Lecture 8

OWL (continued)

Four Highlights making OWL a powerful language on the top of RDFS:

I. The W3C Web Ontology Language (OWL) is **a Semantic Web language** designed to represent **rich and complex knowledge** about **things, groups of things, and relations between things.**

II. OWL is a **computational logic-based language** such that knowledge expressed in OWL **can be exploited by computer programs**, e.g., **to verify the consistency of that knowledge or to make implicit knowledge explicit.**

III. OWL documents, **known as ontologies**, can be published in the World Wide Web **and may refer to or be referred from other OWL ontologies.**

IV. OWL is part of the W3C's Semantic Web technology stack, which includes RDF, RDFS, SPARQL, etc.

Source https://www.w3.org/OWL/

Four Highlights of OWL:

The current version of OWL, also referred to as **"OWL 2",** was developed by the [W3C OWL Working Group] (now closed) and published in 2009, with a Second Edition published in **2012**.

OWL 2 is **an extension and revision** of the 2004 version of OWL developed by the [W3C Web Ontology Working Group] (now closed) and published in 2004.

The deliverables that make up the OWL 2 specification include a Document Overview, which serves as an introduction to OWL 2, **describes the relationship between OWL 1 and OWL 2**, and provides an entry point to the remaining deliverables via a Documentation Roadmap.

Source https://www.w3.org/OWL/

Enumerations with owl:oneOf

```
<owl:Class rdf:ID="weekdays">
    <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>
</owl:Class>
```

Lecture Outline

1. Basic Ideas of OWL
2. **The OWL Language**
3. Examples
4. The OWL Namespace
5. Future Extensions

Non-Unique-Names

OWL **does not adopt** the unique-names assumption **of database systems**

- If 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 two staff members
  - An OWL reasoner **does not** flag an error
  - **Instead** it infers that the two resources are equal

Unique-name assumption

- **In general**, when two individuals are known by different names, **sometimes** it is assumed they are different individuals.
- This is an assumption that sometimes works (ex. Product codes), and **sometimes doesn't** (ex. Social environment)
- OWL does not make the unique-name assumption.

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 (2)

- 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 datatypes**
  - 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**.

Restriction of Features in OWL

- Property Separation
  - The set of **object properties** and **data type properties are disjoint**
  - **Therefore the following can never be specified for data type properties:**

owl:inverseOf

owl:FunctionalProperty

owl:InverseFunctionalProperty

owl:SymmetricProperty

Inheritance in Class Hierarchies

- Range restriction: **Courses must be taught by academic staff members only**
- Michael Maher is a professor
- He **inherits** the ability to teach from the class of academic staff members
- This is done in RDF Schema by fixing the semantics of "is a subclass of"
  - **It is not the responsibility of an application (RDF processing software) to interpret "is a subclass of**

- OWL is based on Description Logic
- Description Logic is **a fragment** of first-order logic
- OWL **inherits** from Description Logic two assumptions:
  I. The open-world assumption
  II. The non-unique-name assumption

Open-world assumption (OWA)

- We cannot conclude some statement x to be false simply because we cannot show x to be true
- We may not deduce falsity from the absence of truth

Open-world assumption example

- **Question:** "Did it rain in Tokyo yesterday?"
- **Answer:** "I don't know that it rained , but that's not enough reason to conclude that it didn't rain"

Closed-world assumption (CWA)

- Closed-world assumption allow deriving falsity from the inability to derive truth
- Example:
  - **Question:** " Was there a big earthquake disaster in Tokyo yesterday?"
  - **Answer:** " I don't know that there was, but if there had been such a disaster, I'd have heard about it. Therefore I conclude that there wasn't such a disaster"

OWL vs Databases

- Systems such as databases have tended to support closed worlds and unique names

Whereas

- Knowledge representation systems and theorem plovers support open worlds and non-unique names

QUESTION

By the use of **two examples**, one in databases and the other in web, describe how a database uses closed world and web uses open world assumptions.

ANSWER

Web present data about **uncounted concepts**, Consider the following statement: “Tim Smith with the ID of 785654549 is a citizen of France.” Now, what if you ask “Is Tim Smith with the ID of 785654549 a citizen of Malaysia?” Under a closed world assumption, the answer is “no” whereas under the open world assumption the answer is “not known”. Web is full of such kinds of data. Hence, open world assumption is used in a system with incomplete information, to which Web belongs.

On the other hand **databases** usually presents data about **limited concepts**, say a university. For instance, if the database of UOW shows that Tim Smith with the ID of 785654549 is not UOW’s student, **definitely** he is not UOW’s student.

