Semantic Web

Tutorial 5-6

Iryna Dubrovska

People and Knowledge Networks

Tutorial 5

Task 1. OWL Ontology

1.1 Consider the following statements:

- Tiger is a sub-class of class Animal.
- Predator is a class whose members are exactly those animals who eat other animals.
- MatureTiger is a class whose members are exactly those tigers that are older than 4 years.
- Mature tigers may have children who are also tigers.

Write an OWL ontology that models the statements.

https://www.w3schools.com/xml/xml_elements.asp

Task 1. OWL Ontology

```
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Animal"/>
<owl:Class rdf:ID="Predator">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Animal"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="eats"/>
          </owl:onProperty>
          <owl:someValuesFrom>
            <owl:Class rdf:ID="Animal"/>
          </owl:someValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="MatureTiger">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Tiger"/>
        <owl:Restriction>
          <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
            4
          </owl:minCardinality>
          <owl:onProperty>
            <owl:DatatypeProperty rdf:ID="age"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Task 1. OWL Ontology

```
    <owl:Class rdf:about="#Tiger">
      <rdfs:subClassOf rdf:resource="#Animal"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="hasChild">
      <rdfs:range rdf:resource="#Tiger"/>
      <rdfs:domain rdf:resource="#MatureTiger"/>
      <owl:inverseOf>
        <owl:ObjectProperty rdf:ID="hasParent"/>
      </owl:inverseOf>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="#hasParent">
      <rdfs:range rdf:resource="#MatureTiger"/>
      <owl:inverseOf rdf:resource="#hasChild"/>
      <rdfs:domain rdf:resource="#Tiger"/>
    </owl:ObjectProperty>
    <owl:DatatypeProperty rdf:about="#age">
      <rdfs:domain rdf:resource="#Animal"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    </owl:DatatypeProperty>
    <owl:ObjectProperty rdf:about="#eats">
      <rdfs:range rdf:resource="#Animal"/>
      <rdfs:domain rdf:resource="#Animal"/>
    </owl:ObjectProperty>
  </rdf:RDF>
```

Task 1. OWL Ontology

1.2 The statements above can be seen as a T-Box. Your task is to define a simple A-Box of a tiger and model it in OWL according to your ontology from 1.1. Two or three statements are enough for the A-Box.
Check the validity of XML according to its schema. Point out the issues and rewrite XML accordingly.

```
<Tiger rdf:ID="Tommy">
  <age rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</age>
  <hasParent>
    <MatureTiger rdf:ID="Anna">
      <hasChild rdf:resource="#Tommy"/>
    </MatureTiger>
  </hasParent>
</Tiger>
</rdf:RDF>
```

Task 2. RDFS

2.1 Specify a RDFS vocabulary for the given RDF

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xo="http://example.org/ontology#"
  xmlns:x="http://example.org/resource/">
  <rdf:Description
        rdf:about="http://example.org/resource/p3123">
    <rdf:type
      rdf:resource="http://example.org/ontology#MalePatient"/>
    <xo:name>Gerard Williams</xo:name>
    <xo:age>63</xo:age>
    <xo:nextOfKin>
      <xo:Person
        rdf:about="http://example.org/resource/p1231">
        <xo:name>Annabelle Williams</xo:name>
      </xo:Person>
    </xo:nextOfKin>
    <xo:medicalStatus>in intesive care</xo:medicalStatus>
    <xo:treatedBy>
      <xo:Physician
        rdf:about="http://example.org/resource/m2443">
        <xo:name>Caroline Smith, MD</xo:name>
      </xo:Physician>
    </xo:treatedBy>
   </rdf:Description>
</rdf:RDF>
```

