

INFO-H509: XML & Web Technologies

Semantic Web Exercises

SOLUTIONS

Lecturer: Stijn Vansummeren

Assistant: Michael Waumans

2014–2015

1 RDF

Exercise 1.1

1. See the included files for the graph drawings.
2. See the included files for the graph drawings.
3. See the included files for the graph drawings.
4. The triples, written in the turtle syntax for reasons of readability, are the following:

```
@prefix ulb: <http://code.ulb.ac.be/example/terms/> .
@prefix infoh509: <http://code.ulb.ac.be/example/courses/infoh509/> .
@prefix catalogue: <http://code.ulb.ac.be/example/catalogue/> .
@prefix staff: <http://code.ulb.ac.be/example/staff#> .
@prefix infoh509media: <http://cs.ulb.ac.be/public/_media/teaching/infoh509/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ev: <http://purl.org/rss/1.0/modules/event/>

infoh509:lecture2          ev:location          "UA4.218" .
infoh509:lecture2          ev:startDate          "2009-02-12T10:00:00"^^xs:dateTime .
infoh509:lecture2          ev:endDate            "2009-02-12T12:00:00"^^xs:dateTime .
infoh509:lecture2          rdf:type              ulb:Lecture .
infoh509:lecture2          ulb:teaching_material infoh509media:2-xpath.pdf .
infoh509media:2-xpath.pdf  dc:title              "Cours 2: XPath" .
infoh509media:2-xpath.pdf  dc:creator            staff:svansumm .
```

Exercise 1.2

The file `catalog.rdf` is modified as follows. The course INFO-H-200 is added:

```
<rdf:Description rdf:about="infoh200">
  <dc:title>Object Oriented Programming</dc:title>
  <ulb:lecturer rdf:resource="&staff;ezimanyi" />
  <ulb:assistant rdf:resource="&staff;boverhae" />
  <ulb:prerequisite rdf:resoucre="&ulb;infoh100"/>
</rdf:Description>
```

And the prerequisite is added to the course INFO-H-415:

```
<rdf:Description rdf:about="infoh415">
  ...
  <ulb:prerequisite rdf:resoucre="&ulb;infoh303"/>
</rdf:Description>
```

Exercise 1.3

```
@prefix myconcepts: <http://www.example.org/johndoe/> .
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .
@prefix catalogue: <http://code.ulb.ac.be/example/catalogue/> .

myconcepts:me      a                foaf:Person .
myconcepts:me      foaf:name        "John Doe" .
myconcepts:me      foaf:mbox        "jd@gmail.com" .
myconcepts:me      myconcepts:has_course catalogue:infoh509 .
```

2 RDF Schema

Exercise 2.1

The following needs to be added to the file `inference.ttl` to state that all personnel members (`ulb:Faculty`) are people (`foaf:Person`).

```
ulb:Faculty      rdfs:subClassOf    foaf:Person .
```

Exercise 2.2

```
ulb:prerequisite  rdfs:range        ulb:Course .
ulb:prerequisite  rdfs:domain       ulb:Course .
```

Supplementary exercise: In a similar vain, still using `rdfs:range` and `rdfs:domain`, add rules that describe the properties `lecturer` and `assistant` in more detail.

```
ulb:lecturer     rdfs:range        ulb:Faculty .
ulb:lecturer     rdfs:domain       ulb:Course .
ulb:assistant     rdfs:range        ulb:Faculty .
ulb:assistant     rdfs:domain       ulb:Course .
```

Exercise 2.3

```
ulb:workHomepage  rdfs:subPropertyOf foaf:homepage .
```

Note that this property is of type `rdf:Property`, but also of type `rdfs:Resource`. Recall that RDFS comes equipped with a number of standard “axioms” that state, among others, that the domain and range of `subPropertyOf` are `rdf:Property` and that `rdf:Property` is a subclass of `rdfs:Resource`.

In the file `staff.rdf`, the homepage properties can then be replaced by `workHomePage` as follows.

```
<ulb:Professor rdf:ID="svansumm">
  ...
  <ulb:workHomepage
```

```

        rdf:resource="http://code.ulb.ac.be/code.people.php?id=992"/>
</ulb:Professor>
<ulb:Professor rdf:ID="ezimanyi">
    ...
    <ulb:workHomepage rdf:resource="http://cs.ulb.ac.be/members/esteban/" />
</ulb:Professor>

```

3 OWL

Exercise 3.1

Recall that OWL DL defines the following property characteristics.

