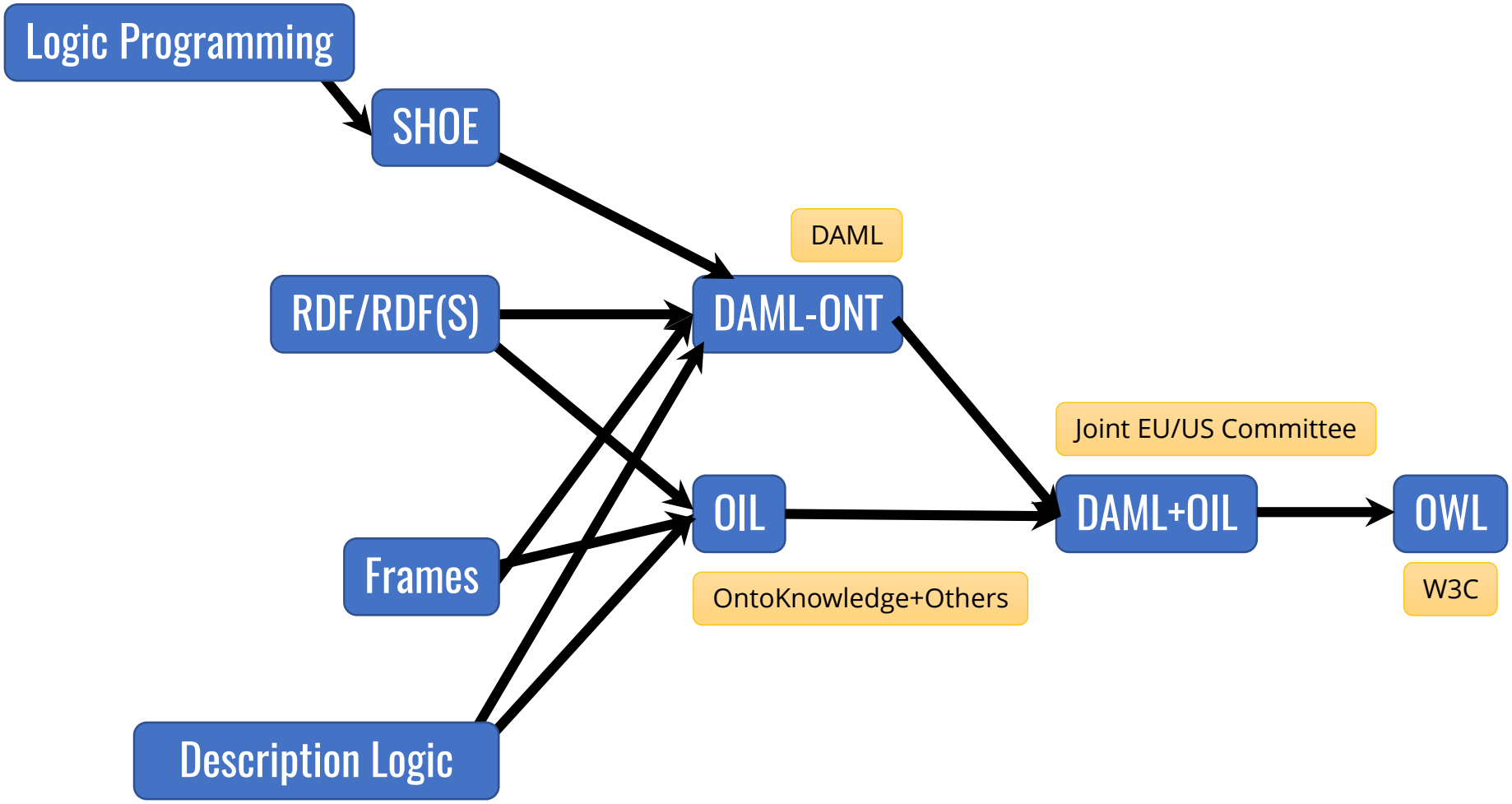
SEMANTIC WEB

08
OWL
ONTOLOGY WEB LANGUAGE

IMRAN IHSAN
ASSISTANT PROFESSOR, AIR UNIVERSITY, ISLAMABAD
WWW.IMRANIHSAN.COM

BASED ON SLIDES FROM GRIGORIS ANTONIOU AND FRANK VAN HARMELEN

THE OWL FAMILY TREE



A BRIEF HISTORY OF OWL: SHOE

- **S**imple **H**TML **O**ntology **E**xtensions

- Sean Luke, Lee Spector, and David Rager, 1996
  - SHOE allows World-Wide Web authors to annotate their pages with ontology-based knowledge about page contents. We present examples showing how the use of SHOE can support a new generation of knowledge-based search and knowledge discovery tools that operate on the World-Wide Web.

- Supported adding “semantic” tags defined in an ontology plus prolog-like rules to web pages.

A BRIEF HISTORY OF OWL: SHOE

```
<META HTTP-EQUIV="Instance-Key" CONTENT="http://www.cs.umd.edu/~george">

<USE-ONTOLOGY "our-ontology" VERSION="1.0"
                             PREFIX="our" URL="http://ont.org/our-ont.html">

…

<CATEGORY "our.Person">
<RELATION "our.firstName" TO="George">
<RELATION "our.lastName" TO="Cook">
<RELATION "our.marriedTo" TO="http://www.cs.umd.edu/~helena">
<RELATION "our.employee" FROM="http://www.cs.umd.edu">
```

A BRIEF HISTORY OF OWL: OIL

- Developed by group of (largely) European researchers
  - (several from EU OntoKnowledge project)
- Based on frame-based KR languages
- Strong emphasis on formal rigor
- Semantics in terms of description logics
- RDFS based syntax

