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

Monika Lanzenberger
• Basic Ideas of OWL
• The OWL Language
  OWL Lite: Simple Classes and Individuals
  OWL Lite: Property Characteristics and Restrictions
  OWL Lite: Constructs
  OWL DL: Complex Classes
• Summary
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.
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 \( \mathbf{x} \) is an instance of a class \( \mathbf{C} \),
  and \( \mathbf{C} \) is a subclass of \( \mathbf{D} \),
  then we can infer that \( \mathbf{x} \) is an instance of \( \mathbf{D} \).

• Equivalence of classes

  If class \( \mathbf{A} \) is equivalent to class \( \mathbf{B} \),
  and class \( \mathbf{B} \) is equivalent to class \( \mathbf{C} \),
  then \( \mathbf{A} \) is equivalent to \( \mathbf{C} \), 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 $A$.

[Antoniou and van Harmelen, 2004]
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.
• 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, RACER, Pellet.
• Description logics are a subset of predicate logic for which efficient reasoning support is possible.

[Antoniou and van Harmelen, 2004]
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.
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.
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”).

[Antoniou and van Harmelen, 2004]
• 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.

[Antoniou and van Harmelen, 2004]
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 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 (syntactically 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.
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!
• 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

[Antoniou and van Harmelen, 2004]
• 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
• The OWL Language
  - OWL Lite: Simple Classes and Individuals
  - OWL Lite: Property Characteristics and Restrictions
  - OWL Lite: Constructs
  - OWL DL: Complex Classes
• Summary
OWL Lite Constructs: Simple Classes and Individuals

Simple Named Classes:

Class

\texttt{rdfs:subClassOf}

Individual

Defining Properties:

\texttt{rdf:Property}

\texttt{subproperties:}

\texttt{owl:ObjectProperty}

(Instance - Instance)

\texttt{owl:DatatypeProperty}

(Instance - \texttt{rdfs:Literal} / XML Schema datatypes)

\texttt{rdfs:subPropertyOf}

\texttt{rdfs:domain}

\texttt{rdfs:range}

Properties of Individuals

\begin{verbatim}
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
\end{verbatim}
Simple Named Classes:

Class
rdfs:subClassOf

Individual

Defining Properties:

rdf:Property
  subproperties:
  owl:ObjectProperty
    (Instance - Instance)
  owl:DatatypeProperty
    (Instance - rdfs:Literal /
     XML Schema datatypes)

rdfs:subPropertyOf
rdfs:domain
rdfs:range

Properties of Individuals
Simple Named Classes:

- **Class**
  - `rdfs:subClassOf`

**Individual**

Defining Properties:

- `rdf:Property`
  - subproperties:
    - `owl:ObjectProperty`
      - (Instance - Instance)
    - `owl:DatatypeProperty`
      - (Instance - `rdfs:Literal` / XML Schema datatypes)

- `rdfs:subPropertyOf`
- `rdfs:domain`
- `rdfs:range`

Properties of Individuals
Simple Named Classes:

- **Class**
- rdfs:subClassOf

**Individual**

Defining Properties:

- rdf:Property
  - subproperties:
    - owl:ObjectProperty (Instance - Instance)
    - owl:DatatypeProperty (Instance - rdfs:Literal / XML Schema datatypes)
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range

Properties of Individuals

<Region rdf:ID="CentralCoastRegion" />

<owl:Thing rdf:ID="CentralCoastRegion" />
<owl:Thing rdf:about="#CentralCoastRegion">
  <rdf:type rdf:resource="#Region"/>
</owl:Thing>
OWL Lite Constructs: Simple Classes and Individuals

Simple Named Classes:
- Class
  - rdfs:subClassOf

Individual

Defining Properties:
- rdf:Property
  - subproperties:
    - owl:ObjectProperty
      (Instance - Instance)
    - owl:DatatypeProperty
      (Instance - rdfs:Literal / XML Schema datatypes)
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range

