Chapter 4 (3)

OWL

Based on slides from Grigoris Antoniou and Frank van Harmelen

Outline

1. A bit of history
2. Basic Ideas of OWL
3. The OWL Language
4. **Examples**
5. **Beyond OWL**

African Wildlife Ontology

- An small example using OWL for an ontology of African animals and plants
- Used in 2nd edition of the Semantic Web Primer
- Used by Maria Keet for her course and book **An Introduction to Ontology Engineering**
- See The African Wildlife Ontology tutorial ontologies: requirements, design, and content
- See the ontology in Turtle here

African Wildlife Ontology

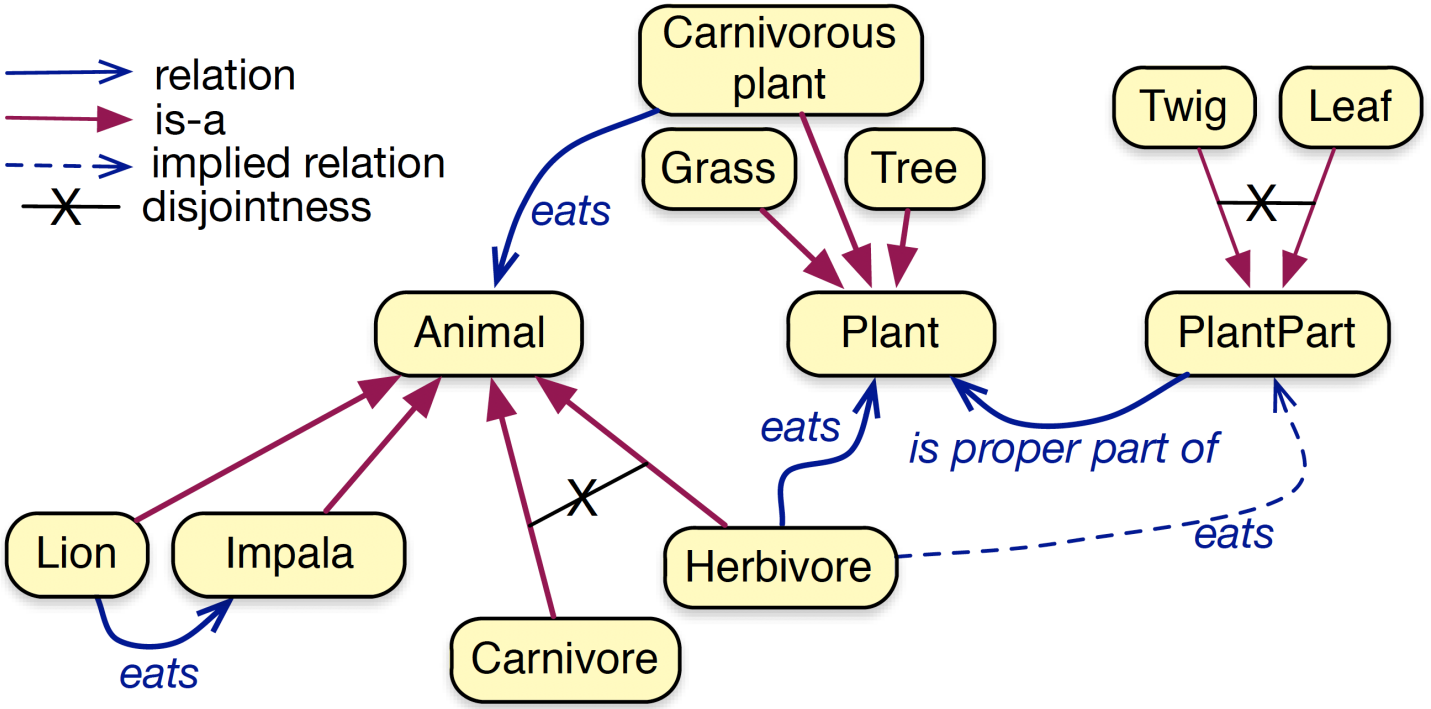
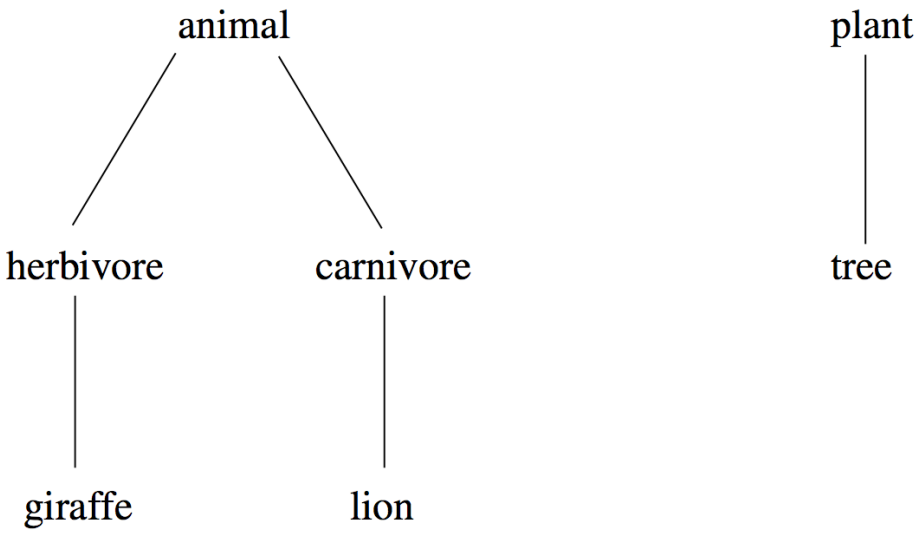
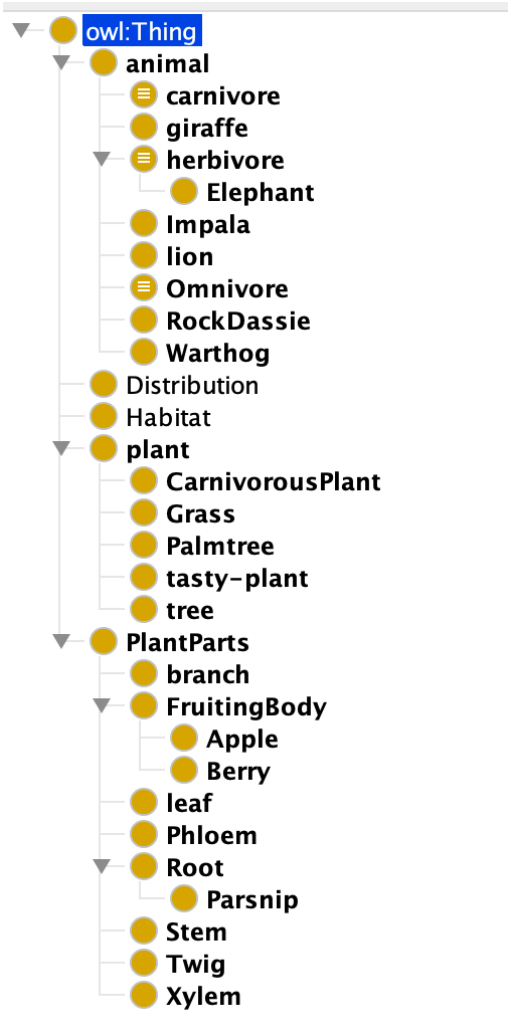