A BRIEF HISTORY OF OWL: DAML-ONT

- **DAML: D**arpa **A**gent **M**arkup **L**anguage

- Developed by DARPA DAML Program
  - Largely US based researchers

- Extended RDFS with constructors from OO and frame-based languages
- Rather weak semantic specification
  - Problems with machine interpretation
  - Problems with human interpretation

A BRIEF HISTORY OF OWL: DAML+OIL

- Merging of DAML-ONT and OIL
- Basically a DL with an RDFS-based syntax
- Development was carried out by “Joint EU/US Committee on Agent Markup Languages”
- Extends (“DL subset” of) RDF
- Submitted to W3C as basis for standardisation
  - Web-Ontology (**WebOnt**) Working Group formed

A BRIEF HISTORY OF OWL: OWL

- W3C Recommendation (February 2004)
- Based largely on the March 2001 DAML+OIL specification
- Well defined RDF/XML serializations
- Formal semantics
  - First order logic
  - Relationship with RDF
- Comprehensive test cases for tools/implementations
- Growing industrial take up.

OWL 2

- Is an **extension of OWL**
  - Addresses deficiencies identified by users and developers (at **OWLED workshop**)
- Is based on more expressive DL: **SROIQ**
  - OWL is based on **SHOIN**
- W3C working group chartered
  - http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
  - Became a W3C recommendation October 2009
- **Supported** by popular OWL tools
  - **Protégé**, TopBraid, FaCT++, Pellet

ONTOLOGY AND DATA

- In philosophy, an ontology is a model of what exists in the world
  - (kinds of things, properties, etc.)
- In information systems, it's roughly a schema for information or data
- In a KR language, it's typically a model of classes and relations/properties and axioms that go with them (e.g., subPropertyOf is transitive)
- Data comprise information about individual instances expressed with terms in the ontology
  - Some instances might be considered part of the ontology
  - (e.g., God, George Washington, Baltimore)

REQUIREMENTS FOR ONTOLOGY LANGUAGES

- 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

EXPRESSIVE POWER VS EFFICIENT REASONING

- There's always a tradeoff between expressive power and efficient reasoning support
- The richer the language is, the more inefficient the reasoning support becomes
- Reasoning can be undecidable or semi-decidable and even if decidable can be exponentially hard
- We need a compromise:
  - A language supported by reasonably efficient reasoners
  - A language that can express large classes of ontologies and knowledge

KINDS OF REASONING ABOUT KNOWLEDGE

- **Class membership**
  - If x is an instance of a class C, and C is a subclass of D,
  - then we can infer that x is an instance of D
- **Equivalence of Classes**
  - If class A is equivalent to class B, and class B is equivalent to class C,
  - then A is equivalent to C, too
- **Consistency**
  - X instance of classes A and B, but A and B are disjoint
  - This is an indication of an error in the ontology or data
- **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

USES FOR REASONING

- Reasoning support is important for
  - Deriving new relations and properties
  - Automatically classifying instances in classes
  - Checking the consistency of the ontology and the knowledge
  - Checking for unintended relationships between classes

- Checks like these are valuable for
  - Designing large ontologies, where multiple authors are involved
  - Integrating and sharing ontologies from various sources

REASONING SUPPORT FOR OWL

- 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 and Pellet
- Description logics are a subset of predicate logic for which efficient reasoning support is possible

RDFS'S EXPRESSIVE POWER LIMITATIONS

- **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. male and female)
- **Boolean combinations of classes**
  - Sometimes we wish to build new classes by combining other classes using union, intersection, and complement
  - E.g. person is the disjoint union of the classes male and female

RDFS'S EXPRESSIVE POWER LIMITATIONS

- **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 "is mother of")
  - A property is the inverse of another property (like "eats" and "is eaten by")

COMBINING OWL WITH RDF SCHEMA

- 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

THREE SPECIES OF OWL 1

- W3C'sWeb Ontology Working Group defined OWL as three different sublanguages:
  - OWL Full
  - OWL DL
  - OWL Lite

- Each sublanguage geared toward fulfilling different aspects of requirements

OWL FULL

- It uses all the OWL languages primitives
- It allows the combination of these primitives in arbitrary ways with RDF and RDF Schema
- OWL Full is fully upward-compatible with RDF, both syntactically and semantically
- OWL Full is so powerful that it's undecidable
  - No complete (or efficient) reasoning support

SOUNDNESS AND COMPLETENESS

- A sound reasoner only makes conclusions that logically follow from the input, i.e., all of it's conclusions are correct
  - We almost always require our reasoners to be sound

- A complete reasoner can make all conclusions that logically follow from the input
  - We can not guarantee complete reasoners for full FOL (First Order Logic) and many subsets
  - We can't do it for OWL

OWL DL

- OWL DL (Description Logic) is a sublanguage of OWL Full that restricts application of the constructors from OWL and RDF
  - Application of OWL's constructors' to each other is disallowed
  - It corresponds to a well studied description logic

- OWL DL permits efficient reasoning support

- But we lose full compatibility with RDF:
  - Not every RDF document is a legal OWL DL document
  - Every legal OWL DL document is a legal RDF document

OWL LITE

- An even further restriction limits OWL DL to a subset of the language constructors
  - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality

- The advantage of this is a language that is easier to
  - grasp, for users
  - implement, for tool builders

