

# Web Ontology Language (OWL)

- ▶ Need meaning beyond an object-oriented type system
  - ▶ RDF (with RDFS) captures the basics, approximating an object-oriented type system
  - ▶ OWL provides some of the rest
- ▶ OWL standardizes constructs to capture such subtleties of meaning
- ▶ OWL builds on RDF, by limiting it
  - ▶ Limiting syntax
  - ▶ Limiting possible interpretations
- ▶ OWL assigns standard semantics to new terms

# OWL in Brief

- ▶ Specifies classes and properties in description logic (DL)
  - ▶ Class operators analogous to Boolean operators and, not, and or
  - ▶ Constraints on properties: transitive, ...
  - ▶ Restrictions: constructs unique to DL
- ▶ OWL 1 has “Species” or Dialects
  - ▶ OWL Full
  - ▶ OWL DL
  - ▶ OWL Lite

## Custom Metadata Vocabularies

- ▶ Metadata for services and information resources presupposes custom vocabularies
- ▶ Need standard semantics for the metadata to remove ambiguity despite heterogeneity

```
<Mammal rdf:ID='Mary'/>  
<Mammal rdf:ID='John'>  
  <hasParent rdf:resource='#Mary'/>  
</Mammal>
```

# Ontologies to Define Vocabulary Semantics

Example of a trivial ontology defining our vocabulary

- ▶ Uses simple subclasses and properties
  - ▶ Disjointness goes beyond RDF
  - ▶ Object properties refine RDF properties; relate two objects

```
<owl:Class rdf:ID="Mammal">  
  <rdfs:subClassOf rdf:resource="#Animal" />  
  <owl:disjointWith rdf:resource="#Reptile" />  
</owl:Class>  
  
<owl:ObjectProperty rdf:ID="hasParent">  
  <rdfs:domain rdf:resource="#Animal" />  
  <rdfs:range rdf:resource="#Animal" />  
</owl:ObjectProperty>
```

# Simple Inference

Find a model, if any exists

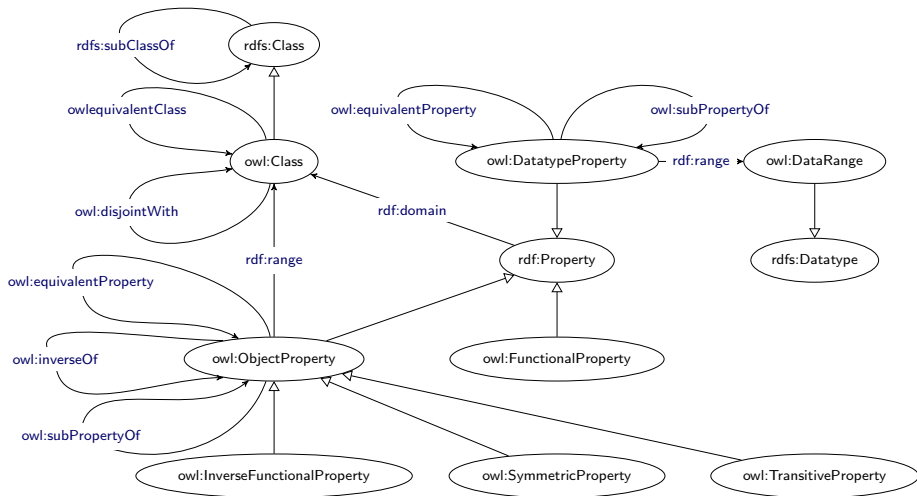
- ▶ Given the definition for the property `hasParent` and the snippet

```
<owl:Thing rdf:ID="Fido">  
  <hasParent rdf:resource="#Rover"/>  
</owl:Thing>
```

we can infer that Fido is an Animal

# OWL Entities and Relationships

Illustrative as a summary: not a proper OWL diagram



# Constructing OWL Classes

## ► Explicitly

```
<owl:Class rdf:ID="Mammal">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <owl:disjointWith rdf:resource="#Reptile"/>
</owl:Class>
```

## ► Anonymously via formal expressions using operators analogous to set operators:

intersectionOf, unionOf, complementOf

```
<owl:Class rdf:ID='SugaryBread'>
  <owl:intersectionOf rdf:parseType='Collection'>
    <owl:Class rdf:about='#Bread' />
    <owl:Class rdf:about='#SweetFood' />
  </owl:intersectionOf>
</owl:Class>
```

# Restrictions Conceptually

A unique feature of description logics

- ▶ Analogous to division in arithmetic: define classes in terms of a restriction that they satisfy with respect to a given property
- ▶ Anonymous: typically included in a class def to enable referring them
- ▶ Key primitives are
  - ▶ someValuesFrom a specified class
  - ▶ allValuesFrom a specified class
  - ▶ hasValue equal to a specified individual or data type
  - ▶ minCardinality
  - ▶ maxCardinality
  - ▶ cardinality (when maxCardinality equals minCardinality)



# Examples of Restrictions: 1

```
<owl:Restriction >
  <owl:onProperty rdf:resource="#hasFather"/>
  <owl:maxCardinality rdf:datatype="xsd:nonNegativeInteger">
    1
  </owl:maxCardinality>
</owl:Restriction >

<owl:Restriction >
  <owl:onProperty rdf:resource='#bakes'/>
  <owl:someValuesFrom rdf:resource='#Bread'/>
</owl:Restriction >
```