Properties of Individuals

```xml
<owl:Class rdf:ID="Grape">
  ...
</owl:Class>

<owl:Class rdf:ID="WineGrape">
  <rdfs:subClassOf rdf:resource="&food;Grape" />
</owl:Class>

<WineGrape rdf:ID="CabernetSauvignonGrape" />
Simple Named Classes:
- **Class**
- **rdfs:subClassOf**

**Individual**

Defining Properties:
- **rdf:Property**
  - subproperties:
    - **owl:ObjectProperty**
      - (Instance - Instance)
    - **owl:DatatypeProperty**
      - (Instance - rdfs:Literal/ XML Schema datatypes)
    - **rdfs:subPropertyOf**
    - **rdfs:domain**
    - **rdfs:range**

Properties of Individuals
Simple Named Classes:

Class

rdfs:subClassOf

Individual

Defining Properties:

rdf:Property

subproperties:

owl:ObjectProperty
(Instance - Instance)

owl:DatatypeProperty
(Instance - rdfs:Literal / XML Schema datatypes)

rdfs:subPropertyOf

rdfs:domain

rdfs:range

Properties of Individuals

<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal"/>
  <rdfs:range rdf:resource="#MealCourse"/>
</owl:ObjectProperty>
Simple Named Classes:

Class

rdfs:subClassOf

Individual

Defining Properties:

rdf:Property

subproperties:

owl:ObjectProperty
(Instance - Instance)

owl:DatatypeProperty
(Instance - rdfs:Literal/
XML Schema datatypes)

rdfs:subPropertyOf

rdfs:domain

rdfs:range

Properties of Individuals

<owl:Class rdf:ID="WineDescriptor" />

<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />
  ...
</owl:Class>

<owl:ObjectProperty rdf:ID="hasWineDescriptor">
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineDescriptor" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasColor">
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
  <rdfs:range rdf:resource="#WineColor" />
  ...
</owl:ObjectProperty>
Simple Named Classes:

Class

\texttt{rdfs:subClassOf}

Individual

Defining Properties:

\texttt{rdf:Property}

subproperties:

\texttt{owl:ObjectProperty}

(Instance - Instance)

\texttt{owl:DatatypeProperty}

(Instance - \texttt{rdfs:Literal}/\texttt{XML Schema datatypes})

\texttt{rdfs:subPropertyOf}
\texttt{rdfs:domain}
\texttt{rdfs:range}

Properties of Individuals
Simple Named Classes:

Class

rdfs:subClassOf

Individual

Defining Properties:

rdf:Property

subproperties:

owl:ObjectProperty
(Instance - Instance)

owl:DatatypeProperty
(Instance - rdfs:Literal/
XML Schema datatypes)

rdfs:subPropertyOf
rdfs:domain
rdfs:range

Properties of Individuals
Simple Named Classes:

Class

\textit{rdfs:subClassOf}

Individual

Defining Properties:

\textit{rdf:Property}

subproperties:

\textit{owl:ObjectProperty}
(Instance - Instance)

\textit{owl:DatatypeProperty}
(Instance - \textit{rdfs:Literal} / XML Schema datatypes)

\textit{rdfs:subPropertyOf}

\textit{rdfs:domain}

\textit{rdfs:range}

Properties of Individuals

\texttt{<Region rdf:ID="SantaCruzMountainsRegion" >}
\texttt{<locatedIn rdf:resource="#CaliforniaRegion" />}
\texttt{</Region>}

\texttt{<Winery rdf:ID="SantaCruzMountainVineyard" />}

\texttt{<CabernetSauvignon}
\texttt{ rdf:ID="SantaCruzMountainVineyardCabernetSauvignon" >}
\texttt{<locatedIn rdf:resource="#SantaCruzMountainsRegion"/>}
\texttt{<hasMaker rdf:resource="#SantaCruzMountainVineyard"/>}
\texttt{</CabernetSauvignon>}

[W3Ca, W3Cb]
... powerful mechanism for enhanced reasoning about a property ...

**TransitiveProperty**

\[ P(x,y) \text{ and } P(y,z) \implies P(x,z) \]

**SymmetricProperty**

\[ P(x,y) \text{ iff } P(y,x) \]

**FunctionalProperty**

\[ P(x,y) \text{ and } P(x,z) \implies y = z \]

**InverseOf**

\[ P1(x,y) \text{ iff } P2(y,x) \]

**InverseFunctionalProperty**

\[ P(y,x) \text{ and } P(z,x) \implies y = z \]
OWL Lite Constructs: Property Characteristics

... powerful mechanism for enhanced reasoning about a property ...

TransitiveProperty

SymmetricProperty

FunctionalProperty

inverseOf

InverseFunctionalProperty

\[
P(x,y) \text{ and } P(y,z) \text{ implies } P(x,z)
\]

\[
P(x,y) \text{ iff } P(y,x)
\]

\[
P(x,y) \text{ and } P(x,z) \text{ implies } y = z
\]

\[
P1(x,y) \text{ iff } P2(y,x)
\]

\[
P(y,x) \text{ and } P(z,x) \text{ implies } y = z
\]

