Ontologies

- are data models that represent a domain and are used to reason about the objects in that domain and the relations between them
  - this applies only to computer and information science
- ontologies generally describe:
  - individuals: the basic or "ground level" objects
  - classes: sets, collections, or types of objects
  - attributes: properties, features, characteristics, or parameters that objects can have and share
  - relations: ways that objects can be related to one another
- to encode ontologies, formal ontology languages are used

Individuals

- Individuals (instances) are the basic, "ground level" components of an ontology. The individuals in an ontology may include concrete objects such as people, animals, tables, automobiles, molecules, and planets, as well as abstract individuals such as numbers and words. Strictly speaking, an ontology need not include any individuals, but one of the general purposes of an ontology is to provide a means of classifying individuals, even if those individuals are not explicitly part of the ontology.

Classes

- Classes are abstract groups, sets, or collections of objects. They may contain individuals, other classes, or a combination of both. Some examples of classes:
  - **person**, the class of all people
  - **number**, the class of all numbers
  - **vehicle**, the class of all vehicles
  - **car**, the class of all cars
  - **individual**, representing the class of all individuals
  - **class**, representing the class of all classes
  - **thing**, representing the class of all things

Attributes

- Objects in the ontology can be described by assigning attributes to them. Each attribute has at least a name and a value, and is used to store information that is specific to the object it is attached to. For example the Ford Explorer object has attributes such as:
  - *Name*: Ford Explorer
  - *Number-of-doors*: 4
  - *Engine*: {4.0L, 4.6L}
  - *Transmission*: 6-speed
- The value of an attribute can be a complex data type.

Relationships

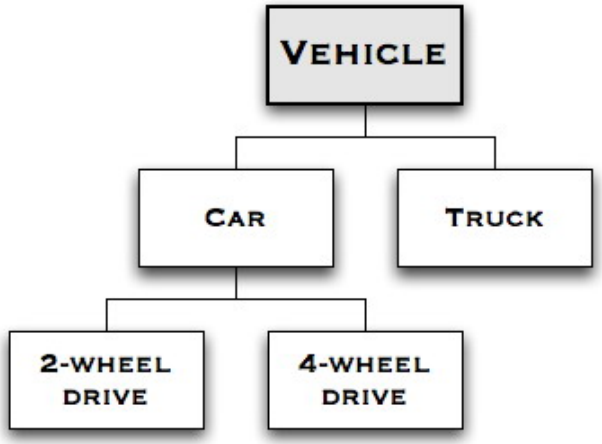
- An important use of attributes is to describe the *relationships* (or simply *relations*) between objects in the ontology. Typically a relation is an attribute whose value is another object in the ontology. For example in the ontology that contains the Ford Explorer and the Ford Bronco, the Ford Bronco object might have the following attribute:
  - *Successor*: Ford Explorer
- This means that the Explorer is the model that replaced the Bronco. Much of the power of ontologies comes from the ability to describe these relations. Together, the set of relations describes the semantics of the domain.

Domain vs. Upper Ontologies

- domain ontologies (or domain-specific ontologies)
  - model a specific domain, or part of the world
  - represent the particular meanings of terms as they apply to that domain
- upper ontologies (or foundation ontologies)
  - are models of the common objects that are generally applicable across a wide range of domain ontologies
  - contain a core glossary in whose terms objects in a set of domains can be described, e.g Dublin Core

Partitions (I)

