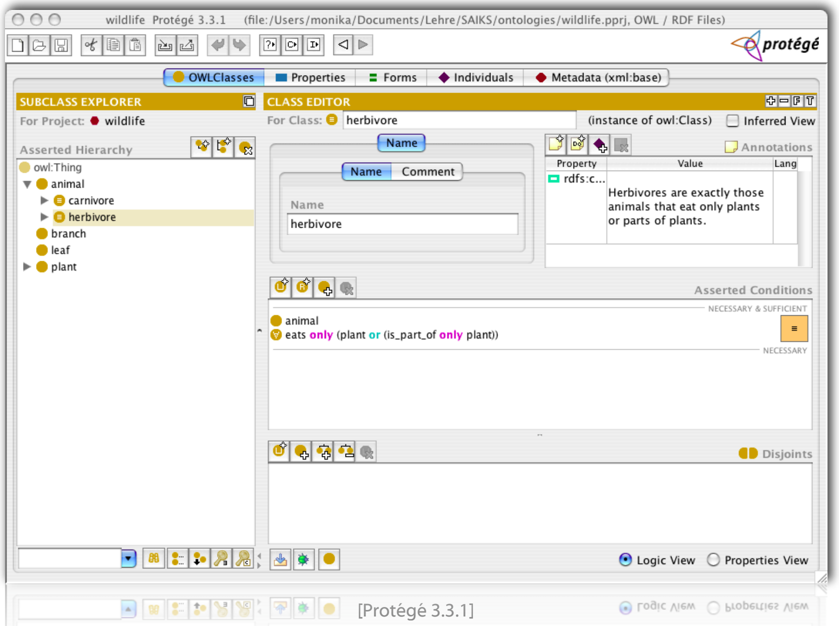

```
<owl:Class rdf:ID="carnivore">
    <rdfs:comment>Carnivores are exactly those animals
    that eat also animals.</rdfs:comment>
    <owl:intersectionOf rdf:parsetype="Collection">
        <owl:Class rdf:about="#animal"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#eats"/>
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


An African Wildlife Ontology – Herbivores


```
<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


```
<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>
```



[Protégé 3.3.1]


```
<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>
```

[Antoniou and van Harmelen, 2004]


```
<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>
```

[Antoniou and van Harmelen, 2004]


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

  ...                                              ?

</owl:Class>
```

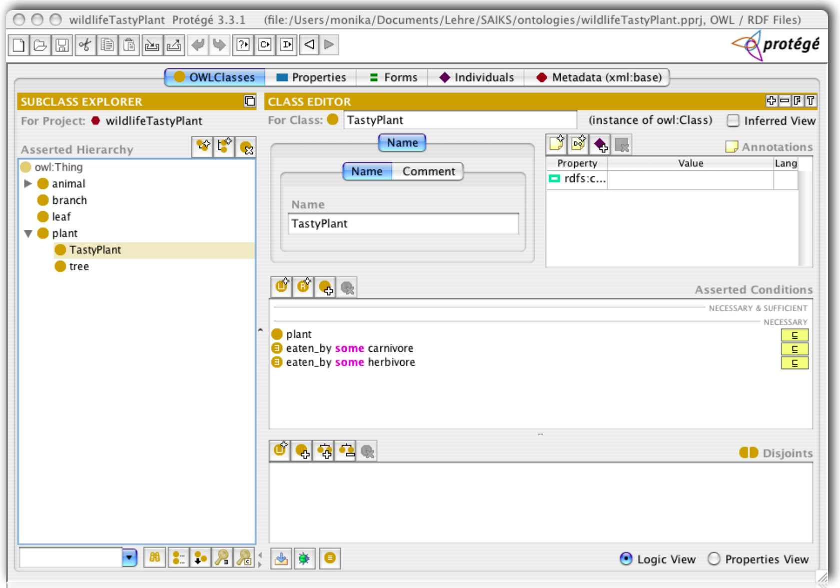
[Antoniou and van Harmelen, 2004]

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>
```

!

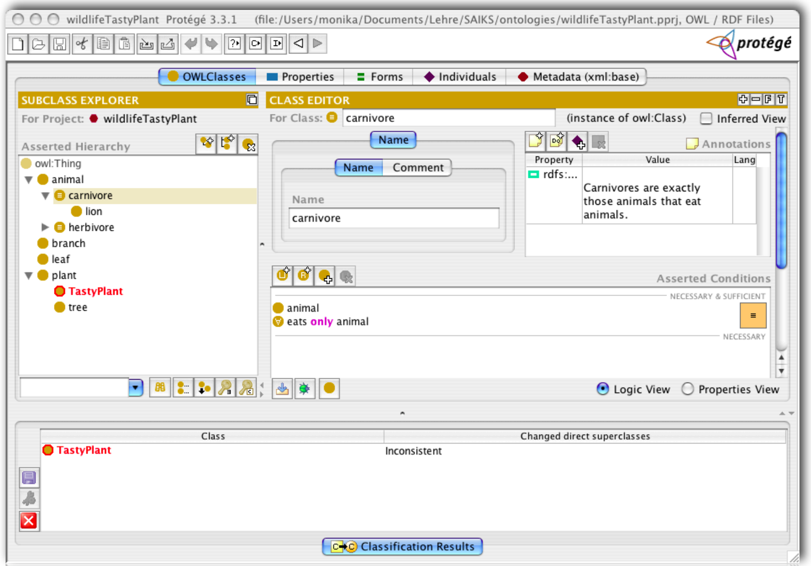
An African Wildlife Ontology – Tasty Plants