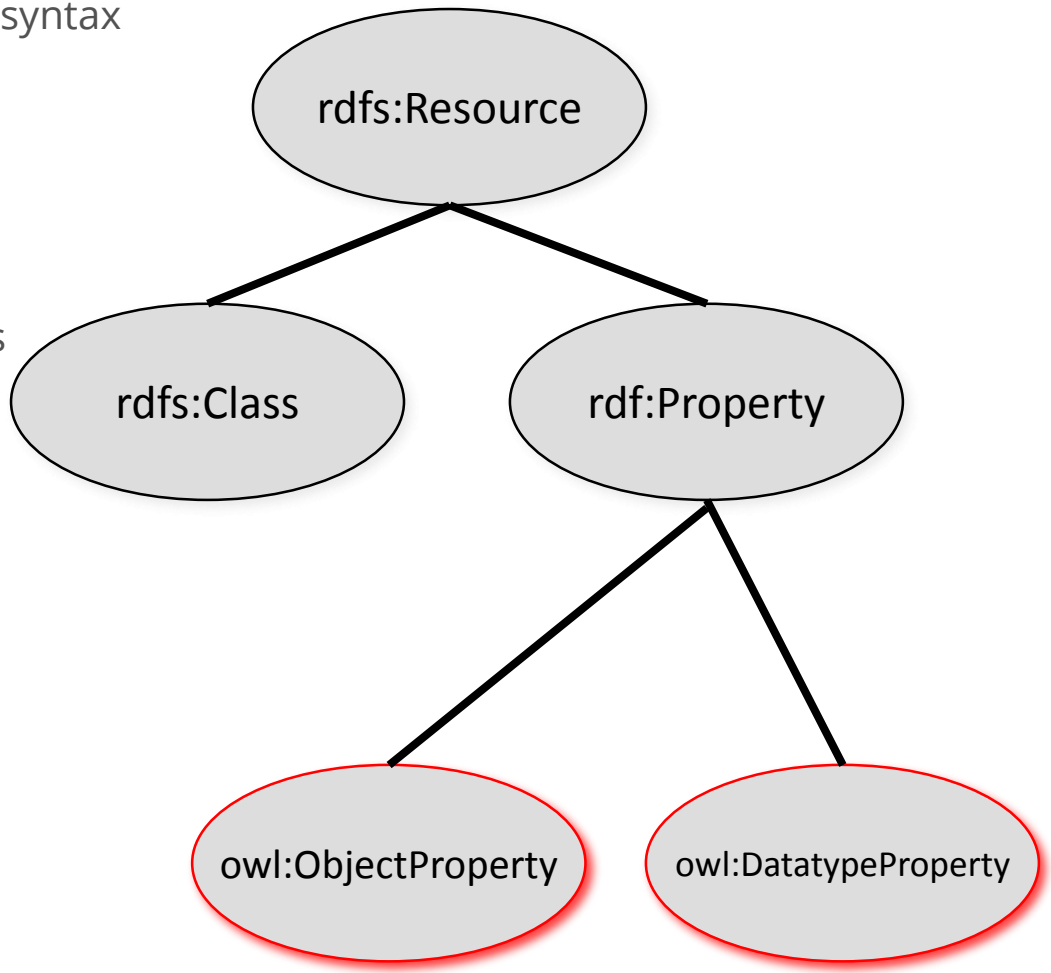
- The disadvantage is restricted expressivity

UPWARD COMPATIBILITY FOR OWL SPECIES

- 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 COMPATIBILITY WITH RDF SCHEMA

- All varieties of OWL use RDF for their syntax
- Instances are declared as in RDF, using RDF descriptions
- Typing information OWL constructors are specializations of their RDF counterparts



OWL SYNTACTIC VARIETIES

- OWL builds on RDF and uses RDF's XML-based syntax

- Other syntactic forms for OWL have also been defined:
  - An alternative, more readable XML-based syntax
  - RDF serializations – Turtle, Ntriples, etc.
  - Several abstract syntaxes that are more compact and readable than XML
  - A graphic syntax based on the conventions of UML

OWL XML/RDF SYNTAX: HEADER

```
<rdf:RDF
        xmlns:owl ="http://www.w3.org/2002/07/owl#"
        xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd ="http://www.w3.org/2001/XLMSchema#">
```

- OWL documents are RDF documents
- and start with a typical declaration of namespaces
- The W3C recommendation for owl has the namespace http://www.w3.org/2002/07/owl#"

OWL:ONTOLOGY

```
<owl:Ontology rdf:about="">
        <rdfs:comment>Example OWL ontology</rdfs:comment>
        <owl:priorVersion rdf:resource="http://www.mydomain.org/uni-ns-old"/>
        <owl:imports rdf:resource="http://www.-mydomain.org/persons"/>
        <rdfs:label>University Ontology</rdfs:label>
</owl:Ontology>
```

- **owl:imports,**
  - a transitive property, indicates that the document commits to all of the terms as defined in its target
- **owl:priorVersion** points to an earlier version of this document

OWL CLASSES

```
<owl:Class rdf:about="#associateProfessor">
        <owl:disjointWith rdf:resource="#professor"/>
        <owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>
```

- Classes are defined using **owl:Class**
  - **owl:Class** is a subclass of rdfs:Class

- Owl:Class is disjoint with datatypes (aka literals)
- Disjointness is defined using **owl:disjointWith**
  - Two disjoint classes are can share no instances

OWL CLASSES

```
<owl:Class rdf:ID="faculty">
        <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

- **owl:equivalentClass** defines equivalence of classes

- **owl:Thing** is the most general class, which contains everything
  - i.e., every owl class is rdfs:subClassOf owl:Thing

- **owl:Nothing** is the empty class
  - i.e., owl:NoThing is rdf:subClassOf every owl class

OWL PROPERTIES

- In OWL there are two kinds of properties

- **Object properties** relate objects to other objects
  - owl:ObjectProperty
  - E.g. is-TaughtBy, supervises

- **Data type properties** relate objects to datatype values
  - owl:DatatypeProperty
  - E.g. phone, title, age, etc.

- These were made separate to make it easier to create sound and complete reasoners

DATATYPE PROPERTIES

- OWL uses XML Schema data types, exploiting the layered architecture of the Semantic Web

