Knowledge Engineering and Semantic Web

Exercise Sheet: 5

Will be discussed on: June 13,2023

Leibniz Universität Hannover

TUTORS:
Yaser Jaradeh, Hassan Hussien, and some other ORKG members

QUESTIONS: Please don't hesitate to ask any questions. Questions help you and your peers.
PRINT: Please consider the environment before printing the exercise.

1 Review Questions

1. Among the following statements about OWL, which ones are correct?

   (a) `owl:Nothing` is the subclass of all OWL classes. **Solution:**
   ✓ **By definition,** `Nothing` **is a class which includes no entity. So it's subclass of all classes.**

   (b) All object properties are functional.
   ✗ **To make an object property functional, it should be defined as a functional property. e.g** `:p rdf:type owl:FunctionalProperty.`

   (c) `:A owl:disjointWith :B` means the classes `:A` and `:B` don't have any instance in common. ✓ **By definition.**

   (d) `owl:sameAs` denotes that two classes in OWL are the identical.
   ✗ `owl:sameAs` **stands for showing two individuals are identical, not classes.**

2. Of the statements below, which one is correct about `owl:NegativePropertyAssertion`?

   (a) It's supported by all versions of OWL.
   ✗ `owl:NegativePropertyAssesrtion` **is only cover by OWL 2.0.**

   (b) It can take `Classes` and `Individuals` as its source.
   ✗ `owl:NegativeProprety Assertion` **is applied to show the negative facts, so it takes individual as its resource.**

   (c) `owl:targetIndividual` is an `owl:ObjectProperty`.
   ✓ `owl:targetIndividual` **is an** `owl:ObjectProperty`**, which indicating the target(object) of a negative fact.**

   (d) It's used to express negative facts in an OWL ontology. ✓ **By definition.**

3. Which ones of the statements below are correct?

   (a)
   ```
   :p   a   owl:ObjectProperty ;
        rdfs:range   xsd:string .
   ```
   ✗ `owl:ObjectProperty` **relates two individuals. So the range of an object property couldn't be a data type such as string.**

   (b)
   ```
   :a   :p   :c .
   :b   :p   :c .
   :p   a    owl:FunctionalPropery .
   →

   :a   owl:sameAs   :b.
   ```
   ✗ **A** `owl:FunctionalProperty`**, takes similar objects for the same subjects, not vice versa.**

   (c)
   ```
   :p   a   owl:DatatypeProperty .
   :b   :p   :c .
   →

   :c   rdfs:subClassOf   owl:Class .
   ```
   ✗ **If** `:p` **is a** `owl:DatatypeProperty`**, then** `:c` **is a literal value, which is not a resource and so it couldn't be subclass of an** `owl:Class`**.**

(d)
```
:a   :p   :c .
:b   :p   :c .
:p   a   owl:inverseFunctionalProperty .
```
$\rightarrow$
```
:a   owl:sameAs   :b .
```
✓ **By definition**

2 Given are the following OWL expressions in RDF/XML syntax.

a.
```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:someValuesFrom rdf:resource="#Physician" />
</owl:Restriction>
```

b.
```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Tosca" />
        <owl:Thing rdf:about="#Salome" />
      </owl:oneOf>
    </owl:Class>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Turandot" />
        <owl:Thing rdf:about="#Tosca" />
      </owl:oneOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

c.
```
<owl:Class rdf:about="#MusicDrama">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Opera"/>
        <owl:Class rdf:about="#Operetta"/>
        <owl:Class rdf:about="#Musical"/>
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#Opera">
  <rdfs:subClassOf rdf:resource="#MusicDrama"/>
</owl:Class>

<owl:Class rdf:about="#Operetta">
  <rdfs:subClassOf rdf:resource="#MusicDrama"/>
  <owl:disjointWith rdf:resource="#Opera"/>
</owl:Class>

<owl:Class rdf:about="#Musical">
  <rdfs:subClassOf rdf:resource="#MusicDrama"/>
  <owl:disjointWith rdf:resource="#Opera"/>
  <owl:disjointWith rdf:resource="#Operetta"/>
</owl:Class>
```

1. Explain the meaning of each expression in your own words.
2. Represent each expression in the Turtle or Manchester-syntax serialization.

Solution:

a- This expression defines a class of individuals which have at least one parent who is a physician.
Turtle representation:

```
[]   a   owl:Restriction;
     owl:onProperty   :hasParent;
     owl:someValuesFrom   :Physician.
```

b- In this expression the value of `owl:intersectionOf` is a list of two class descriptions, namely two enumerations, both describing a class with two individuals. The resulting intersection is a class with one individual, namely Tosca as this is the only individual that is common to both enumerations.
Turtle representation:

```
[]   a   owl:Class;
     owl:intersectionOf   ([a   owl:AllDifferent;
                            owl:distinctMembers(:Tosca   :Salome)]
                           [a   owl:AllDifferent;
                            owl:distinctMembers(:Turandot   :Tosca)]).
```

c- In this expression, `owl:disjointWith` statements are used together with `owl:unionOf` in order to define a set of mutually disjoint and complete subclasses of a superclass. In natural language: every MusicDrama is either an opera, an Operetta, or a Musical (the subclass partitioning is complete) and individuals belonging to one subclass, e.g., Opera, cannot belong to another subclass, e.g., Musical (disjoint or non-overlapping subclasses).
Turtle representation:

```
:MusicDrama   a   owl:Class;
     owl:equivalentClass [a   owl:Class;
                          owl:unionOf   (:Opera   :Operreta   :Musical)].
:Opera   rdfs:subClassOf   :MusicDrama.
:Operetta   rdfs:subClassOf   :MusicDrama;
            owl:disjointWith   :Opera.
:Musical   rdfs:subClassOf   :MusicDrama;
           owl:disjointWith   :Opera;
           owl:disjointWith   :Operetta;
```

3 Indicate which of the following statements are logical consequences of the knowledge base below.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix ex: <http://example.org#>.
ex:LuckyLuke   a   ex:Person;
               ex:isFriendOf   ex:JollyJumper, ex:Shrief .
ex:JollyJumper   a   ex:Horse .
ex:Rantanplan   a   ex:Dog ;
                ex:isFriendOf   ex:JollyJumper .
ex:Dog   rdfs:subClassOf   ex:Animal .
ex:Horse   rdfs:subClassOf   ex:Animal .
ex:LukePet   rdfs:subClassOf   [
  rdf:type   owl:Class ;
  owl:intersectionOf  ( ex:Animal   [
                                    owl:equivalentClass [
                                    rdf:type   owl:Restriction;
                                    owl:onProperty   ex:isFriendOf;
                                    owl:hasValue   ex:LuckyLuke]
                                        ])] .
ex:Creature   rdfs:subClassOf   [
                                 a   owl:Class;
                                 owl:unionOf   (ex:Animal   ex:Person)].
ex:LuckyLuke   ex:isEnemyOf   ex:JoeDalton .
ex:isEnemyOf   a   owl:SymmetricProperty;
               rdfs:subPropertyOf   ex:knows ;
               rdfs:domain   ex:Person .
```

Statements:

a. `ex:Rantanplan  a  ex:creature .`
✗ `ex:Rantanplan` **is a** `ex:Dog` **and so it is an** `ex:Animal`**. And we know that** `ex:Creature` **is subclass of a union set of** `ex:Animal` **and** `ex:Person`**, but it doesn't guarantee that it covers all animals or all persons (because it's not the union of them, but subclass of these two classes.)**

b. `ex:Rantanplan  ex:isFriendOf  ex:Shrief .`
✗ **We just know that** `ex:Rantanplan` **is friend of** `ex:JollyJumper`**, but have no more information, so we cannot infer that it is friend of** `ex:Shrief`

c. `ex:Shrief  a  ex:LukePet .`
✗ **By definition of** `ex:LukePet` **we know it's a class, which is subclass of intersection of animal and another class which including individuals which are friend of** `ex:LuckyLuke`**(satisfying the restriction** `ex:isFriendOf`**). But,** `ex:Shrief` **is just friend of** `ex:LuckyLuke` **and it's not mentioned that it's** `ex:Animal` **too.**

d. `ex:Rantanplan  a  ex:LukePet .`
✗ `ex:Rantanplan` **is animal, but there is no chain of triples which shows it's a friend of** `ex:LuckyLuke`**.**

e. `ex:JoeDalton  ex:knows  ex:LuckyLuke .`
✓ **We know** `ex:LuckLuke` **is enemy of** `ex:JoeDalton` **by triple:**
`ex:LuckyLuke  ex:isEnemyOf  ex:JoeDalton .`
And `ex:isEnemyOf` **is a symmetric property. So we infer that:**
`ex:JoeDalton  ex:isEnemyOf  ex:LuckyLuke.`
And as `ex:isEnemyOf` **is a subproperty of** `ex:knows`**, then we can infer that:**
`ex:JoeDalton  ex:knows  ex:LuckyLuke.`

f. `ex:JoeDalton  a  ex:creature .`
. ✗ **Regarding the triples:**
`ex:LuckyLuke  ex:isEnemyOf  ex:JoeDalton .`