Figure 1 The African Wildlife Ontology at a glance. The main classes and relations of the African Wildlife ontology (v1) and an illustrative selection of its subclasses.

African Wildlife Ontology: Classes





See awo1.ttl

African Wildlife Ontology: Classes

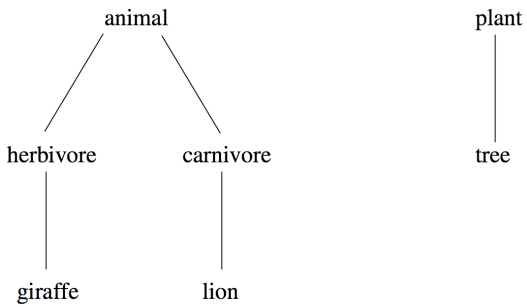
:animal owl:disjointWith :plant .

:herbivore rsds:subClassOf :animal;
owl:disjointWith :carnivore .

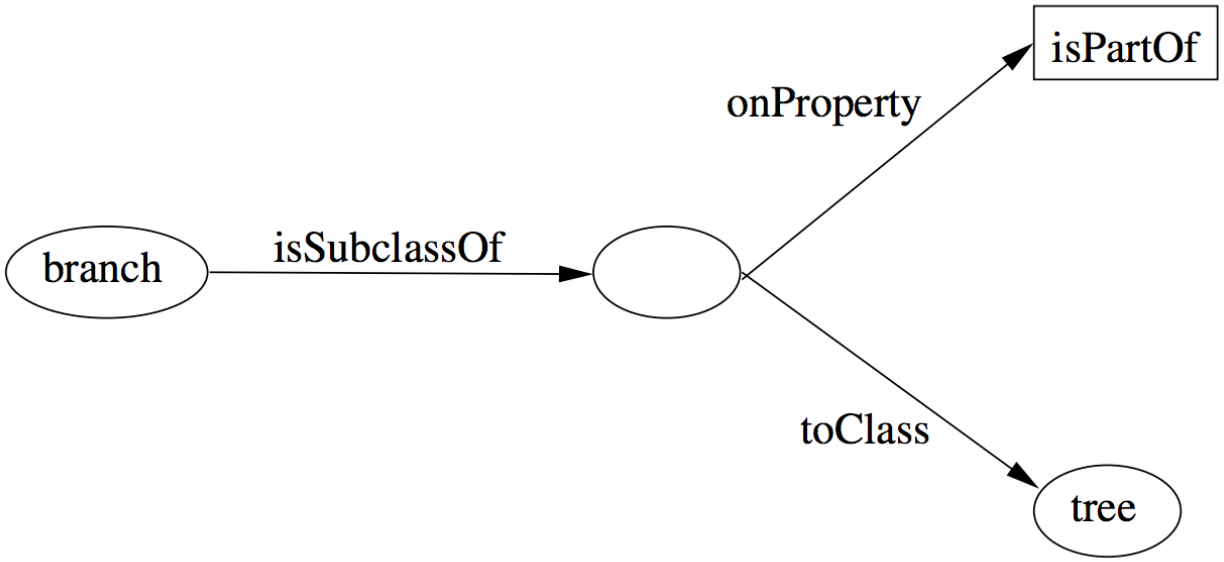
:giraffe rdfs:subClassOf :herbivore .

:carnivore rdfs:subClassOf :animal .

:lion rdfs:subClassOf :carnivore .



Branches are parts of trees



African Wildlife: Properties

 # e.g, hand part of arm, arm part of body
:isPartOf a owl:TransitiveProperty .

only animals eat things
:eats :domain :animal.

the inverse of :eats in :eatenBy
:eats owl:inverseOf :eatenBy.

An African Wildlife: Branches

```
# plants and animals are disjoint
:plant owl:disjointWith :animal

# trees are plants
:tree rdfs:subClassOf :plant

# branches are only parts of trees
:branch rdfs:subClassOf
  [a owl:Restriction;
    owl:allValuesFrom :tree
    owl:onProperty :isPartOf]
```

African Wildlife: Leaves

```
# leaves are only parts of branches
:leaf rdfs:subClassOf
  [a owl:Restriction;
    owl:allValuesFrom :branch
    owl:onProperty :isPartOf]
```

African Wildlife: Carnivores

```
# carnivores are exactly those animals
# that eat animals
:Carnivore owl:intersectionOf
    (:Animal,
        [a owl:Restriction;
            owl:someValuesFrom :Animal
            owl:onProperty :eats]
    ) .
```

Can carnivores eat plants?

African Wildlife: Herbivores

How can we define Herbivores?

African Wildlife: Herbivores

Here is a start

```
:herbivore a owl:Class;
  rdfs:comment "Herbivores are exactly those
      animals that eat only plants or parts of
      plants” .
```

African Wildlife: Herbivores

```
:Herbivore owl:equivalentClass
  [a owl:Class;
   owl:intersectionOf
    (:Animal
     [a owl:Restriction
      owl:onProperty :eats;
      owl:allValuesFrom
       [a owl:Class;
        owl:equivalentClass
         owl:unionOf
          (:Plant
           [a owl:Restriction;
            owl:onProperty :isPartOf;
            owl:allValuesFrom :Plant])]])]
```

African Wildlife: Giraffes

```
# giraffes are herbivores, and eat only leaves
Giraffe rdfs:subClassOf
    :Herbavore,
    [owl:Restriction
        owl:onProperty :eats;
        owl:allValues:From :Leaf] .
```

African Wildlife: Lions

 # Lions are animals that eat only herbivores

```
:lion rdfs:subClassOf
    :Carnivore,
    [a Restriction
       owl:onProperty :eats;
       owl:allValuesFrom :Herbavore] .
```

African Wildlife: Tasty Plants

#tasty plants are eaten both by herbivores & carnivores

??????????????

African Wildlife: Tasty Plants

```
#tasty plants are eaten both by herbivores & carnivores
:TastyPlant
    rdfs:subClassOf
        :Plant,
        [a Restriction
            owl:onProperty :eatenBy;
            owl:someValuesFrom :Herbavore],
        [a Restriction
            owl:onProperty :eatenBy;
            owl:someValuesFrom :Carnivore .]
```

Outline

1. A bit of history
2. Basic Ideas of OWL
3. The OWL Language
4. Examples
5. **Beyond OWL**

Modules and Imports

- 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

Closed World Assumption

- 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

Defaults and nonmonotonic reasoning

- Many practical knowledge representation systems allow inherited values to be overridden by more specific cases
  - treat inherited values as defaults
  - Assume a bird can fly, unless we know otherwise
- No consensus on the right formalization for the nonmonotonic behaviour of default values
- Some systems, like RDFox, support this along with truth maintenance