```
<owl:DatatypeProperty rdf:ID="age">
        <rdfs:range
         rdf:resource="http://www.w3.org/2001/XLMSchema#nonNegativeInteger"/>
        <rdfs:domain rdf:resource="foaf:Person">
</owl:DatatypeProperty>
```

OWL OBJECT PROPERTIES

- Typically user-defined data types

```
<owl:ObjectProperty rdf:ID="isTaughtBy">
        <rdfs:domain rdf:resource="#course"/>
        <rdfs:range rdf:resource= "#academicStaffMember"/>
        <rdfs:subPropertyOf rdf:resource="#involves"/>
</owl:ObjectProperty>
```

INVERSE PROPERTIES

```
<owl:ObjectProperty rdf:ID="teaches">
        <rdfs:range rdf:resource="#course"/>
        <rdfs:domain rdf:resource= "#academicStaffMember"/>
        <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

- A partial list of axioms:

```
owl:inverseOf rdfs:domain owl:ObjectProperty;
              rdfs:range owl:ObjectProperty;
              a owl:SymmetricProperty.
{?P owl:inverseOf ?Q. ?S ?P ?O} => {?O ?Q ?S}.
{?P owl:inverseOf ?Q. ?P  rdfs:domain ?C} => {?Q rdfs:range ?C}.
{?A owl:inverseOf ?C. ?B owl:inverseOf ?C} => {?A rdfs:subPropertyOf ?B}.
```

EQUIVALENT PROPERTIES

```
<owl:equivalentProperty>
        <owl:ObjectProperty rdf:ID="lecturesIn">
        <owl:equivalentProperty rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

- Two properties have the same property extension
- Axioms

```
        {?A rdfs:subPropertyOf ?B. ?B rdfs:subPropertyOf ?A}
        <=> {?A owl:equivalentProperty ?B}.
```

PROPERTY RESTRICTIONS

- In OWL we can declare that the class C satisfies certain conditions
  - All instances of C satisfy the conditions

- This is equivalent to saying that C is subclass of a class C', where C collects all objects that satisfy the conditions
  - C' can remain anonymous

- Example:
  - People whose sex is male and have at least one child whose sex is female and whose age is six
  - Things with exactly two arms and two legs

PROPERTY RESTRICTIONS

- The **owl:Restriction** element describes such a class
- This element contains an **owl:onProperty** element and one or more **restriction declarations**
- One type defines **cardinality restrictions** (at least one, at most 3,...)
- The other type defines restrictions on the kinds of values the property may take
  - **owl:allValuesFrom** specifies universal quantification
  - **owl:hasValue** specifies a specific value
  - **owl:someValuesFrom** specifies existential quantification

OWL:ALLVALUESFROM

- Describe a class where all of the values of a property match some requirement
- E.g., Math courses taught by professors.

```
<!-- First year courses that are taught by professors -->
<owl:Class rdf:about="#firstYearCourse">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#isTaughtBy"/>
         <owl:allValuesFrom rdf:resource="#Professor"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

OFFSPRING OF PEOPLE ARE PEOPLE

```
<!- The offspring of a Person is a Person -->
<rdf:Description rdf:about="foaf:Person">
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="bio:offspring"/>
        <owl:allValuesFrom rdf:resource="foaf:Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

- Literally: Person is a sub-class of things all of whose offspring are necessarily of type Person

```
{?X a foaf:Person. ?X bio:offspring ?O} => {?O a Person}
```

OFFSPRING OF PEOPLE ARE PEOPLE

```
<rdf:RDF
  xmlns:="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:bio="http://example.com/bio/" >
<Description about="foaf:Person">
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty resource="bio:offspring" />
        <owl:allValuesFrom resource="foaf:Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</Description>
```

AND IN TURTLE

```
@prefix : <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

    <foaf:Person> a :Class;
         rdfs:subClassOf
             [ a :Restriction;
               :allValuesFrom <foaf:Person>;
               :onProperty <bio:offspring> ] .

# a thing of type owl:restriction
# where all of its bio:offspring values are of type foaf:Person
```

WHAT FOLLOWS?

```
<foaf:Person> rdfs:subClassOf

    [owl:allValuesFrom <foaf:Person>;
     owl:onProperty <bio:offspring>] .

???

:bio:offspring rdfs:domain :animal;

               rdfs:range :animal.

???

:alice a foaf:Person;

       bio:offspring :bob.

???

:carol a foaf:Person.

:don bio:offspring :carol.

???
```

WHAT FOLLOWS?

```
<foaf:Person> rdfs:subClassOf
    [owl:allValuesFrom <foaf:Person>;
     owl:onProperty <bio:sprungFrom>] .

bio:sprungFrom rdfs:domain :animal;
               rdfs:range :animal;
               owl:inverse bir:offspring.

:carol a foaf:Person.
:don bio:offspring :carol.
???
```

OWL:HASVALUE

- Describe a class with a particular value for a property
  - E.g., Math courses taught by Professor Longhair

```
<!- Math courses taught by #949352 ->
<owl:Class>
  <rdfs:subClassOf rdf:resource="#mathCourse”/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource= "#isTaughtBy"/>
      <owl:hasValue rdf:resource= "#949352"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

A TYPICAL EXAMPLE

```
:male    owl:equivalentClass
         owl:intersectionOf
              ( foaf:Person,
                   [:onProperty :sex;
                    :owl:hasValue “male”]
              ).
```

WHAT FOLLOWS?

```
:ed a :male?

???

:frank a foaf:Person; :sex “male”.

???

:gerry a foaf:Person; :sex “male”; :sex “female” .
```

OWL:SOMEVALUESFROM

- Describe a class based on a requirement that it must have at least one value for a property matching a description.
  - E.g., Academic staff members who teach an undergraduate course.

```
<owl:Class rdf:about="#academicStaffMember">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teaches"/>
      <owl:someValuesFrom rdf:resource="#undergraduateCourse"/>
    </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

CARDINALITY RESTRICTIONS

- We can specify minimum and maximum number using **owl:minCardinality** & **owl:maxCardinality**
  - Courses with fewer than 10 students
  - Courses with between 10 and 100 students
  - Courses with more than 100 students

- It is possible to specify a precise number by using the same minimum and maximum number
  - Courses with exactly seven students

- For convenience, OWL offers also **owl:cardinality**
  - E.g., exactly N

CARDINALITY RESTRICTIONS

- E.g. courses taught be at least two people.