Moreover in web non-unique names are used, and two different ID may show the same resource, whereas in database this not the case

Lecture Outline

1. Basic Ideas of OWL
2. The OWL Language
3. **Examples**
4. The OWL Namespace
5. Future Extensions

An African Wildlife Ontology – Class Hierarchy

An African Wildlife Ontology – Properties

```
<owl:TransitivePropertyrdf:ID="is-part-of"/>
<owl:ObjectPropertyrdf:ID="eats">
    <rdfs:domainrdf:resource="#animal"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="eaten-by">
    <owl:inverseOfrdf:resource="#eats"/>
</owl:ObjectProperty>
```

An African Wildlife Ontology – Plants and Trees

```
<owl:Class rdf:ID="plant">
    <rdfs:comment>Plants form a class disjoint from
    animals. </rdfs:comment>
    <owl:disjointWith rdf:resource="#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 Ontology – 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>
```

An African Wildlife Ontology – 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>
```

An African Wildlife Ontology – 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>
```

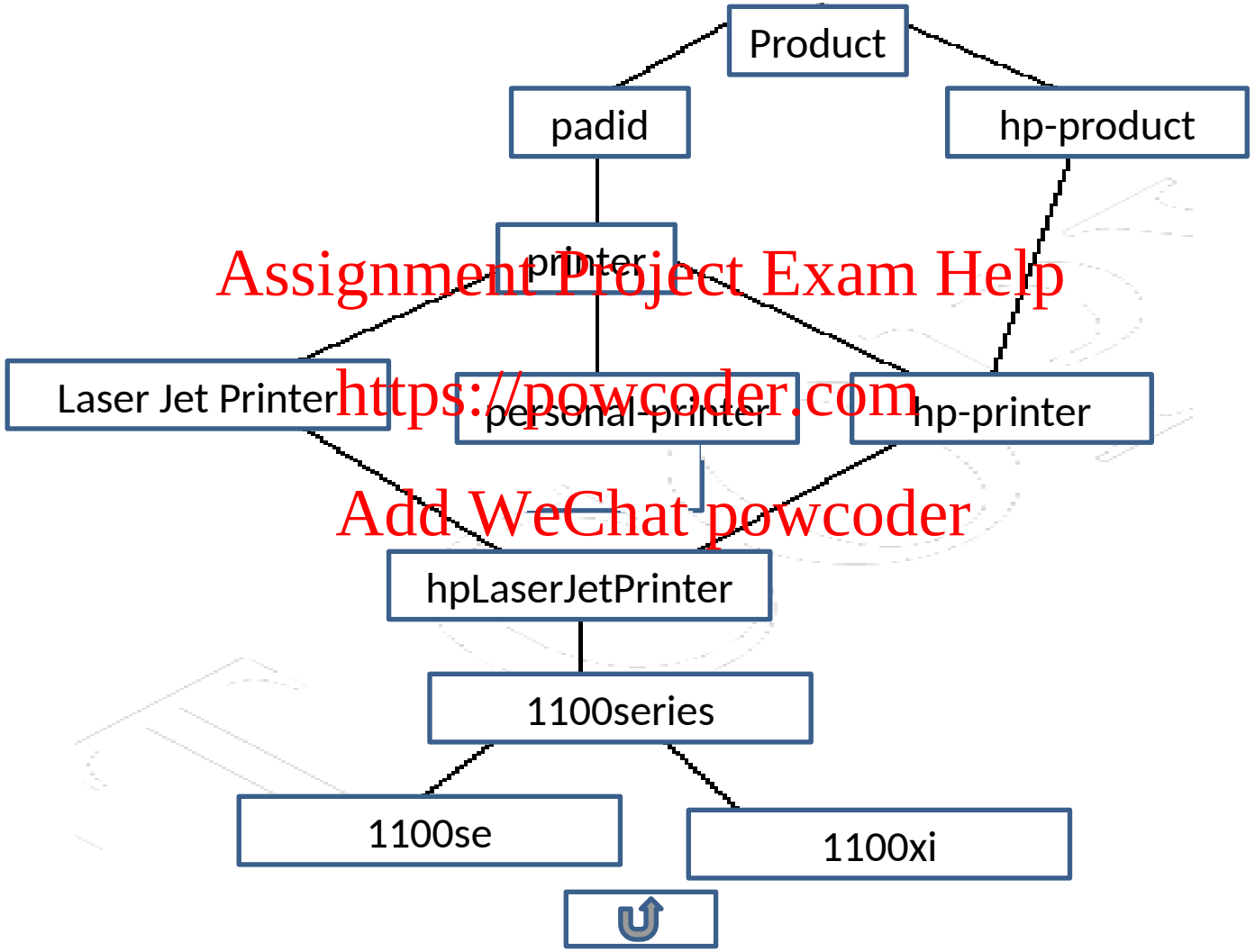
An African Wildlife Ontology – Giraffes

```
<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>
```