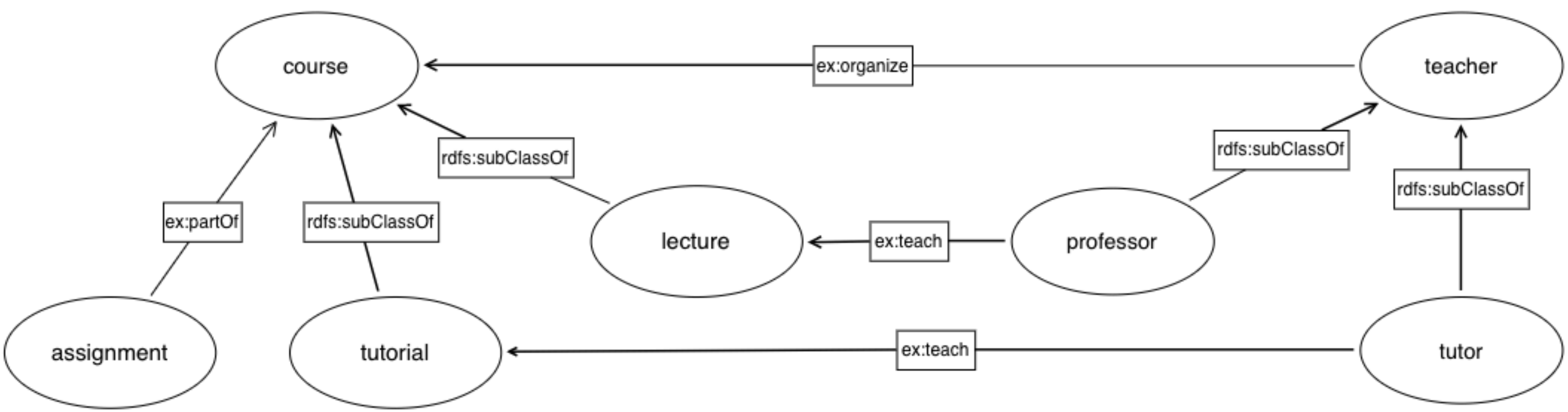
```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF[
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base="http://example.org/ontology/">

        <rdfs:Class rdf:about="Person">
        </rdfs:Class>

        <rdfs:Class rdf:about="Physician">
                <rdfs:subClassOf rdf:resource="Person"/>
        </rdfs:Class>

        <rdfs:Class rdf:about="Patient">
                <rdfs:subClassOf rdf:resource="Person"/>
        </rdfs:Class>

        <rdfs:Class rdf:about="MalePatient">
                <rdfs:subClassOf rdf:resource="Patient"/>
        </rdfs:Class>

        <rdfs:Class rdf:about="FemalePatient">
                <rdfs:subClassOf rdf:resource="Patient"/>
        </rdfs:Class>


        <rdfs:Property rdf:about="nextOfKin">
                <rdfs:domain rdf:resource="Person" />
                <rdfs:range rdf:resource="Person" />
        </rdfs:Property>

        <rdfs:Property rdf:about="age">
                <rdfs:domain rdf:resource="Person" />
                <rdfs:range rdf:resource="&xsd;integer" />
        </rdfs:Property>

        <rdfs:Property rdf:about="name">
                <rdfs:domain rdf:resource="Person" />
                <rdfs:range rdf:resource="&xsd;string" />
        </rdfs:Property>

        <rdfs:Property rdf:about="treatedBy">
                <rdfs:domain rdf:resource="Patient" />
                <rdfs:range rdf:resource="Physician" />
        </rdfs:Property>

</rdf:RDF>
```