```
<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
          2
       </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

WHAT DOES THIS SAY?

```
<owl:Class rdf:ID="Parent">
   <owl:equivalentClass>
       <owl:Restriction>
           <owl:onProperty rdf:resource="#hasChild" />
           <owl:minCardinality rdf:datatype=
              "&xsd;nonNegativeInteger">1</owl:minCardinality>
       </owl:Restriction>
   </owl:equivalentClass>
</owl:Class>
```

DEFINITION OF A PARENT

- The parent class is equivalent to the class of things that have at least one child

$$\text{All(x): Parent(x)} \leftrightarrow \text{Exists(y) hasChild(x, y)}$$

- If hasChild is defined as having Person as it's domain, then Parents are also people.

SPECIAL PROPERTIES

- **owl:TransitiveProperty** (transitive property)
  - E.g. “has better grade than”, “is ancestor of”
- **owl:SymmetricProperty** (symmetry)
  - E.g. “has same grade as”, “is sibling of”
- **owl:FunctionalProperty** defines a property that has at most one value for each object
  - E.g. “age”, “height”, “directSupervisor”
- **owl:InverseFunctionalProperty** defines a property for which two different objects cannot have the same value

SPECIAL PROPERTIES

```
<owl:ObjectProperty rdf:ID="hasSameGradeAs">
   <rdf:type rdf:resource="&owl;TransitiveProperty"/>
   <rdf:type rdf:resource="&owl;SymmetricProperty"/>
   <rdfs:domain rdf:resource="#student"/>
   <rdfs:range rdf:resource="#student"/>
</owl:ObjectProperty>
```

BOOLEAN COMBINATIONS

- We can combine classes using Boolean operations (union, intersection, complement)
- Negation is introduced by the complementOf, e.g., courses not taught be staffMembers

```
<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teaches"/>
      <owl:allValuesFrom>
        <owl:complementOf rdf:resource="#staffMember"/>
      <owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

BOOLEAN COMBINATIONS

- The new class is not a subclass of the union, but rather equal to the union
  - We have stated an equivalence of classes

- E.g., university people is the union of staffMembers and Students

```
<owl:Class rdf:ID="peopleAtUni">
        <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#staffMember"/>
                <owl:Class rdf:about="#student"/>
        </owl:unionOf>
</owl:Class>
```

BOOLEAN COMBINATIONS

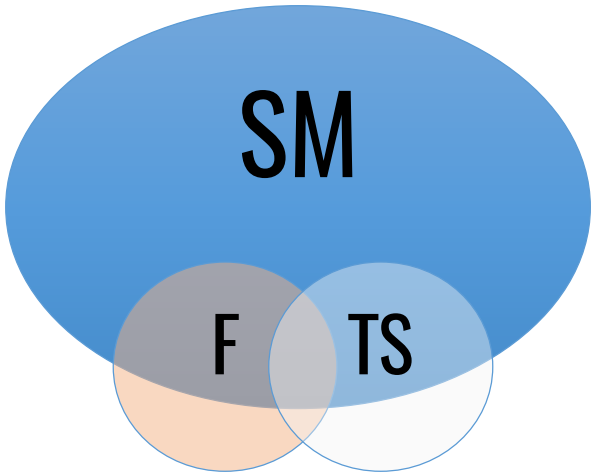
- E.g., CS faculty is the intersection of faculty and things that belongsTo the CS Department.

```
<owl:Class rdf:ID="facultyInCS">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#faculty"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#belongsTo"/>
      <owl:hasValue rdf:resource="#CSDepartment"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

NESTING OF BOOLEAN OPERATORS

- E.g., administrative staff are staff members who are not faculty or technical staff members

```
<owl:Class rdf:ID="adminStaff">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#staffMember"/>
    <owl:complementOf>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#faculty"/>
        <owl:Class rdf:about="#techSupportStaff"/>
      </owl:unionOf>
   </owl:complementOf>
 </owl:intersectionOf>
</owl:Class>
```



ENUMERATIONS WITH OWL:ONEOF

- E.g., a thing that is either Monday, Tuesday, …

```
<owl:oneOf rdf:parseType="Collection">
  <owl:Thing rdf:about="#Monday"/>
  <owl:Thing rdf:about="#Tuesday"/>
  <owl:Thing rdf:about="#Wednesday"/>
  <owl:Thing rdf:about="#Thursday"/>
  <owl:Thing rdf:about="#Friday"/>
  <owl:Thing rdf:about="#Saturday"/>
  <owl:Thing rdf:about="#Sunday"/>
</owl:oneOf>
```

DECLARING INSTANCES

- Instances of classes are declared as in RDF, as in these examples

```
<rdf:Description rdf:ID="949352">
  <rdf:type rdf:resource="#academicStaffMember"/>
</rdf:Description>

<academicStaffMember rdf:ID="949352">
  <uni:age rdf:datatype="&xsd;integer">
    39
  <uni:age>
</academicStaffMember>
```

NO UNIQUE-NAMES ASSUMPTION

- OWL does not adopt the unique-names assumption of database systems
  - That two instances have a different name or ID does not imply that they are different individuals