An African Wildlife Ontology – 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>
```

A Printer Ontology – Class Hierarchy



OWL for the Printer Ontology

```
<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>
```


The Printer Ontology – HP Products

```
<owl:Class rdf:ID="hpProduct">
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#product" />
        <owl:Restriction>
          <owl:onProperty
    rdf:resource="#manufactured_by" />
            <owl:hasValue rdf:datatype="&xsd;string">
                Hewlett Packard
            </owl:hasValue>
        </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```


The Printer Ontology – 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>
```


The Printer Ontology – 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 rdf:datatype="&xsd;integer">
                450
            </owl:hasValue>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

The 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>
```


Lecture Outline

1. Basic Ideas of OWL
2. The OWL Language
3. Examples
4. **The OWL Namespace**
5. Future Extensions

OWL in OWL

- We present a part of the definition of OWL in terms of itself
- The following captures some of OWL's meaning in OWL
  - It does **not** capture the entire semantics
  - A separate semantic specification is necessary
- The URI of the OWL definition is defined as the default namespace

Classes of Classes (Metaclasses)

- The class of all OWL classes is itself a subclass of the class of all RDF Schema classes:

note that this definition has been stated in OWL document:


```
<rdfs:Class rdf:ID="Class">
    <rdfs:label>Class</rdfs:label>
    <rdfs:subClassOf rdf:resource="&rdfs;Class"/>
</rdfs:Class>
```

Classes of Classes (Metaclasses) – Thing and Nothing (2)

```
<Class rdf:ID="Thing">
        <rdfs:label>Thing</rdfs:label>
        <unionOf rdf:parseType="Collection">
            <Class rdf:about="#Nothing"/>
            <Class>
                <complementOf
    rdf:resource="#Nothing" />
            </Class>
        </unionOf>
</Class>

<Class rdf:ID="Nothing">
        <rdfs:label>Nothing</rdfs:label>
        <complementOf rdf:resource="#Thing"/>
</Class>
```

Class Disjointness

```
<rdf:Property rdf:ID="disjointWith">
        <rdfs:label>disjointWith</rdfs:label>
        <rdfs:domain rdf:resource="#Class"/>
        <rdfs:range rdf:resource="#Class"/>
</rdf:Property>
```

Equality and Inequality

- Equality and inequality can be stated between arbitrary things.

  - In OWL Full this statement can also be applied to classes

- Properties **sameIndividualAs**, **sameAs** and **differentFrom**

Equality and Inequality (2)

```
<rdf:Property rdf:ID="sameIndividualAs">
        <rdfs:domain rdf:resource="#Thing"/>
        <rdfs:range rdf:resource="#Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="differentFrom">
    <EquivalentProperty rdf:resource=
    "#sameIndividualAs"/>
</rdf:Property>
```

Properties (3)

- **owl:inverseOf** relates two object properties:


```
<rdf:Property rdf:ID="inverseOf">
        <rdfs:label>inverseOf</rdfs:label>
        <rdfs:domain rdf:resource="#ObjectProperty"/>
        <rdfs:range rdf:resource="#ObjectProperty"/>
</rdf:Property>
```

Lecture Outline

1. Basic Ideas of OWL
2. The OWL Language
3. Examples
4. The OWL Namespace
5. **Future Extensions**

Future Extensions of OWL is involved with working on:

- Modules and Imports
- Defaults
- Closed World Assumption
- Unique Names Assumption
- Procedural Attachments
- Rules for Property Chaining

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
  - An open question is that **how to define appropriate module mechanism for Web ontology languages**

Defaults

- Many practical knowledge representation systems allow inherited values to be overridden by more specific classes in the hierarchy
  - treat inherited values as defaults
- No consensus has been reached on the right formalization for the nonmonotonic behaviour of default values

Closed World Assumption

- OWL currently adopts the open-world assumption: we cannot conclude some statement x to be false simply because we cannot show x to be true
- Closed-world assumption: a statement is true when its negation cannot be proved
  - tied to the notion of defaults, leads to nonmonotonic behaviour

Summary

- OWL is the proposed standard for Web ontologies
- OWL builds upon RDF and RDF Schema:
  - (XML-based) RDF syntax is used
  - Instances are defined using RDF descriptions
  - Most RDFS modeling primitives are used