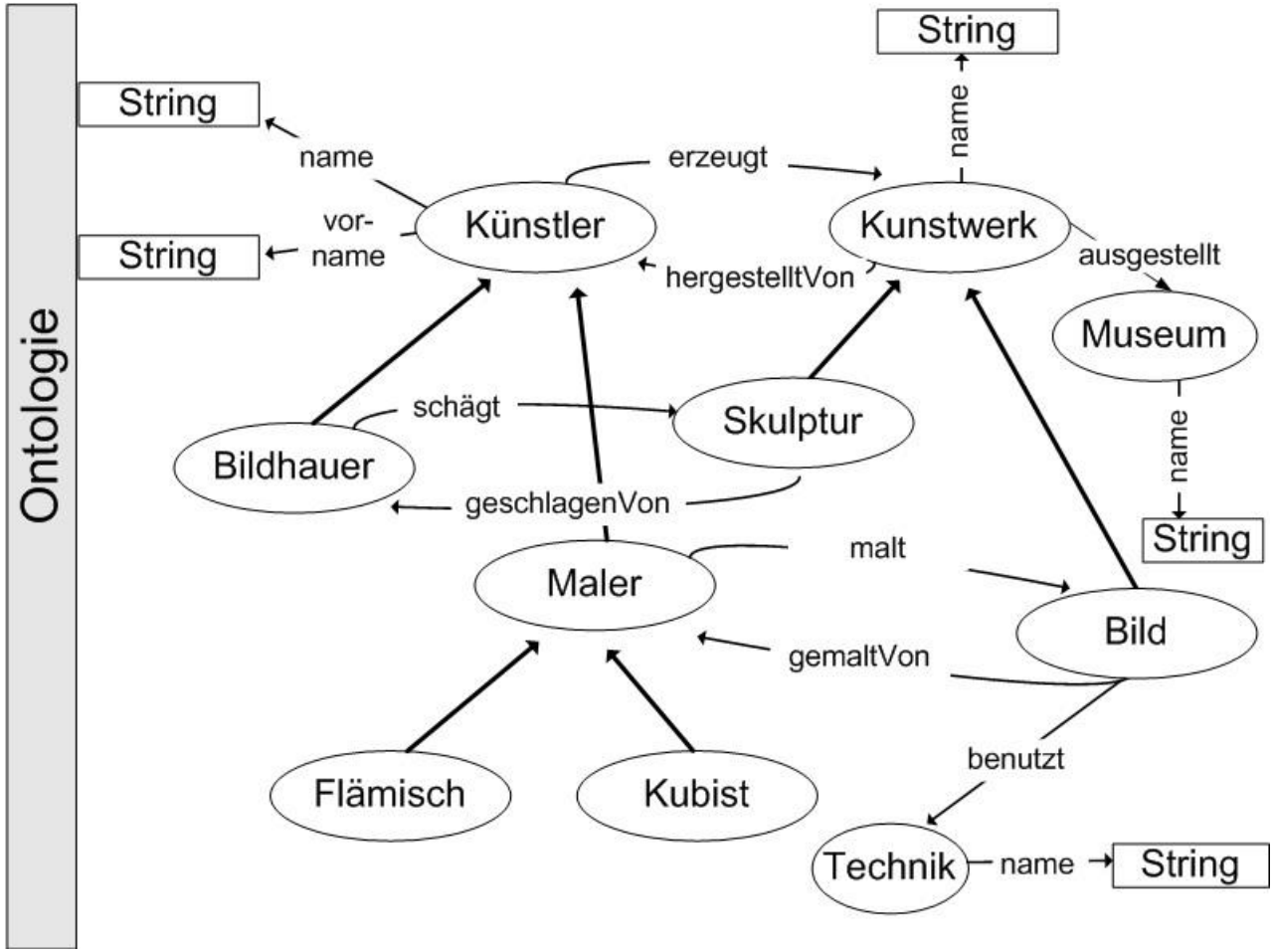
- A partition is a set of related classes and associated rules that allow objects to be placed into the appropriate class. For example, the partial diagram of an ontology has a partition of the Car class into the classes 2-Wheel Drive and 4-Wheel Drive.
- The partition rule determines if a particular car is placed in the 2-Wheel Drive or the 4-Wheel Drive class.



