Methods and Tools for Management Information Systems

Lecture 5

8. Dezember 2008
Web Ontology Language (OWL)

- Intended to be used when the information contained in documents needs to be processed by *applications* rather than humans
- Explicitly represents the *meaning* of terms in vocabularies and the relationships between those terms
- Representation of terms and their interrelationships is called an *ontology*
- Offers more facilities for expressing meaning and semantics than XML, RDF and RDF Schema
OWL is part of the growing stack of W3C recommendations related to the Semantic Web:

- 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 datatypes
- RDF is a datamodel for objects (i.e. resources) and relations between them, provides a simple semantics for this datamodel, and these datamodels 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:

- Relations between classes (e.g. disjointness)
- Cardinality (e.g. exactly one)
- Equality
- Richer typing of properties
- Characteristics of properties (e.g. symmetry)
- Enumerated classes
OWL provides three increasingly expressive sublanguages:

- **OWL Lite** supports classification hierarchies and simple constraints
- **OWL DL** allows for maximum expressiveness while retaining *computational completeness* (all conclusions are guaranteed to be computable) and *decidability* (all computations will finish in finite time)
- **OWL Full** allows for maximum expressiveness and the syntactic freedom of RDF with no computational guarantees

Each of these sublanguages is an extension of its simpler predecessor:

- 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
OWL Lite Overview:

**RDF Schema Features:**
- Class (Thing, Nothing)
- rdfs:subClassOf
- rdf:Property
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
- Individual

**Property Characteristics:**
- ObjectProperty
- inverseOf
- SymmetricProperty
- InverseFunctionalProperty

**(In)Equality:**
- equivalentClass
- equivalentProperty
- sameAs
- differentFrom
- AllDifferent
- distinctMembers
- DatatypeProperty
- TransitiveProperty
- FunctionalProperty
OWL Lite Overview (2):

**Property Restrictions:**
- Restriction
- onProperty
- allValuesFrom
- someValuesFrom

**Restricted Cardinality:**
- minCardinality
- maxCardinality
- cardinality

**Header Information:**
- Ontology
- imports

**Class Intersection:**
- intersectionOf

**Datatypes:**
- xsd datatypes
OWL Lite Overview (3):

**Versioning:**

- versionInfo
- priorVersion
- backwardCompatibleWith
- incompatibleWith
- DeprecatedClass
- DeprecatedProperty

**Annotation Properties:**

- rdfs:label
- rdfs:comment
- rdfs:seeAlso
- rdfs:isDefinedBy
- AnnotationProperty
- OntologyProperty
OWL DL and Full Overview:

Class Axioms:

⇒ oneOf
⇒ disjointWith
⇒ equivalentClass
  (applied to class expressions)
⇒ rdfs:subClassOf
  (applied to class expressions)

Arbitrary Cardinality:

⇒ minCardinality
⇒ maxCardinality
⇒ cardinality

Boolean Combinations of Class Expressions:

⇒ unionOf
⇒ complementOf
⇒ intersectionOf

Filler Information:

⇒ hasValue
**Class:**

⇒ A group of individuals that belong together because they share some properties

⇒ Classes can be organized in a specialization hierarchy using subClassOf

⇒ There is a built-in most general class named Thing that is the class of all individuals and is a superclass of all OWL classes

⇒ There is also a built-in most specific class named Nothing that is the class that has no instances and a subclass of all OWL classes

**Example:**

```xml
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```
**rdfs:subClassOf:**

⇒ Class hierarchies may be created by making one or more statements that a class is a subclass of another class

⇒ May only be used in conjunction with single class names (as opposed to OWL DL and Full)

*Example:*

```xml
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing"/>
</owl:Class>
```
**rdf:Property:**

⇒ Properties can be used to state relationships between individuals (*owl:ObjectProperty*) or from individuals to XSD data values and RDF literals (*owl:DatatypeProperty*)

⇒ *owl:ObjectProperty* and *owl:DatatypeProperty* are subclasses of the RDF class *rdf:Property*

*Example:*

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

⇒ Property hierarchies may be created by making one or more statements that a property is a subproperty of one or more other properties

*Example:*