- Suppose we state that each course is taught by at most one staff member, and that a given course is taught by #949318 and is taught by #949352
  - An OWL reasoner does not flag an error
  - Instead it infers that the two resources are equal

DISTINCT OBJECTS

- To ensure that different individuals are indeed recognized as such, we must explicitly assert their inequality:

```
<lecturer rdf:about="949318">
  <owl:differentFrom rdf:resource="949352"/>
</lecturer>
```

DISTINCT OBJECTS

- OWL provides a shorthand notation to assert the pairwise inequality of all individuals in a given list

```
<owl:allDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <lecturer rdf:about="949318"/>
    <lecturer rdf:about="949352"/>
    <lecturer rdf:about="949111"/>
  </owl:distinctMembers>
</owl:allDifferent>
```

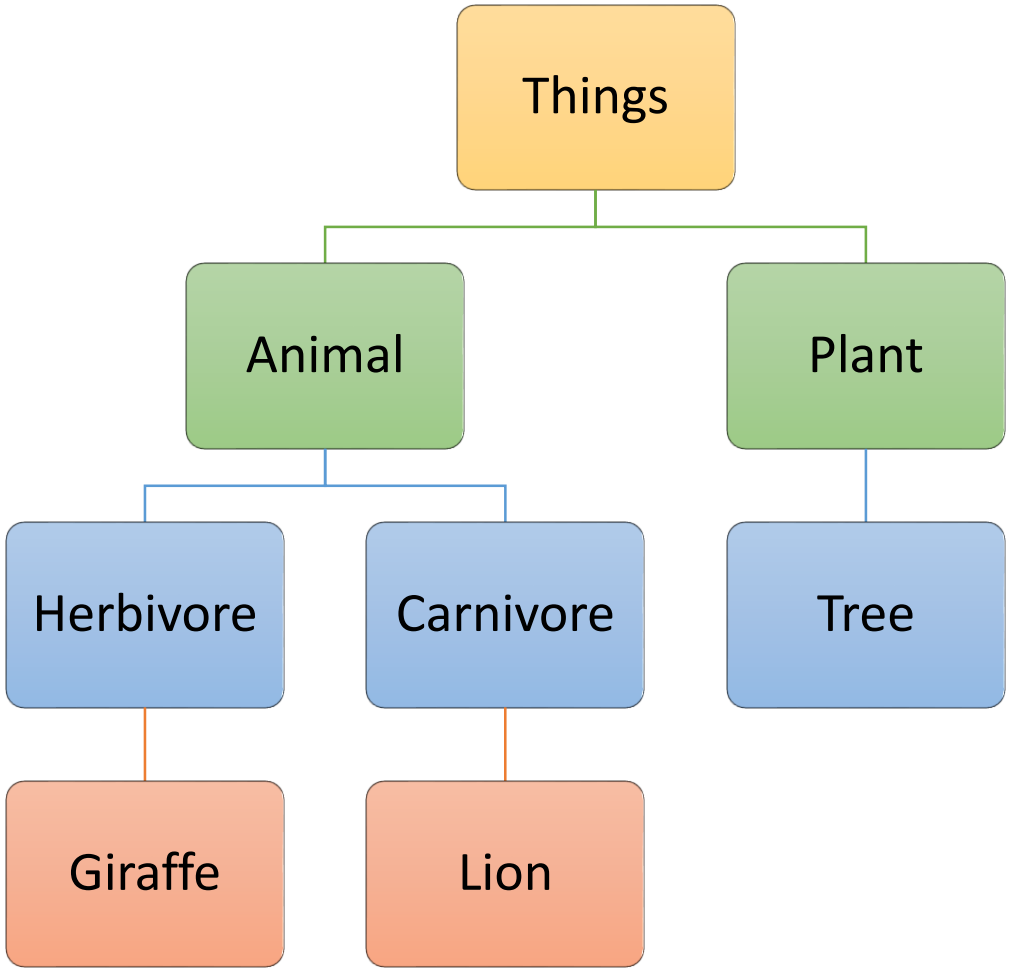
DATA TYPES IN OWL

- XML Schema provides a mechanism to construct user-defined data types
  - E.g., the data type of adultAge includes all integers greater than 18

- Such derived data types cannot be used in OWL
- The OWL reference document lists all the XML Schema data types that can be used
- These include the most frequently used types such as **string**, **integer**, **Boolean**, **time**, and **date**.