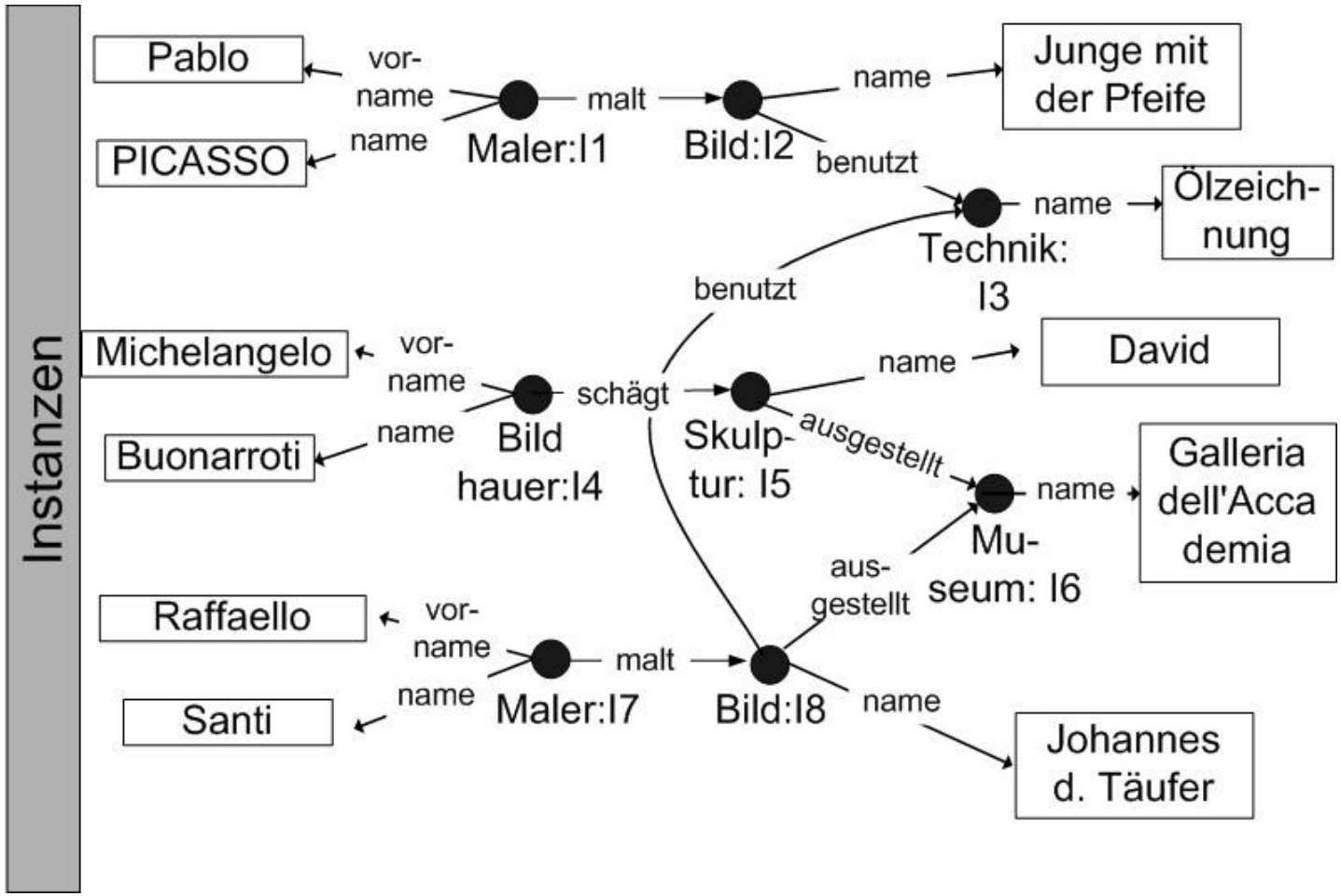
Partitions (II)

- If the partition rule(s) guarantee that a single Car object cannot be in both classes, then the partition is called a *Disjoint Partition*. If the partition rules ensure that every concrete object in the super-class is an instance of at least one of the partition classes, then the partition is called an *Exhaustive Partition*.

Ontology – Example (I)



Ontology – Example (II)



Web Ontology Language (OWL)

- 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 semantic for this datamodel, and these datamodels can be represented in an XML syntax
  - RDFS 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 (such as relations between classes, cardinality, equality, richer typing of properties…)

OWL Sublanguages

- 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 (I)

- RDF Schema Features:
  - Class (Thing, Nothing)
  - rdfs:subClassOf
  - rdf:Property
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
  - Individual
- (In)Equality:
  - equivalentClass
  - equivalentProperty
  - sameAs
  - differentFrom
  - AllDifferent
  - distinctMembers
- Property Characteristics:
  - ObjectProperty
  - DatatypeProperty
  - inverseOf
  - TransitiveProperty
  - SymmetricProperty
  - FunctionalProperty
  - InverseFunctionalProperty

OWL Lite Overview (II)

- Property Restrictions:
  - Restriction
  - onProperty
  - allValuesFrom
  - someValuesFrom
- Restricted Cardinality:
  - minCardinality
  - maxCardinality
  - cardinality
- Header Information:
  - Ontology
  - imports
- Class Intersection:
  - intersectionOf
- Datatypes:
  - xsd datatypes

OWL Lite Overview (III)

- 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)
- Boolean Combinations of Class Expressions:
  - unionOf
  - complementOf
  - intersectionOf
- Arbitrary Cardinality:
  - minCardinality
  - maxCardinality
  - cardinality
- Filler Information:
  - hasValue

OWL Vocabulary (I)

- Class
- a group of individuals that belong together because they share some properties
  - classes can be organized in a specialisation 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 is a subclass of all OWL classes
- example:
  - ```
    <owl:Class rdf:ID="Winery"/>
    <owl:Class rdf:ID="Region"/>
    <owl:Class rdf:ID="ConsumableThing"/>
    ```
    ```

OWL Vocabulary (II)

- 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:
  - ```
    <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>
    ```
    ```

