INF3580, Semantic Technologies
Mandatory Exercise (oblig) 2

Published:	30.04.2010
Due:	14.05.2010
Corrected until:	20.05.2010
Second attempt due:	27.05.2010

Formalities

Note: the dates given above are firm deadlines to ensure that all results are registered in due time before the exam. Any exceptions have to be negotiated with the student administration.

All questions are to be solved individually. On demand, you have to be able to explain your answers.

Put all files constituting your answers into one directory that has the same name as your UiO login ID. Make a ZIP or TAR archive of this directory. Please make sure that unpacking the archive produces a directory that has the same name as your ID, and put all files into that directory without further sub-directories. Send the archive by mail to `martingi@ifi.uio.no`, with a subject of "INF3580 Oblig 2".

On the semester web page, there is a link to "Comments on Oblig 1". Please take the advice given there into account when preparing the answer to this assignment!

Question 1

Create an OWL ontology that describes the way places and distances were modeled in the first mandatory exercise. The ontology should use the same namespaces:

```
@prefix places: <http://inf3580.ifi.uio.no/places#>
@prefix geo:    <http://www.w3.org/2003/01/geo/wgs84_pos#>
```

Hint: In Protégé, it is easiest to set the "Ontology URI" to a default namespace `http://inf3580.ifi.uio.no/places#` when creating the ontology. For the entities in the `geo:` namespace, create them in `places:`

first, then use "Change entity URI" or "Change multiple entity URIs" in the "Refactor" menu to change their URIs.

The ontology should express at least the following:

- Every `places:Place` has
  - at least one `places:name`
  - exactly one `places:position`, which is a `geo:Point`.
- Every `geo:Point` has
  - exactly one `geo:lat` which is a `xsd:double`
  - exactly one `geo:long` which is a `xsd:double`
- Every `places:Connection` has
  - exactly two `places:endpoints`
  - exactly one `places:distance` which is a `xsd:double`

Deliverable: A file `places.owl` containing the ontology in OWL/RDF format.

Question 2

Given a set of triples

```
:x a places:Connection ;
   places:endpoint :a ;
   places:endpoint :b .
```

and an ontology like the one from Question 1 that states that all `places:Connection` objects are connected to exactly two `places:Place` objects by the property `places:endpoint`.

(a) Does it follow from the ontology and the given triples that the interpretations of `:a` and `:b` are different? If no, give a counter-model.

(b) Does it follow from the ontology and the given triples that the interpretations of `:a` and `:b` are the same? If no, give a counter-model.

(c) Which triples would you have to add to ensure that the interpretations of `:a` and `:b` *are* different, resp. the same?

Deliverable: A file `question2.txt` with answers to the three questions.

Question 3

We will now consider places with information about their number of inhabitants.

Extend the ontology from Question 1 as follows:

- add a functional datatype property `places:population` with domain `places:Place` and range `xsd:integer`.
- add a class `places:Town` that is defined to contain all places with a population of at least 10 000.
- add a class `places:Metropolis` that is defined to contain all places with a population of at least 1 000 000.

Design a SPARQL query that finds all towns y that lie within 10 km of a metropolis x. The query should use the classes defined in the ontology, and *not* the `places:population` property.

Test your query on the example data in `popPlaces.ttl`, which you can download from the semester web page, using an extension of the query execution program you wrote in the weekly exercises:

- It needs to create an inference model with a Pellet reasoner attached to it before executing the query.
- In addition to the RDF file with places, it needs to load the ontology `popPlaces.owl` into the model.

Deliverable: The following three files:

- The extended ontology in a file `popPlaces.owl`
- The SPARQL query in a file `suburbs.rq`
- The source code of the extended query execution program