```xml
<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>
```
rdfs:domain:
  ⇒ A domain of a property limits the individuals to which the property can be applied

Example:

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

⇒ The range of a property limits the individuals that the property may have as its value

*Example:*

```xml
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```
Individual:
⇒ Individuals are instances of classes, and properties may be used to relate one individual to another

Example:

```xml
<owl:Thing rdf:ID="CentralCoastRegion"/>
<owl:Thing rdf:about="#CentralCoastRegion">
  <rdf:type rdf:resource="#Region"/>
</owl:Thing>

<WineGrape rdf:ID="CabernetSauvignonGrape"/>
```
**equivalentClass:**

- Two classes may be stated to be equivalent
- May only be used in conjunction with single class names (as opposed to OWL DL and Full)
- Equivalent classes have the same instances
- Equality can be used to create synonymous classes

**Example:**

```xml
<owl:Class rdf:ID="Wine">
  <owl:equivalentClass rdf:resource="&vin;Wine"/>
</owl:Class>
```
equivalentProperty:
⇒ Two properties may be stated to be equivalent
⇒ Equivalent properties have the same range and domain
⇒ Equality can be used to create synonymous properties

Example:

```xml
<rdf:Property rdf:ID="weight">
  <owl:equivalentProperty rdf:resource="&ex;weight"/>
</rdf:Property>
```
sameAs:

⇒ Two individuals may be stated to be the same
⇒ These constructs may be used to create a number of different names that refer to the same individual

Example:

<Wine rdf:ID="MikesFavoriteWine">
   <owl:sameAs rdf:resource="#TexasWhite"/>
</Wine>
**differentFrom:**

⇒ An individual may be stated to be different from other individuals

*Example:*

```xml
<WineSugar rdf:ID="Dry"/>

<WineSugar rdf:ID="Sweet">
  <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>
```
**AllDifferent:**

⇒ A number of individuals may be stated to be mutually distinct in one statement

⇒ Particularly useful when there are sets of distinct objects and when modelers are interested in enforcing the unique names assumption within those sets of objects

**Example:**

```xml
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red"/>
    <vin:WineColor rdf:about="#White"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```
**distinctMembers:**
⇒ All members of a list are distinct and pairwise disjoint
⇒ Can only be used in combination with owl:AllDifferent

*Example:*

```xml
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red"/>
    <vin:WineColor rdf:about="#White"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```
inverseOf:
⇒ One property may be stated to be the inverse of another property:

\[ P_1(x, y) \Rightarrow P_2(y, x) \]

Example:

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

<owl:ObjectProperty rdf:ID="producesWine">
    <owl:inverseOf rdf:resource="#hasMaker"/>
</owl:ObjectProperty>
```
TransitiveProperty:
⇒ Properties may be stated to be transitive:
\[ P(x, y) \land P(y, z) \Rightarrow P(x, z) \]

Example:

```xml
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="&owl;Thing"/>
  <rdfs:range rdf:resource="#Region"/>
</owl:ObjectProperty>
```
**SymmetricProperty:**

⇒ Properties may be stated to be symmetric:

\[ P(x, y) \Rightarrow P(y, x) \]

*Example:*

```xml
<owl:ObjectProperty rdf:ID="adjacentRegion">
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:domain rdf:resource="#Region"/>
  <rdfs:range rdf:resource="#Region"/>
</owl:ObjectProperty>
```
FunctionalProperty:

⇒ If a property is a owl:FunctionalProperty, then it has no more than one value for each individual (unique property):

\[ P(x, y) \land P(x, z) \Rightarrow y = z \]

⇒ Shorthand for stating that the property’s minimum cardinality is zero and its maximum cardinality is 1

Example:

```xml
<owl:ObjectProperty rdf:ID="hasVintageYear">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Vintage"/>
  <rdfs:range rdf:resource="#VintageYear"/>
</owl:ObjectProperty>
```
**InverseFunctionalProperty:**

⇒ If a property is inverse functional then the inverse of the property is functional (unambiguous property):

\[ P(y, x) \land P(z, x) \Rightarrow y = z \]

**Example:**

```xml
<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <owl:inverseOf rdf:resource="#hasMaker"/>
</owl:ObjectProperty>
```
**Restriction:**

⇒ For placing restrictions on the usage of properties by class instances

⇒ Applies to its containing class definition only

**Example:**

```xml
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <!-- .... -->
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
**onProperty:**

⇒ Indicates the restricted property

**Example:**

```xml
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker"/>
      <!-- .... -->
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
**allValuesFrom:**

⇒ Stated on a property with respect to a class (local range restriction)
⇒ Values of the property are all members of the class indicated

**Example:**

```xml
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker"/>
      <owl:allValuesFrom rdf:resource="#Winery"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
**someValuesFrom:**

⇒ Stated on a property with respect to a class (local range restriction)

⇒ At least one value for that property is of a certain type

*Example:*

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

⇒ Stated on a property with respect to a particular class
⇒ Permits the specification of the minimum number of elements in a relation (0 or 1)

Example:

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasMother"/>
  <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
</owl:Restriction>
```
**maxCardinality:**

⇒ Stated on a property with respect to a particular class

⇒ Permits the specification of the maximum number of elements in a relation (0 or 1)

**Example:**

```xml
<owl:Restriction>
    <owl:onProperty rdf:resource="#hasMother"/>
    <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
    </owl:maxCardinality>
</owl:Restriction>
```
**cardinality:**

⇒ Permits the specification of exactly the number of elements in a relation (0 or 1)

*Example:*

```xml
<owl:Class rdf:ID="Vintage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasVintageYear"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
Ontology:
⇒ An ontology is a resource and may be described using properties
⇒ If the value of rdf:about is empty, the name of the ontology is the base URI of the owl:Ontology element

Example:

```xml
<owl:Ontology rdf:about="">
  <owl:versionInfo>
    v 1.17 2003/02/26 12:56:51 mdean
  </owl:versionInfo>
  <rdfs:comment>
    Comments are used to annotate ontologies.
  </rdfs:comment>
</owl:Ontology>
```
imports:
⇒ References another OWL ontology containing definitions, whose meaning is considered to be part of the meaning of the importing ontology
⇒ owl:imports is a property with the class owl:Ontology as its domain and range
⇒ Imports are transitive

Example:

<owl:Ontology rdf:about="">
  <!-- .... -->
  <owl:imports rdf:resource="http://www.example.org/foo"/>
</owl:Ontology>
**intersectionOf:**

⇒ Allows for intersections of named classes and restrictions

**Example:**

```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>
```
xsd datatypes:
⇒ OWL uses most of the built-in XML Schema datatypes

Example:

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

<owl:DatatypeProperty rdf:ID="yearValue">
  <rdfs:domain rdf:resource="#VintageYear"/>
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```
versionInfo:
⇒ A string giving information about the version of the corresponding OWL construct

Example:

```xml
<owl:Ontology rdf:about="">
  <owl:versionInfo>
    v 1.17 2003/02/26 12:56:51 mdean
  </owl:versionInfo>
</owl:Ontology>
```
priorVersion:
⇒ Reference to another ontology that identifies the specified ontology as a prior version of the containing ontology

Example:

<owl:Ontology rdf:about="">
  <owl:priorVersion rdf:resource="http://www.example.org/old"/>
</owl:Ontology>
**backwardCompatibleWith:**

⇒ Reference to another ontology that identifies the specified ontology as a prior version of the containing ontology, and further indicates that it is backward compatible with it

*Example:*

```xml
<owl:Ontology rdf:about="">
  <owl:backwardCompatibleWith
    rdf:resource="http://www.example.org/old"/>
</owl:Ontology>
```
incompatibleWith:

⇒ Reference to another ontology that identifies the specified ontology as a prior version of the containing ontology, and further indicates that is not backward compatible with it