```xml
<owl:ObjectProperty rdf:ID="adjacentRegion">
  <rdfs:domain rdf:resource="#Region"/>
  <rdfs:range rdf:resource="#Region"/>
</owl:ObjectProperty>

<Region rdf:ID="MendocinoRegion">
  <locatedIn rdf:resource="#CaliforniaRegion"/>
  <adjacentRegion rdf:resource="#SonomaRegion"/>
</Region>
```

[W3Ca, W3Cb]
OWL Lite Constructs: Property Characteristics

... powerful mechanism for enhanced reasoning about a property ...

**TransitiveProperty**

\[ P(x,y) \text{ and } P(y,z) \text{ implies } P(x,z) \]

**SymmetricProperty**

\[ P(x,y) \text{ iff } P(y,x) \]

**FunctionalProperty**

\[ P(x,y) \text{ and } P(x,z) \text{ implies } y = z \]

**InverseOf**

\[ P1(x,y) \text{ iff } P2(y,x) \]

**InverseFunctionalProperty**

\[ P(y,x) \text{ and } P(z,x) \text{ implies } y = z \]
... powerful mechanism for enhanced reasoning about a property ...

**TransitiveProperty**

\[ P(x,y) \text{ and } P(y,z) \text{ implies } P(x,z) \]

**SymmetricProperty**

\[ P(x,y) \text{ iff } P(y,x) \]

**FunctionalProperty**

\[ P(x,y) \text{ and } P(x,z) \text{ implies } y = z \]

**inverseOf**

\[ P1(x,y) \text{ iff } P2(y,x) \]

**InverseFunctionalProperty**

\[ P(y,x) \text{ and } P(z,x) \text{ implies } y = z \]

```xml
<owl:ObjectProperty rdf:ID="hasMaker">
  <rdf:type rdf:resource="#FunctionalProperty"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="producesWine">
  <owl:inverseOf rdf:resource="#hasMaker"/>
</owl:ObjectProperty>
```
OWL Lite Constructs: Property Characteristics

... powerful mechanism for enhanced reasoning about a property ...

- **TransitiveProperty**
  \[ P(x,y) \text{ and } P(y,z) \implies P(x,z) \]

- **SymmetricProperty**
  \[ P(x,y) \iff P(y,x) \]

- **FunctionalProperty**
  \[ P(x,y) \text{ and } P(x,z) \implies y = z \]

- **InverseOf**
  \[ P1(x,y) \iff P2(y,x) \]

- **InverseFunctionalProperty**
  \[ P(y,x) \text{ and } P(z,x) \implies y = z \]

```xml
<owl:ObjectProperty rdf:ID="hasMaker" />
<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="#inverseOf" />
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```
Values:

\textbf{owl:allValuesFrom}

e.g., For all wines, if they have makers, all the makers are wineries.

\textbf{owl:someValuesFrom}

e.g. For all wines, they have at least one maker that is a winery.

Cardinalities (only 0 or 1):

\textbf{owl:minCardinality}
\textbf{owl:maxCardinality}
\textbf{owl:cardinality}
Values:

- **owl:allValuesFrom**
  
e.g., For all wines, if they have makers, all the makers are wineries.

- **owl:someValuesFrom**
  
e.g. For all wines, they have at least one maker that is a winery.