Task 3. RDFS and OWL

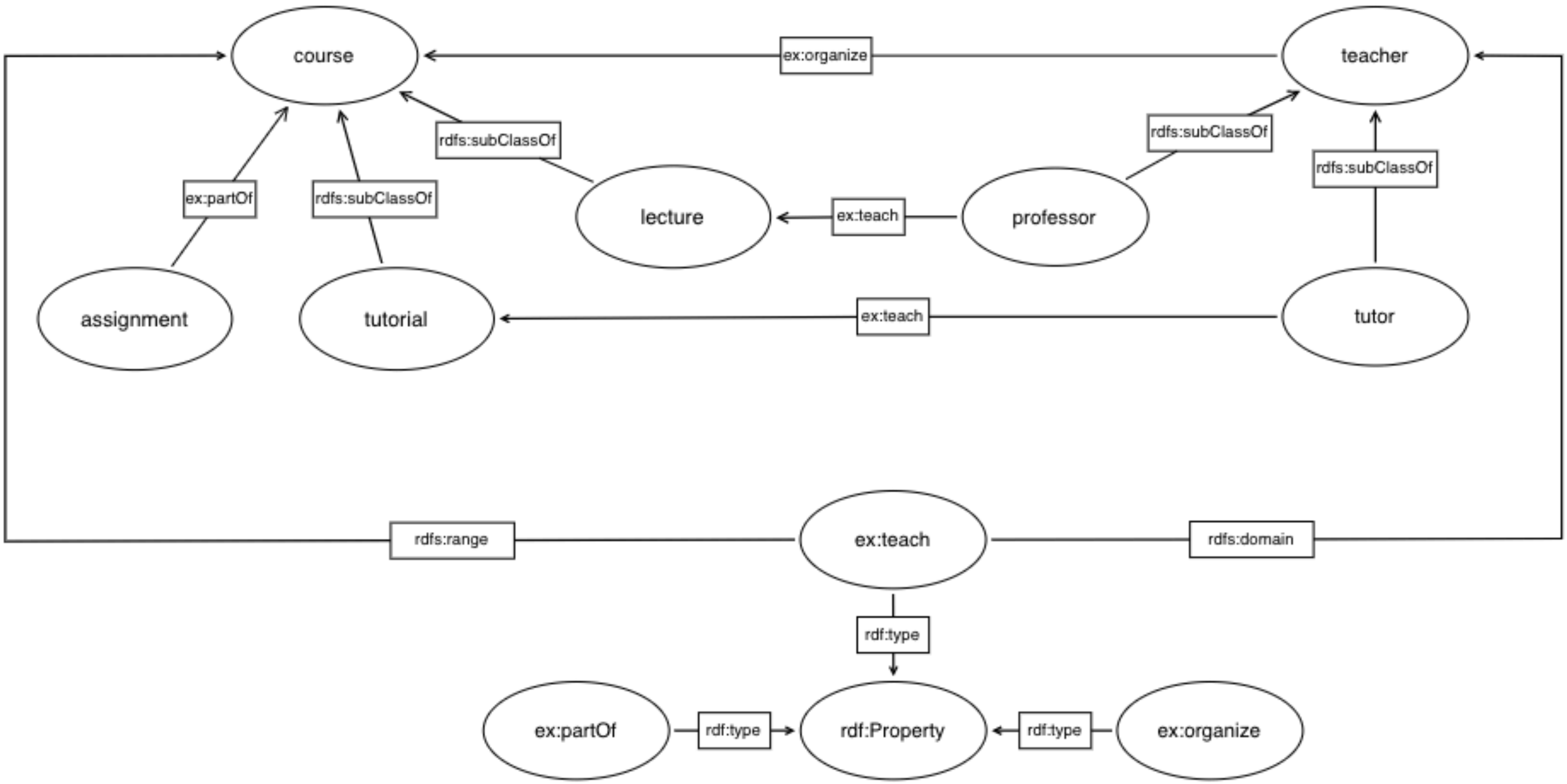
Consider the following scenario:

- In our university, courses can be lectures or tutorials.
- Assignments are part of courses.
- Courses are organized by teachers.
- Teachers can be either professors or tutors.
- Professors teach lectures, and tutors only teach tutorials.