Example:

```xml
<owl:Ontology rdf:about="">
  <owl:incompatibleWith
      rdf:resource="http://www.example.org/old"/>
</owl:Ontology>
```
*DeprecationClass:*

⇒ Class is preserved for backward-compatibility purposes, but may be phased out in the future

*Example:*

```xml
<owl:DeprecatedClass rdf:ID="Car">
  <rdfs:comment>Automobile is now preferred</rdfs:comment>
  <owl:equivalentClass rdf:resource="#Automobile"/>
</owl:DeprecatedClass>
```
**DeprecatedProperty:**

⇒ Property is preserved for backward-compatibility purposes, but may be phased out in the future

*Example:*

```xml
<owl:DeprecatedProperty rdf:ID="hasDriver">
  <rdfs:comment>inverse property drives is now preferred</rdfs:comment>
  <owl:inverseOf rdf:resource="#drives" />
</owl:DeprecatedProperty>
```
**AnnotationProperty:**

⇒ Declares a property that is used as an annotation

⇒ The object of an annotation property must be either a data literal, a URI reference, or an individual

*Example:*

```xml
<owl:AnnotationProperty rdf:about="&dc;creator"/>

<owl:Class rdf:about="#MusicalWork">
    <rdfs:label>Musical work</rdfs:label>
    <dc:creator>N.N.</dc:creator>
</owl:Class>
```
OntologyProperty:

⇒ Properties that apply to the ontology as a whole
⇒ Instances of this class must have the class owl:Ontology as their domain and range

Example:

```xml
<rdf:Property rdf:ID="incompatibleWith">
  <rdfs:label>incompatibleWith</rdfs:label>
  <rdf:type rdf:resource="#OntologyProperty"/>
  <rdfs:domain rdf:resource="#Ontology"/>
  <rdfs:range rdf:resource="#Ontology"/>
</rdf:Property>
```
**oneOf:**

⇒ Classes can be described by enumeration of the individuals that make up the class, i.e., the members of the class are exactly the set of enumerated individuals

*Example:*

```xml
<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>
```
**disjointWith:**

⇒ Classes may be stated to be disjoint from each other

⇒ Such statements are mainly used to avoid inconsistencies

*Example:*

```xml
<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>
```
equivalentClass:
⇒ Same meaning as in OWL Lite
⇒ May additionally be complex class definitions

Example:

```xml
<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>
```
rdfs:subClassOf:
⇒ Same meaning as in OWL Lite
⇒ May additionally be complex class definitions

Example:

```xml
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
        1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
unionOf:
⇒ Boolean combination of classes and/or restrictions

Example:

```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>
```
complementOf:
⇒ Boolean combination of classes and/or restrictions

Example:

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

<owl:Class rdf:ID="NonConsumableThing">
  <owl:complementOf rdf:resource="#ConsumableThing"/>
</owl:Class>
```
**intersectionOf:**

⇒ Boolean combination of classes and/or restrictions

*Example:*

```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>
```
minCardinality:
⇒ Stated on a property with respect to a particular class
⇒ Specifies the minimum number of elements in a relation

Example:

```
<owl:Class rdf:ID="Tricycle">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasWheels"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">3</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
**maxCardinality:**

⇒ Stated on a property with respect to a particular class
⇒ Specifies the maximum number of elements in a relation

*Example:*

```xml
<owl:Class rdf:ID="Tricycle">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasWheels"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">3</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
cardinality:
⇒ Permits the specification of exactly the number of elements in a relation

Example:

```
<owl:Class rdf:ID="Tricycle">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasWheels"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
        3
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
**hasValue:**

⇒ A property can be required to have a certain individual as a value

⇒ Allows for the specification of classes based on the existence of particular property values

*Example:*

```xml
<owl:Class rdf:ID="Burgundy">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSugar"/>
      <owl:hasValue rdf:resource="#Dry"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
References:

- OWL Web Ontology Language Overview
- OWL Web Ontology Language Guide
- OWL Web Ontology Language Reference