OWL Vocabulary (III)

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

OWL Vocabulary (IV)

- 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:
  -
```
<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>
```

OWL Vocabulary (V)

- 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>
```

OWL Vocabulary (VI)

- rdfs:range
  - the range of a property limits the individuals that the property may have as its value
- example:
  -
```
<owl:ObjectProperty rdf:ID="madeFromGrape">
<rdfs:domain rdf:resource="#Wine"/>
<rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

OWL Vocabulary (VII)

- Individual
  - individuals are instances of classes, and properties may be used to relate one individual to another
- example:
  -
```
<owl:Thing rdf:ID="CentralCoastRegion"/>
<owl:Thing rdf:about="#CentralCoastRegion">
<rdf:type rdf:resource="#Region"/>
</owl:Thing>

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

OWL Vocabulary (VIII)

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

OWL Vocabulary (IX)

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

OWL Vocabulary (X)

- 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>
    ```
    ```

OWL Vocabulary (XI)

- differentFrom
  - an individual may be stated to be different from other individuals
- example:
  -
```
<WineSugar rdf:ID="Dry"/>

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

OWL Vocabulary (XII)

- 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:
  - ```
    <owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red"/>
    <vin:WineColor rdf:about="#White"/>
    </owl:distinctMembers>
    </owl:AllDifferent>
    ```
    ```

OWL Vocabulary (XIII)

- distinctMembers
  - all members of a list are distinct and pairwise disjoint
  - can only be used in combination with owl:AllDifferent
- example:
  -
```
<owl:AllDifferent>
<owl:distinctMembers rdf:parseType="Collection">
<vin:WineColor rdf:about="#Red"/>
<vin:WineColor rdf:about="#White"/>
</owl:distinctMembers>
</owl:AllDifferent>
```

OWL Vocabulary (XIV)

- inverseOf
  - one property may be stated to be the inverse of another property: P1(x, y) => P2(y, x)
- example:
  -
```
<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>
```

OWL Vocabulary (XV)

- TransitiveProperty
  - properties may be stated to be transitive: P(x, y) ^ P(y, z) => P(x, z)
- example:
  -
```
<owl:ObjectProperty rdf:ID="locatedIn">
<rdf:type rdf:resource="&owl;TransitiveProperty"/>
<rdfs:domain rdf:resource="&owl;Thing"/>
<rdfs:range rdf:resource="#Region"/>
</owl:ObjectProperty>
```

OWL Vocabulary (XVI)

- SymmetricProperty
  - properties may be stated to be symmetric: P(x, y) => P(y, x)
- example:
  -
```
<owl:ObjectProperty rdf:ID="adjacentRegion">
<rdf:type rdf:resource="&owl;SymmetricProperty"/>
<rdfs:domain rdf:resource="#Region"/>
<rdfs:range rdf:resource="#Region"/>
</owl:ObjectProperty>
```

OWL Vocabulary (XVII)

- FunctionalProperty
  - if a property is a owl:FunctionalProperty, then it has no more than one value for each individual (unique property): P(x, y) ^ P(x, z) => y = z
  - shorthand for stating that the property's minimum cardinality is zero and its maximum cardinality is 1
- example:
  - ```
    <owl:ObjectProperty rdf:ID="hasVintageYear">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Vintage"/>
    <rdfs:range rdf:resource="#VintageYear"/>
    </owl:ObjectProperty>
    ```
    ```

OWL Vocabulary (XVIII)

- InverseFunctionalProperty
  - if a property is inverse functional then the inverse of the property is functional (unambiguous property): P(y, x) ^ P(z, x) => y = z
- example:
  - ```
    <owl:ObjectProperty rdf:ID="producesWine">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <owl:inverseOf rdf:resource="#hasMaker"/>
    </owl:ObjectProperty>
    ```
    ```

OWL Vocabulary (XIX)

- Restriction
  - for placing restrictions on the usage of properties by class instances
  - applies to its containing class definition only
- example:
  - ```
    <owl:Class rdf:ID="Wine">
    <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
    <rdfs:subClassOf>
    <owl:Restriction>
    <!-- .... -->
    </owl:Restriction>
    </rdfs:subClassOf>
    </owl:Class>
    ```
    ```

OWL Vocabulary (XX)

- onProperty
  - indicates the restricted property
- example:

```
<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>
```

OWL Vocabulary (XXI)

- 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:

```
<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>
```

OWL Vocabulary (XXII)