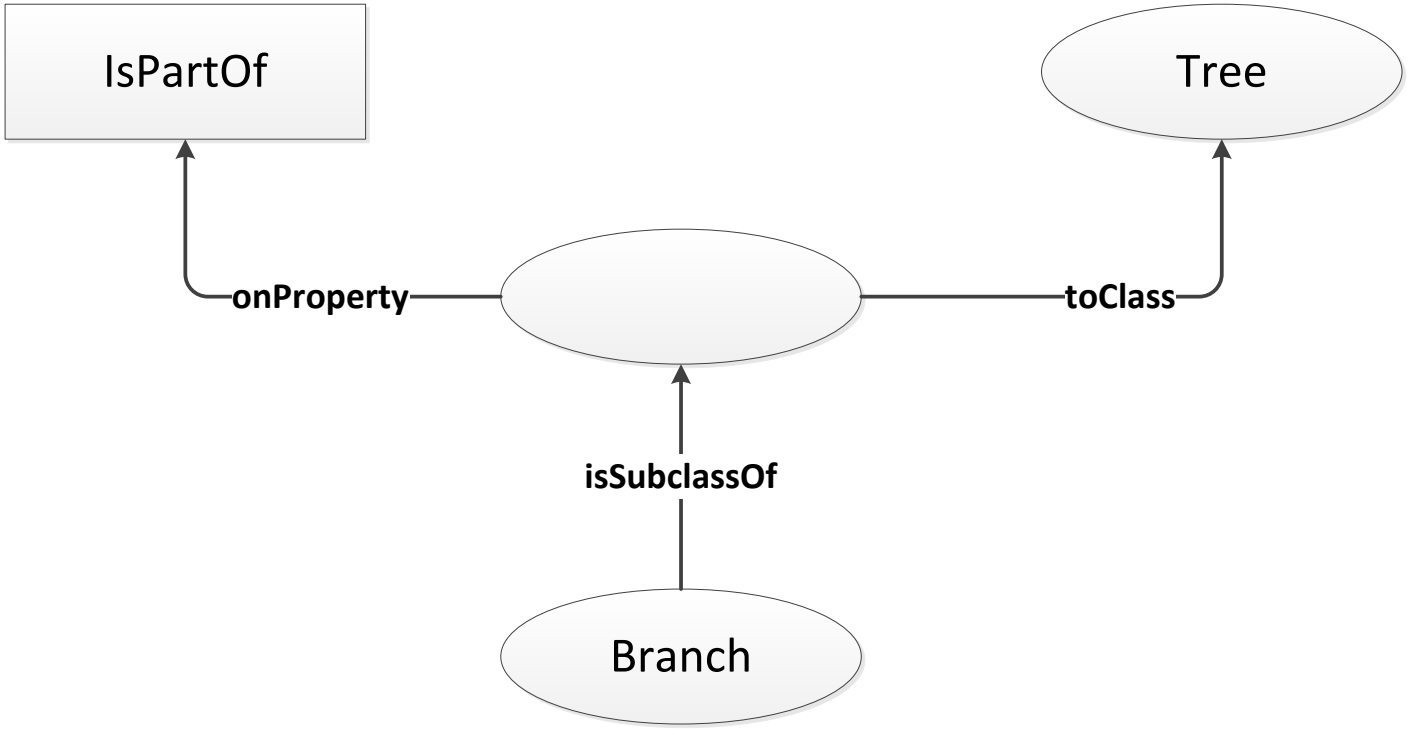
EXAMPLES

AFRICAN WILDLIFE ONTOLOGY: CLASSES



AFRICAN WILDLIFE: SCHEMATIC REPRESENTATION

- Branches are parts of trees



AFRICAN WILDLIFE: PROPERTIES

```
<owl:TransitiveProperty rdf:ID="is-part-of"/>

<owl:ObjectProperty rdf:ID="eats">
        <rdfs:domain rdf:resource="#animal"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="eaten-by">
        <owl:inverseOf rdf:resource="#eats"/>
</owl:ObjectProperty>
```

AFRICAN WILDLIFE: PLANTS AND TREES

```
<owl:Class rdf:ID="plant">
        <rdfs:comment>Plants are disjoint from animals. </rdfs:comment>
        <owl:disjointWith="#animal"/>
</owl:Class>

<owl:Class rdf:ID="tree">
        <rdfs:comment>Trees are a type of plant.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#plant"/>
</owl:Class>
```

AN AFRICAN WILDLIFE: BRANCHES

```
<owl:Class rdf:ID="branch">
  <rdfs:comment>Branches are parts of trees. </rdfs:comment>
  <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="#is-part-of"/>
        <owl:allValuesFrom rdf:resource="#tree"/>
     </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

AFRICAN WILDLIFE: LEAVES

```
<owl:Class rdf:ID="leaf">
  <rdfs:comment>Leaves are parts of branches. </rdfs:comment>
  <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="#is-part-of"/>
        <owl:allValuesFrom rdf:resource="#branch"/>
     </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

AFRICAN WILDLIFE: CARNIVORES

```
<owl:Class rdf:ID="carnivore">
  <rdfs:comment>
        Carnivores are exactly those animals that eat animals.
  </rdfs:comment>
  <owl:intersectionOf rdf:parsetype="Collection">
    <owl:Class rdf:about="#animal"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats"/>
      <owl:someValuesFrom rdf:resource="#animal"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

QUIZ

How can we define Herbivores?

AFRICAN WILDLIFE: HERBIVORES

HERE IS ONE APPROACH

```
Here is one approach


<owl:Class rdf:ID="herbivore">
    <rdfs:comment>
      Herbivores are exactly those animals that
         eat only plants or parts of plants.
    </rdfs:comment>
    ??????????????????
</owl:Class>
```

AFRICAN WILDLIFE: HERBIVORES

```
<owl:Class rdf:ID="herbivore">
 <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf=about="#animal"/>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#eats"/>
    <owl:allValuesFrom>
      <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:resource="plant"/>
          <owl:Restriction>
            <owl:onProperty rdf;resource="#is_part_of"/>
            <owl:allValuesFrom rdf:resource="#plant"/>
           </owl:Restriction>
        </owl:unionOf>
      </class>
    </owl:allValuesFrom>
   </owl:Restrcition>
 </owl:intersectionOf>
</owl:Class>
```

AFRICAN WILDLIFE: GIRAFFES

```
<owl:Class rdf:ID="giraffe">
        <rdfs:comment>Giraffes are herbivores, and they
        eat only leaves.</rdfs:comment>
        <rdfs:subClassOf rdf:type="#herbivore"/>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#eats"/>
                        <owl:allValuesFrom rdf:resource="#leaf"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