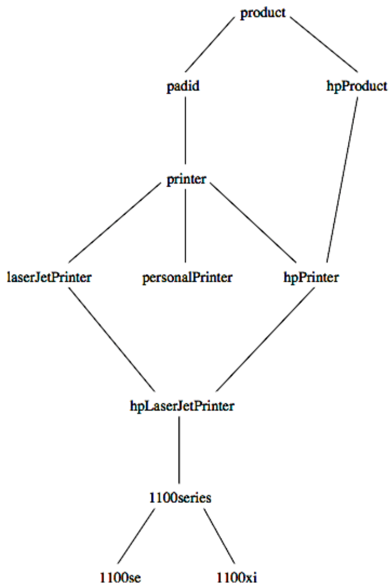
An African Wildlife Ontology

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

An African Wildlife Ontology – Tasty Plants



A Printer Ontology: Class Hierarchy



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>
```


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>
```

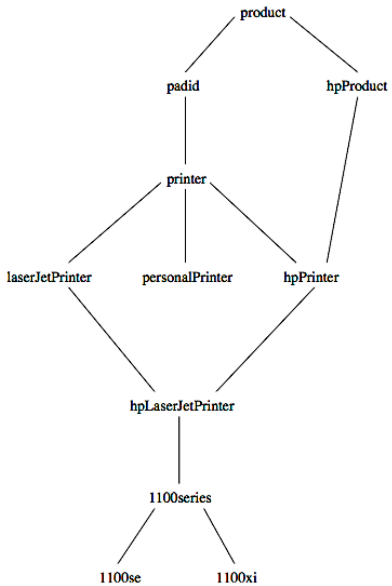
A Printer Ontology: Printers and Personal Printers


```
<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



Outline

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

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

- 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.

- 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.

- 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

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

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

Not a linear process!

- 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

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?

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

- 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.

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

Enumerate Terms

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

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

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

- 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

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

- 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.

Define Properties (2)

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

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

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

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

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.

[Rector, et al., 2004]

Outline

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 around 1 million entries
  http://www.getty.edu/research/conducting_research/vocabularies/tgn

[Antoniou and van Harmelen, 2004]

Existing Domain-Specific Ontologies (2)

Cultural domain:

- Art and Architecture Thesaurus (AAT)
  with 131,000 terms in the cultural domain
  http://www.getty.edu/research/conducting_research/vocabularies/aat

- Union List of Artist Names (ULAN)
  with 293,000 names and biographical and bibliographic information about artists and architects
  http://www.getty.edu/research/conducting_research/vocabularies/ulan

[http://www.getty.edu]

- 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.

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

- 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 4,830,584 sites hierarchically organized in over 590,000 categories.
  http://dmoz.org

- 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 around 150,000 word senses.
  http://wordnet.princeton.edu

- Protégé Ontology Library
Links over 80 Ontologies
http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library

- Taxonomy Warehouse ... available free to users and vocabulary publishers to help organizations maximize their information assets and break through today's information overload:
  Links over 670 Taxonomies
  Classified by 73 subject domains
  Produced by 288 publishers
  39 languages

http://www.taxonomywarehouse.com

Taxonomy Warehouse

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

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

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

[Miller] Miller, Eric: W3C Layer Cake, http://www.w3.org/2001/09/06-ecdl/slide17-0.html (checked online 6. Jan. 2008)

[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 (checked online 6. Jan. 2008).

[REASE] REASE- the EASE repository for learning units, http://ubp.l3s.uni-hannover.de/ubp/baseapp@home (checked online 6. Jan. 2008).

[Protégé 3.3.1] Stanford Medical Informatics, Protégé-OWL, http://protege.stanford.edu/download/download.html (checked online 6. Jan. 2008).

[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 6. Jan. 2008).

[W3Ca] OWL Web Ontology Language Overview, http://www.w3.org/TR/2004/REC-owl-features-20040210/, W3C Recommendation 10 February 2004, (checked online 6. Jan. 2008)

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

for making nice slides of their presentations available.