And `ex:isEnemyOf a owl:SymmetricProperty.`
We can infer `ex:JoeDalton ex:isEnemyOf ex:LuckyLuke.`
And from: `ex:isEnemyOf rdfs:domain ex:Person.`
We infer that `ex:JoeDalton a ex:Person.`
But, it's not still enough to call `ex:JoeDalton a ex:Person.`
Because similar to explanation of part a, this class doesn't necessarily cover all persons or animals as it's just subclass of `ex:Animal` ∪ `ex:Person`

g. Talk about the statement d, in the case we add these triples to our knowledge base.

```
ex:isFriendOf a owl:SymmetricProperty, owl:transitiveProperty.
```

✗ **In this case,** `ex:Rantanplan` **would have both conditions of intersection statement in definition of** `ex:LukePet`**. But, take it into consideration that** `ex:LukePet` **is a subclass of this intersection statement, so it may not cover** `ex:Rantanplan`**.**

4 Modeling in OWL:

Given are some facts about the DSDL research group and the "Knowledge Engineering and Semantic Web" lecture. Model them in an appropriate way as an OWL ontology.

1. "KESW" and "Knowledge Engineering and Semantic Web" are two names for the same lecture.
2. KESW *lecture* is different from KESW *seminar*.
3. If a mentor is the supervisor of a student, then that student is supervised by that mentor.
4. DSDL has some PhD or Master students. (In your model, take it into consideration that one student cannot be PhD and Master student in the same time.)
5. All tutors of KESW are students and enrolled at LUH.
6. A student eligible to register their master thesis should have achieved at least 1 and at most 2 seminars.
7. Professor Jens Lehmann is not a lecturer of KESW.
8. The DSDL group offers two *different* lectures: KESW and AIKG.
9. Tutors of KESW are Yaser, Hassan, Julia, Hamed, and Mahsa.
10. Students who failed KESW are students who have enrolled in KESW and haven't passed it.

Solution:

1. `ex:KESW owl:sameAs ex:KnowledgeEngineeringSemanticWeb.`

2. `ex:KESWLecture owl:differenFrom ex:KESWSeminar.`

3. `ex:superviserOf owl:inverseOf ex:supervisedBy.`

4.
```
ex:DSDL rdfs:subClassOf [rdf:type owl:Restriction ;
                        owl:onProperty ex:hasMember;
                        owl:someValuesFrom [
                                rdf:type owl:Class;
                                owl: unionOf (ex:MasterStudent ex:PhDStudent)]].
ex:MasterStudent owl:disjointWith ex:PhDStudents.
```

5.
```
ex:Tutor   rdfs:subClassOf [a   owl:Restriction;
                            owl:onProperty   ex:hasMember;
                            owl:allValuesFrom   [
                                      a   owl:Class;
                                      owl:intersectionOf (ex:Student [
                                                a   owl:Restriction;
                                                owl:onProperty   ex:enroledAt;
                                                owl:hasValue   ex:LUH])].
```

6.
```
ex:EligableStudent   a   owl:Class;
                     rdfs:subClassOf [
                               a   owl:Restriction;
                               owl:onProperty   ex:achievedSeminar;
                               owl:minCardinality   "1"^^xsd:nonNegativeInteger].
  ex:EligableStudent   a   owl:Class;
                     rdfs:subClassOf [
                              a   owl:Restriction;
                              owl:onProperty   ex:achievedSeminar;
                              owl:maxCardinality   "2"^^xsd:nonNegativeInteger].
```

7.
```
[]   a   owl:NegativePropertyAssertion;
     owl:sourceIndividual    ex:JensLehmann;
     owl:assertionProperty   ex:isLecturerOf;
     owl:targetIndividual    ex:KESW.
```

8.
```
ex:DSDL   ex:offersLectures [a   owl:AllDifferent;
                            owl:DistinctMembers(ex:KESW   ex:AIKG)].
```

9.
```
ex:KESWTutors   a   owl:Class;
                owl:oneOf(ex:Yaser   ex:Hassan   ex:Julia   ex:Hamed   ex:Mahsa).
```

10.
```
ex:FailedKESW   a   owl:Class;
                rdfs:subClass [a   owl:Restriction;
                               owl:onPropety   ex:enrolledIn;
                               owl:hasValue   ex:KESW];
                owl:complementOf   ex:PassedKESW.
```