1. `owl:TransitiveProperty`
2. `owl:SymmetricProperty`
3. `owl:FunctionalProperty`
4. `owl:InverseFunctionalProperty`

For each of the following properties, list which of these property characteristics could apply.

1. The prerequisite of a course (`ulb:prerequisite`)
2. The student number of a student
3. Birthdate
4. `owl:sameAs`
5. `owl:inverseOf`

Supplementary exercise: Complete the description of `ulb:prerequisite` in the file `inference.ttl`. Verify the effect using the `inferencetool.jar` utility.

	Transitive	Symmetric	Functional	InverseFunctional
Prerequisite of a course	O			
Student number			O	O
Birthdate			O	
<code>owl:sameAs</code>	O	O		
<code>owl:inverseOf</code>		O		

Transitive If we need to have followed course c_1 in order to be able to follow course c_2 and similarly, if we need to follow course c_2 in order to be able to follow course c_3 , then it stands to reason that we need to follow c_1 before we can follow c_3 . Analogously, if a is identical to b and b is identical to c , then a is identical to c . `prerequisite` and `sameAs` are hence transitive.

Symmetric If a is identical to b , then clearly b is identical to a . Similarly, if a is the inverse of b , then b is the inverse of a . `sameAs` et `inverseOf` are hence symmetric.

Functional A person has exactly one birthdate. Moreover, a person has at most one student id number. These properties are hence functional.

Functionally inverse A student id number is never attributed to more than one person. As such, it is functionally inverse.

Note Note, however, that, since this property links an individual to a literal data item it is a `owl:DataObjectProperty`. Therefore, if we specify that it is functionally inverse, we get an ontology that is in OWL Full, and not OWL DL. It is hence possible that we will not be able to automatically reason with it.

Supplementary exercise:

The property `prerequisite` is defined to be transitive as follows:

```
ulb:prerequisite    a                owl:TransitiveProperty .
```

Exercise 3.2

```
ulb:teaches    owl:inverseOf    ulb:lecturer .
ulb:teaches    rdfs:domain        ulb:Faculty .
ulb:teaches    rdfs:range        ulb:Course .
```

Exercise 3.3

```
staff:fpicalau    owl:sameAs    <http://my.opera.com/fpicalausa/xml/foaf#me> .
```

Exercise 3.4

```
@prefix terms: <http://code.ulb.ac.be/example/terms/> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
```

```
terms:PizzaTopping a owl:Class .
```

```
terms:Vegetable a          owl:Class ;
rdfs:subClassOf terms:PizzaTopping .
```

```
terms:Pizza    a owl:Class ;
owl:disjointWith terms:PizzaTopping .
```

```
terms:aubergine a    terms:Vegetable .
```

```
terms:hasTopping a    owl:ObjectProperty ;
  rdfs:domain terms:Pizza ;
  rdfs:range  terms:PizzaTopping .
```

```
terms:noMeatPizza a    owl:Class ;
  rdfs:subClassOf terms:Pizza .
```

```
terms:noFishPizza a    owl:Class ;
  rdfs:subClassOf terms:Pizza .
```

```
terms:VegPizza    a    owl:Class ;
  owl:intersectionOf (terms:noMeatPizza terms:noFishPizza) .
```

```
terms:hasTopping rdfs:subPropertyOf terms:hasIngredient .
```

```
terms:boring a terms:noMeatPizza .
terms:boring a terms:noFishPizza .
```

- Add an individual to both the class `PizzaTopping` and `Pizza`. Run the inference tool. What do you get? What should you do to remedy this?

Answer Since we have declare `PizzaTopping` and `Pizza` to be disjoint, the document has become inconsistent (i.e., it does not have any model). We should remove the individual from one of the two classes to regain consistency.

- Add an individual to both the class `NoMeatPizza` and `NoFishPizza`. What do you expect to get?

Answer: Since we have declared the class `VegPizza` to consist of those elements which are in the class `NoMeatPizza` and in the class `NoFishPizza`, the tool correctly infers that the individual belongs to the class `VegPizza`.

Exercise 3.5

Continuing Exercise 3.4, use OWL DL to model the following sentences.