Cardinalities (only 0 or 1):

- **owl:minCardinality**
- **owl:maxCardinality**
- **owl:cardinality**

```xml
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="#PotableLiquid" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:someValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
Values:

\texttt{owl:allValuesFrom}

e.g., For all wines, if they have makers, all the makers are wineries.

\texttt{owl:someValuesFrom}

e.g. For all wines, they have at least one maker that is a winery.

Cardinalities (only 0 or 1):

\texttt{owl:minCardinality}
\texttt{owl:maxCardinality}
\texttt{owl:cardinality}
owl:equivalentClass
owl:equivalentProperty
owl:sameAs
owl:differentFrom
owl:AllDifferent, owl:distinctMembers

```
<owl:Class rdf:ID="Wine">
  <owl:equivalentClass rdf:resource="&vin;Wine"/>
</owl:Class>
```

```
<owl:Class rdf:ID="TexasThings">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#locatedIn" />
      <owl:someValuesFrom rdf:resource="#TexasRegion" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

```
subClassOf  | TexasThings(x) implies locatedIn(x,y) and TexasRegion(y)  
equivalentClass | TexasThings(x) implies locatedIn(x,y) and TexasRegion(y) locatedIn(x,y) and TexasRegion(y) implies TexasThings(x)
```
OWL Lite Language Constructs: (In)Equality

- owl:equivalentClass
- owl:equivalentProperty
- owl:sameAs
- owl:differentFrom
- owl:AllDifferent, owl:distinctMembers
OWL Lite Language Constructs: (In)Equality

owl:equivalentClass
owl:equivalentProperty
owl:sameAs
owl:differentFrom
owl:AllDifferent, owl:distinctMembers

```
<Wine rdf:ID="MikesFavoriteWine">
  <owl:sameAs rdf:resource="#StGenevieveTexasWhite" />
</Wine>
```

```
<owl:Thing rdf:about="#BancroftChardonnay">
  <hasMaker rdf:resource="#Bancroft" />
  <hasMaker rdf:resource="#Beringer" />
</owl:Thing>
```
OWL Lite Language Constructs: (In)Equality

owl:equivalentClass
owl:equivalentProperty
owl:sameAs
owl:differentFrom
owl:AllDifferent, owl:distinctMembers

```xml
<wineSugar rdf:ID="Dry" />
<wineSugar rdf:ID="Sweet">
  <owl:differentFrom rdf:resource="#Dry"/>
</wineSugar>
<wineSugar rdf:ID="OffDry">
  <owl:differentFrom rdf:resource="#Dry"/>
  <owl:differentFrom rdf:resource="#Sweet"/>
</wineSugar>
```
OWL Lite Language Constructs: (In)Equality

owl:equivalentClass
owl:equivalentProperty
owl:sameAs
owl:differentFrom
owl:AllDifferent, owl:distinctMembers

```xml
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red"/>
    <vin:WineColor rdf:about="#White"/>
    <vin:WineColor rdf:about="#Rose"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```
**OWL DL Language Constructs: Complex Classes**

- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`
- `owl:oneOf`
- `owl:disjointWith`

```xml
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor" />
      <owl:hasValue rdf:resource="#White" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```xml
<owl:Class rdf:about="#Burgundy">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#locatedIn" />
      <owl:hasValue rdf:resource="#BourgogneRegion" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
OWL DL Language Constructs: Complex Classes

OWL DL Language Constructs:

- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`
- `owl:oneOf`
- `owl:disjointWith`

Example:

```xml
<owl:Class rdf:ID="WhiteBurgundy">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Burgundy" />
    <owl:Class rdf:about="#WhiteWine" />
  </owl:intersectionOf>
</owl:Class>
```
OWL DL Language Constructs: Complex Classes

```xml
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SweetFruit" />
    <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```

```xml
<owl:Class rdf:ID="Fruit">
  <rdfs:subClassOf rdf:resource="#SweetFruit" />
  <rdfs:subClassOf rdf:resource="#NonSweetFruit" />
</owl:Class>
```
OWL DL Language Constructs: Complex Classes

owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```xml
<owl:Class rdf:ID="ConsumableThing" />
<owl:Class rdf:ID="NonConsumableThing">
  <owl:complementOf rdf:resource="#ConsumableThing" />
</owl:Class>
```
OWL DL Language Constructs: Complex Classes