3.1 Depict the corresponding RDFS according to the given scenario.



Task 3. RDFS and OWL



Task 3. RDFS and OWL

3.2 Write an OWL ontology that models the scenario.

```xml
<rdf:RDF
  xmlns = "http://www.example.org/"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema"
  xmlns:owl = "http://www.w3.org/2002/07/owl#" >
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="course">
  <rdfs:comment>an element of a university programm</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="lecture">
  <rdfs:comment>lecture is a type of a course</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#course"/>
</owl:Class>
<owl:Class rdf:ID="tutorial">
  <rdfs:comment>tutorial is a type of a course</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#course"/>
</owl:Class>
<owl:Class rdf:ID="assignment">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#partOf"/>
      <owl:allValuesFrom rdf:resource="#course"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="teacher">
  <rdfs:comment>teacher is a part of university staff</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="tutor">
  <rdfs:comment>tutor is are teacher that teaches tutorial</rdfs:comment>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#teacher"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teach"/>
      <owl:allValuesFrom rdf:resorce="#tutorial"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
<owl:Class rdf:ID="professor">
  <rdfs:comment>professor is a teacher that gives lecture</rdfs:comment>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#teacher"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teach"/>
      <owl:someValuesFrom rdf:resource="#lecture"/>
    </owl:Restriction>
  </owl:intersectionOf >
</owl:Class>
<owl:TransitiveProperty rdf:ID="partOf"/>
<owl:ObjectProperty rdf:ID="teach">
  <rdfs:domain rdf:resource="#teacher"/>
  <rdfs:range rdf:resource="#course"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="organize">
  <rdfs:domain rdf:resource="#teacher"/>
  <rdfs:range rdf:resource="#course"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Tutorial 6

Ontology Engineering

Create an ontology in the domain of tourism. You can take the city of Koblenz as an example. Your ontology should cover such aspects as accommodation, places of interest, gastronomy and others. Make sure you follow the steps required to design ontology.

1. Design ontology.
   - Go through all the steps of ontology engineering, following the methodology presented in the lecture.
   - Document all the steps (1-7) by outlining the respective results.

Ontology Engineering

Step 1. Determine the domain and scope of the ontology

1. The ontology covers accommodation, showplaces and gastronomy that can be of interest for tourists in Koblenz.

Scope of the domain:

Superclass: Place

Subclasses: Showplace
GastronomicLocations
Accommodations

2. Goal: support a tourist in finding a key information about respective places Koblenz.

3. Competency questions:
   What kind of accommodation/showplace/gastronomy can be found in Koblenz?
   What should be paid attention to when looking for a place?
   What is an average price tag for a chosen place (entrance fee, price per night, etc.)?
   What cuisine is being served?
   What is the level of a restaurant?
   What is the rating of a hotel?
   What are the opening hours?
   Where can I receive further information about the place (phone or website)?

4. The ontology can be *potentially* used and maintained by the team of http://www.koblenz-touristik.de/.

Ontology Engineering

Step 2. Consider reusing existing ontologies (or parts of them)

http://ontologies.sti-innsbruck.at/acco/ns.html or any other existing ontologies about hotels, restaurants and others or parts of the ontologies could be used.

Ontology Engineering

Step 3. Enumerate important terms in the ontology

Terms:

Place:	**Accommodation**	**Gastronomy**	**Showplace**
price	hotel	pub/bar	museum
name	hostel	beer garden	gallery
address	holiday apartment	bistro	culture
phone number	campsite	café	building
website	number of rooms	fast food	church
description	wi-fi	restaurants	site
	numberOfStars	wine tasting	opening hours
	check in time	type of food/drinks	
	check out time	opening hours	
	card acceptance		
	parking place		

Ontology Engineering

Step 4. Define the classes and the class hierarchy

Concept VS Property

“If the concepts with different slot values become restrictions for different slots in other classes, then we should create a new class for the distinction. Otherwise, we represent the distinction in a slot value.”

“If a distinction is important in the domain and we think of the objects with different values for the distinction as different kinds of objects, then we should create a new class for the distinction.” (Noy & McGuinness*)

* Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology

Ontology Engineering

Step 4. Define the classes and the class hierarchy

Place

Accommodation	**Gastronomy**	**Showplace**
hotel	pub	
hostel	beer garden	
holiday apartment	bistro	
campsite	café	
	restaurant	

Ontology Engineering

Step 5. Define the properties of the classes (slots)

Place:
hasPrice
hasName
hasAddress
hasPhoneNumber
hasWebsite
hasDescription

Accommodation:
hasWiFi
hasNumberOfStars
hasCheckIn
hasCheckOut
cardAccepted
hasParking

Hostel/Hotel/Holiday apartment:
hasNumberOfRooms

Campsite:
hasNumberOfPlaces

Gastronomy:
hasOpeningHours
hasTypeOfFood
hasTypeOfDrinks
hasNumberOfSittingPlaces

Restaurants:
hasNumberOfStars

Showplace:
typeOfPlace
hasOpening hours

Ontology Engineering

Step 6. Define the facets of the slots

Place {domain property(type) range}:

1. hasPrice(DataProperty) int – cardinality 1
2. hasName(ObjectProperty) Name – cardinality 1
3. hasAddress(DataProperty) string – cardinality 1
4. hasPhoneNumber(DataProperty) string – cardinality min 1
5. hasWebsite(DataProperty) anyURI – cardinality min 1
6. hasDescription(DataProperty) string

Accommodation {domain property(type) range}:

1. hasWifi(DataProperty) boolean
2. hasNumberOfStars(DataProperty) int
3. hasCheckIn(DataProperty) time
4. hasCheckOut(DataProperty) time
5. cardAccepted(DataProperty) boolean
6. hasParking(DataProperty) boolean

Hotel/Hostel/Holiday apartment:

1. hasNumberOfRooms(DataProperty) int - cardinality min 1

Campsite:

1. hasNumberOfPlaces(DataProperty) int - cardinality min 1

Ontology Engineering

Step 7. Create instances (of classes)

Place

Gastronomy

Beer garden

hasName: Best Beer
hasAddress: Koblenz, Unistr. 1
hasPhoneNumber: 1234567
hasWebsite: bestbeer.de
hasDecription: the biggest beer selection in Koblenz
hasPrice: 10
hasOpeningHours: 12:00-23:00
hasTypeOfFood: fastfood
hasTypeOfDrinks: beer
hasNumberOfSittingPlaces: 60

Ontology Engineering

Axioms

1. Showplaces and GastronomicLocations and Accommodations are disjoint classes.
2. Hotel and Hostel and HolidayApartment and Campsite are disjoint classes.
3. Pub and BeerGarden and Bistro and Café are disjoint classes.
4. BeerGardens hasTypeOfDrinks only Beer.

Questions?