- 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:
  - ```
    <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>
    ```
    ```

OWL Vocabulary (XXIII)

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

OWL Vocabulary (XXIV)

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

OWL Vocabulary (XXV)

- cardinality
  - permits the specification of exactly the number of elements in a relation (0 or 1)
- example:

```
<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>
```

OWL Vocabulary (XXVI)

- 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:
  -
```
<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>
```

OWL Vocabulary (XXVII)

- 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>
```

OWL Vocabulary (XXVIII)

- intersectionOf
  - allows for intersections of named classes and restrictions
- example:
  - ```
    <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 Vocabulary (XXIX)

- xsd datatypes
  - OWL uses most of the built-in XML schema datatypes
- example:
  -
```
<owl:Class rdf:ID="VintageYear"/>


<owl:DatatypeProperty rdf:ID="yearValue">
<rdfs:domain rdf:resource="#VintageYear"/>
<rdfs:range rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

OWL Vocabulary (XXX)

- versionInfo
  - a string giving information about the version of the corresponding OWL construct
- example:
  -
```
<owl:Ontology rdf:about="">
<owl:versionInfo>v 1.17 2003/02/26 12:56:51 mdean</owl:versionInfo>
</owl:Ontology>
```

OWL Vocabulary (XXXI)

- 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>
```

OWL Vocabulary (XXXII)

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

OWL Vocabulary (XXXIII)

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

OWL Vocabulary (XXXIV)

- DeprecatedClass
  - class is preserved for backward-compatibility purposes, but may be phased out in the future
- example:
  -
```
<owl:DeprecatedClass rdf:ID="Car">
<rdfs:comment>Automobile is now preferred</rdfs:comment>
<owl:equivalentClass rdf:resource="#Automobile"/>
</owl:DeprecatedClass>
```

OWL Vocabulary (XXXV)

- DeprecatedProperty
  - property is preserved for backward-compatibility purposes, but may be phased out in the future
- example:
  -
```
<owl:DeprecatedProperty rdf:ID="hasDriver">
<rdfs:comment>inverse property drives is now preferred
</rdfs:comment>
<owl:inverseOf rdf:resource="#drives" />
</owl:DeprecatedProperty>
```

OWL Vocabulary (XXXVI)

- 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:
  -
```
<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>
```

OWL Vocabulary (XXXVII)

- 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:
  - ```
    <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>
    ```
    ```

OWL Vocabulary (XXXVIII)

- 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:
  -
```
<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 Vocabulary (XIL)

- disjointWith
  - classes may be stated to be disjoint from each other
  - such statements are mainly used to avoid inconsistencies
- example:
  -
```
<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 Vocabulary (XL)

- equivalentClass
  - same meaning as in OWL Lite
  - may additionally be complex class definitions
- example:
  -
```
<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>
```

OWL Vocabulary (XLI)

- rdfs:subClassOf
  - same meaning as in OWL Lite
  - may additionally be complex class definitions
- example:
  -
```
<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 Vocabulary (XLII)

- unionOf
  - boolean combination of classes and/or restrictions
- example:
  -
```
<owl:Class rdf:ID="Fruit">
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#SweetFruit"/>
<owl:Class rdf:about="#NonSweetFruit"/>
</owl:unionOf>
</owl:Class>
```

OWL Vocabulary (XLIII)

- complementOf
  - boolean combination of classes and/or restrictions
- example:
  -
```
<owl:Class rdf:ID="ConsumableThing"/>


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

OWL Vocabulary (XLIV)

- intersectionOf
  - boolean combination of classes and/or restrictions
- example:
  -
```
<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 Vocabulary (XLV)

- 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>
```

OWL Vocabulary (XLVI)

- maxCardinality
  - stated on a property with respect to a particular class
  - specifies the maximum number of elements in a relation
- example:
  -
```
<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>
```

OWL Vocabulary (XLVII)

- 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>
```

OWL Vocabulary (XLVIII)

- 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:
  -
```
<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>
```

Resources

- W3C OWL Technical Report
  - http://www.w3.org/TR/owl-features/
- W3C OWL Language Guide
  - http://www.w3.org/TR/owl-guide/
- W3C OWL Language Reference
  - http://www.w3.org/TR/owl-ref/
- BORG – Bremen Ontology Research Group
  - http://www.fb10.uni-bremen.de/ontology/
- Protégé – open source ontology editor with OWL plugin
  - http://protege.stanford.edu/
- Metatomix m3t4.Studio Semantic Toolkit — RDF/OWL editor for Eclipse
  - http://www.m3t4.com