## Examples of Restrictions: 2

- ▶ The maker of a Wine must be a Winery
  - ▶ The allValuesFrom restriction is on the hasMaker property of this Wine class
  - ▶ (Makers of other products such as cheese are not constrained by this local restriction)

```

<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>

```

# Axioms Conceptually

Assertions that are given to be true

- ▶ Can be especially powerful in combination with other axioms, which may come from different documents
- ▶ Some primitives
  - ▶ `rdfs:subClassOf`
  - ▶ `owl:equivalentClass`

## Examples of Axioms

```
<owl:AllDifferent> <!-- in essence, pair-wise inequalities >
  <owl:distinctMembers rdf:parseType='Collection'>
    <ex:Country rdf:ID='India' />
    <ex:Country rdf:ID='Russia' />
    <ex:Country rdf:ID='USA' />
  </owl:distinctMembers />
</owl:AllDifferent >

<ex:Country rdf:ID='Iran' />
<ex:Country rdf:ID='Persia' >
  <owl:sameIndividualAs rdf:resource='#Iran' />
</ex:Country >
```

## Restrictions versus Axioms

- ▶ Axioms are global assertions that can be used as the basis for further inference
- ▶ Restrictions are constructors
- ▶ A restriction on `hasFather` of `maxCardinality` of 1
  - ▶ Does **not** mean all animals have zero or one fathers
  - ▶ Means the class of animals who have zero or one fathers: this class may or may not have any instances
- ▶ Often, to achieve the desired effect, we would have to combine restrictions with axioms (such as based on `equivalentClass`), e.g.,
  - ▶ A restriction on `hasFather` of `maxCardinality` of 1
  - ▶ An axiom asserting this restriction is equivalent to `Animal`

# Inference

Like RDF, OWL is about meaning, not syntax

- ▶ Statements from different documents about the same URI are automatically conjoined
- ▶ OWL can be surprising to the uninitiated
  - ▶ Integrity constraint: no one can have more than one mother
  - ▶ Declare a fact: Mary is John's mother
  - ▶ Declare a fact: Jane is John's mother
- ▶ What will you conclude?
  - ▶ A traditional DBMS would declare an integrity violation
  - ▶ An OWL reasoner would say  $Mary = Jane$

# Dialects Compared

- ▶ OWL DL
  - ▶ Core dialect, includes DL primitives
  - ▶ Not necessarily (but often) tractable
- ▶ OWL Lite
  - ▶ Limits OWL DL constructs to ensure tractability
  - ▶ No disjointWith, complementOf, unionOf, hasValue
  - ▶ Enumeration (oneOf)
  - ▶ intersectionOf only for two or more class names or restrictions
  - ▶ equivalentClass: class names to names or restrictions
  - ▶ rdfs:subClassOf: class names to names or restrictions
  - ▶ allValuesFrom and someValuesFrom: to class names or datatype names
  - ▶ rdf:type: to class names or restrictions
  - ▶ rdf:domain: class names
  - ▶ rdf:range: to class names or datatype names
- ▶ OWL Full
  - ▶ Extremely general: allows all RDF syntax
  - ▶ Potentially intractable (undecidable)
  - ▶ Supports fancy expressiveness needs and introducing new concepts into the standard

# Expressiveness Limitations: 1

OWL DL cannot express some simple requirements

- ▶ Non-tree models: because instance variables are implicit in OWL restrictions, OWL cannot express conditions that require that two variables be identified
  - ▶ Think of siblings—two people who have the same parents—but in terms of classes
  - ▶ Do the same thing with class definitions



## Expressiveness Limitations: 2

### Specialized properties

- ▶ Cannot state that the child of a mammal must be a mammal and so on, without
  - ▶ Defining new child properties for each class
  - ▶ Adding an axiom for each class stating that it is a subclassOf the restriction of hasChild to itself
- ▶ Analogous to the problem in a strongly typed object-oriented language without generics
  - ▶ You have to typecast the contents of a hash table or linked list

# Expressiveness Limitations: 3

## Constraints among individuals and defeasibility

- ▶ Constraints among individuals
  - ▶ Can define ETHusband: class of persons who have been married to Elizabeth Taylor
  - ▶ Cannot define tall person: class of persons whose height is above a certain threshold
- ▶ Cannot capture defeasibility (also known as nonmonotonicity)
  - ▶ Birds fly
  - ▶ Penguins are birds
  - ▶ Penguins don't fly

# OWL Summary

OWL builds on RDF to provide a rich vocabulary for capturing knowledge

- ▶ Synthesizes a lot of excellent work on discrete, taxonomic knowledge representation
- ▶ Fits well with describing information resources—a basis for describing metadata vocabularies
- ▶ Critical for unambiguously describing services so they can be selected and suitably engaged

## Modeling Exercise

- ▶ Student (S); faculty member (F); regular faculty member (R); department (D); thesis committee (T)
- ▶ An S belongs to exactly one D
- ▶ An R is an F
- ▶ An R advises zero or more Ss
- ▶ An F is affiliated with one or more Ds
- ▶ An S is advised by exactly one R
- ▶ An S is evaluated by exactly one T
- ▶ A T evaluates exactly one S
- ▶ A T has three or more Fs as its members
- ▶ Exactly one of the members of a T is its chair, who is an R