1. Every pizza has tomato as a topping.

```
terms:Pizza rdfs:subClassOf [
  a owl:Restriction;
  owl:onProperty terms:hasTopping;
  owl:hasValue terms:tomato
] .
```

2. Every pizza in the class `PizzaMargarita` has exactly tomato and cheese as topping. We actually need two declarations:

```
terms:PizzaMargarita rdfs:subClassOf [
  a owl:Restriction;
  owl:onProperty terms:hasTopping;
  owl:allValuesFrom [
    a owl:Class;
    owl:oneOf (terms:tomato terms:cheese)
  ]
] .
```

```
terms:PizzaMargarita rdfs:subClassOf [
  a owl:Class ;
  owl:intersectionOf
    ( [a owl:Restriction; owl:onProperty terms:hasTopping; owl:hasValue terms:tomato]
      [a owl:Restriction; owl:onProperty terms:hasTopping; owl:hasValue terms:cheese] )
] .
```

The first specifies that all values in the range of `hasTopping` have to be either `terms:tomato` or `terms:cheese`. The second specifies that `hasTopping` should exist with these values.

Note that the `inertools.jar` file does not reason with `owl:OneOf`. As such, the tool will not be able to conclude from the following triples that the ontology has become inconsistent (because `hasTopping` has a value that is distinct from tomato or cheese). There are more powerfull tools out there (like, `pellet` <http://clarkparsia.com/pellet/download/pellet-2.3.1> that allow to draw such conclusions).

```

terms:pm a terms:PizzaMargarita .
terms:pm terms:hasTopping terms:ex1 .
terms:ex1 owl:differentFrom terms:tomato, terms:cheese .

```

4 SPARQL

Exercise 4.1

Write SPARQL queries for the following queries. (All queries should be run with files `catalog.rdf`, `staff.rdf`, and `infoh509.ttl`.)

1. Retrieve the URIs of all the courses.

```

PREFIX ulb: <http://code.ulb.ac.be/example/terms/>
SELECT ?course
WHERE {
    ?course      a      ulb:Course .
}

```

2. Retrieve, for each course, its title and the name of the lecturer.

```

PREFIX ulb: <http://code.ulb.ac.be/example/terms/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?title ?name
WHERE {
    ?course      a      ulb:Course .
    ?course      dc:title      ?title .
    ?course      ulb:lecturer  ?lecturer .
    ?lecturer    foaf:name     ?name .
}

```

3. The name of all Professors who teach a course such that (s)he also teaches a prerequisite for that course.

```

PREFIX ulb: <http://code.ulb.ac.be/example/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?course      a      ulb:Course .
    ?course      ulb:lecturer  ?lecturer .
    ?course      ulb:prerequisite ?pre .
    ?pre         ulb:lecturer  ?lecturer .
    ?lecturer    foaf:name     ?name .
}

```

4. All persons (`foaf:Person`) and their personal homepage, should this be available (i.e., retrieve just the person if no homepage is available).

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?homepage

```

```

WHERE {
  ?person      a                foaf:Person      .
  ?person      foaf:name        ?name           .
  OPTIONAL {
    ?person      foaf:homepage    ?homepage
  }
}

```

5. The title of all courses that have been organized in *UA4.218*.

```

PREFIX ulb:    <http://code.ulb.ac.be/example/terms/>
PREFIX dc:     <http://purl.org/dc/elements/1.1/>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX ev:     <http://purl.org/rss/1.0/modules/event/>
SELECT DISTINCT ?title
WHERE {
  ?course      a                ulb:Course      .
  ?course      dc:title         ?title          .
  ?course      ulb:schedule     ?schedule       .
  ?schedule    ?test            ?item           .
  ?item        ev:location      "UA4.218"       .
}

```

6. **Supplementary exercise:** All persons who know someone who know M.Vansummeren.

```

PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX staff:  <http://code.ulb.ac.be/example/staff#>
SELECT DISTINCT ?name
WHERE {
  ?p2          foaf:knows       ?p1            .
  staff:svansumm foaf:knows     ?p1            .
  ?p2          foaf:name        ?name          .
  FILTER(!sameTerm(?p2, staff:svansumm))
}

```

The filter `!sameTerm` expresses that the person `p2` (who knows someone who know prof. Vansummeren) isn't prof. Vansummeren himself.