**owl:intersectionOf**
**owl:unionOf**
**owl:complementOf**
**owl:oneOf**
**owl:disjointWith**

```
<owl:Class rdf:ID="NonFrenchWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine"/>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#locatedIn" />
          <owl:hasValue rdf:resource="#FrenchRegion" />
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

[W3Ca, W3Cb]
OWL DL Language Constructs: Complex Classes

owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#White"/>
    <owl:Thing rdf:about="#Rose"/>
    <owl:Thing rdf:about="#Red"/>
  </owl:oneOf>
</owl:Class>
```

```
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
  <owl:oneOf rdf:parseType="Collection">
    <WineColor rdf:about="#White"/>
    <WineColor rdf:about="#Rose"/>
    <WineColor rdf:about="#Red"/>
  </owl:oneOf>
</owl:Class>
```
OWL DL Language Constructs: Complex Classes

owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```xml
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
  <owl:oneOf rdf:parseType="Collection">
    <WineColor rdf:about="#White"/>
    <WineColor rdf:about="#Rose"/>
    <WineColor rdf:about="#Red"/>
  </owl:oneOf>
</owl:Class>

<WineColor rdf:about="#White">
  <rdfs:label>White</rdfs:label>
</WineColor>
```
OWL DL Language Constructs: Complex Classes

owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```xml
<owl:DatatypeProperty rdf:ID="tennisGameScore">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf>
        <rdfs:List>
          <rdf:rest>
            <rdf:List>
              <rdf:first rdf:datatype="&xsd;integer">0</rdf:first>
              <rdf:rest>
                <rdf:List>
                  <rdf:first rdf:datatype="&xsd;integer">15</rdf:first>
                  <rdf:rest>
                    <rdf:List>
                      <rdf:first rdf:datatype="&xsd;integer">30</rdf:first>
                      <rdf:rest>
                        <rdf:List>
                          <rdf:first rdf:datatype="&xsd;integer">40</rdf:first>
                          <rdf:rest rdf:resource="#rdf:nil" />
                        </rdf:List>
                      </rdf:rest>
                    </rdf:List>
                  </rdf:rest>
                </rdf:List>
              </rdf:rest>
            </rdf:List>
          </rdf:rest>
        </rdfs:List>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
```

[W3Ca, W3Cb]
owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```
<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing"/>
  <owl:disjointWith rdf:resource="#Meat"/>
  <owl:disjointWith rdf:resource="#Fowl"/>
  <owl:disjointWith rdf:resource="#Seafood"/>
  <owl:disjointWith rdf:resource="#Dessert"/>
  <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```
OWL DL Language Constructs: Complex Classes

owl:intersectionOf
owl:unionOf
owl:complementOf
owl:oneOf
owl:disjointWith

```
<owl:Class rdf:ID="SweetFruit">
  <rdfs:subClassOf rdf:resource="#EdibleThing" />
</owl:Class>

<owl:Class rdf:ID="NonSweetFruit">
  <rdfs:subClassOf rdf:resource="#EdibleThing" />
  <owl:disjointWith rdf:resource="#SweetFruit" />
</owl:Class>

<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SweetFruit" />
    <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```
Outline

• Basic Ideas of OWL
• The OWL Language
  OWL Lite: Simple Classes and Individuals
  OWL Lite: Property Characteristics and Restrictions
  OWL Lite: Constructs
  OWL DL: Complex Classes
• Summary
OWL ...

...is a Web Ontology Language designed for use by applications that need to process the content of information instead of just presenting information to humans.

...is the proposed standard for Web ontologies.

...is used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms -> Ontology.

[W3Ca, Antoniou and van Harmelen, 2004]
OWL ...

... facilitates greater machine interpretability of Web content than XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

... is a revision of the DAML+OIL web ontology language and builds upon RDF and RDF Schema:

- (XML-based) RDF syntax is used.
- Instances are defined using RDF descriptions.
- Most RDFS modeling primitives are used.

... has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

[W3Ca, Antoniou and van Harmelen, 2004]
OWL ...

... provides formal semantics and reasoning support through the mapping of OWL on logics.

... is sufficiently rich to be used in practice, extensions are in the making: They will provide further logical features, including rules.

... needs user-friendly tool-support for automatic or semi-automatic generation of OWL-code.

[W3Ca, Antoniou and van Harmelen, 2004]
References & Resources


Monika Lanzenberger
lanzenberger@ifs.tuwien.ac.at

Vienna University of Technology
Institute of Software Technology and Interactive Systems
1040 Wien, Favoritenstr. 9-11, 2nd Floor, HD 0211

Hours: Tuesday, 10-11h

next: Examples, OWL in practice, Protégé & Ontology Engineering
...Grigoris Antoniou and
...Frank van Harmelen

for making nice slides of their presentations available.