AFRICAN WILDLIFE: LIONS

```
<owl:Class rdf:ID="lion">
        <rdfs:comment>Lions are animals that eat
        only herbivores.</rdfs:comment>
        <rdfs:subClassOf rdf:type="#carnivore"/>
        <rdfs:subClassOf>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#eats"/>
                  <owl:allValuesFrom rdf:resource="#herbivore"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

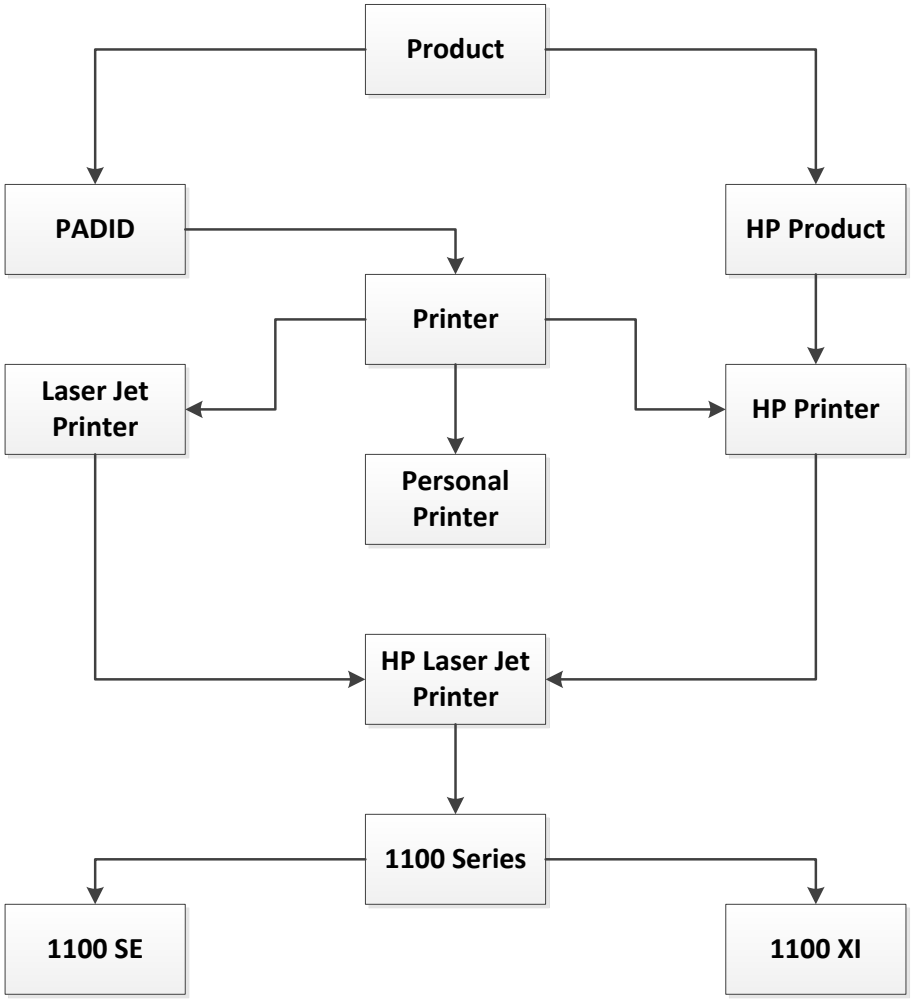
AFRICAN WILDLIFE: TASTY PLANTS

```
<owl:Class rdf:ID="tasty-plant">
        <rdfs:comment>
           Plants eaten both by herbivores and carnivores
        </rdfs:comment>
        <rdfs:comment>
                ??????????????
        <rdfs:comment>
</owl:Class>
```

AFRICAN WILDLIFE: TASTY PLANTS

```
<owl:Class rdf:ID="tasty-plant">
  <rdfs:subClassOf rdf:resource="#plant"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eaten_by"/>
      <owl:someValuesFrom> <owl:Class rdf:about="#herbivore"/>
      </owl:someValuefrom>
     </owl:Restriction>
  </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eaten_by"/>
      <owl:someValuesFrom> <owl:Class rdf:about="#carnivore"/>
        </owl:someValuefrom>
     </owl:Restriction>
   </rdfsSublassOf>
</owl:Class>
```

PRINTER ONTOLOGY – CLASS HIERARCHY



PRINTER ONTOLOGY – PRODUCTS AND DEVICES

```
<owl:Class rdf:ID="product">
        <rdfs:comment>Products form a class. </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="padid">
        <rdfs:comment>Printing and digital imaging devices
        form a subclass of products.</rdfs:comment>
        <rdfs:label>Device</rdfs:label>
        <rdfs:subClassOf rdf:resource="#product"/>
</owl:Class>
```

PRINTER ONTOLOGY – HP PRODUCTS

```
<owl:Class rdf:ID="hpProduct">
        <owl:intersectionOf>
                <owl:Class rdf:about="#product"/>
                <owl:Restriction>
                   <owl:onProperty rdf:resource="#manufactured-by"/>
                   <owl:hasValue>
                         <xsd:string rdf:value="Hewlett Packard"/>
                   </owl:hasValue>
                </owl:Restriction>
        </owl:intersectionOf>
</owl:Class>
```

PRINTER ONTOLOGY – PRINTERS & PERSONAL PRINTERS

```
<owl:Class rdf:ID="printer">

        <rdfs:comment>Printers are printing and digital
                        imaging devices.</rdfs:comment>

        <rdfs:subClassOf rdf:resource="#padid"/>

</owl:Class>


<owl:Class rdf:ID="personalPrinter">

        <rdfs:comment>Printers for personal use form
                        a subclass of printers.</rdfs:comment>

        <rdfs:subClassOf rdf:resource="#printer"/>

</owl:Class>
```

HP LASERJET 1100SE PRINTERS

```
<owl:Class rdf:ID="1100se">
        <rdfs:comment>1100se printers belong to the 1100 series and cost
                        $450.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#1100series"/>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#price"/>
                        <owl:hasValue><xsd:integer rdf:value="450"/>
                        </owl:hasValue>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

A PRINTER ONTOLOGY – PROPERTIES

```
<owl:DatatypeProperty rdf:ID="manufactured-by">
        <rdfs:domain rdf:resource="#product"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="printingTechnology">
        <rdfs:domain rdf:resource="#printer"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```