IN3060/4060 – MANDATORY EXERCISE no. 4

Published date: 27.02.2025.

Due date: 14.03.2025 23:59.

Delivery files: 4: `family.ttl`, `oblig4.pdf`, `Oblig4.java` or `Oblig4.py`, `oblig4.rq`.

Delivery attempts: 1.

1 RDFS modelling

We have already made an RDF representation of some of the members of the Simpson family. In this exercise we will use RDFS to model the classes and properties that we used to describe the Simpson family plus some additional classes and relationships.

1.1 Exercise

Create a new RDF file, called `family.ttl`. Add all work in this section's exercises to this file. Let `fam` be the prefix for the namespace

`http://www.ifi.uio.no/IN3060/family#`

1.2 Exercise

Declare the classes and properties that we used in the first mandatory exercise, i.e., write triples that state the `rdf:type` of the following resources. The `rdfs:Class`

- `fam:Family`,

and the `rdf:Property`-es

- `fam:hasFamilyMember`,
- `fam:hasBrother`,
- `fam:hasSister`,

- `fam:hasParent`,
- `fam:hasMother`,
- `fam:hasFather`,
- `fam:hasSpouse`.

1.3 Exercise

Add more classes and properties. Classes:

- `foaf:Person`
- `fam:Man`
- `fam:Gender`
- `fam:Woman`

Properties:

- `fam:hasGender`
- `fam:hasSibling`

1.4 Exercise

State that `fam:Female` and `fam:Male` are instances of `fam:Gender`.

1.5 Exercise

Add a property `fam:isRelativeOf` which has `foaf:Person` as both domain and range. This is intended as a general property which holds between two persons related by any family relationship.

1.6 Exercise

Set the domain and range of `fam:hasBrother` to respectively `foaf:Person` and `fam:Man`.

1.7 Exercise

Continuing in a similar manner as in the previous exercise, add all correct `rdfs:subClassOf` property assertions between classes in your RDFS vocabulary. For each property add the correct `rdfs:subPropertyOf` property assertions and the correct `rdfs:domain` and `rdfs:range` assertions.

"Correct" in the sense that we assume the "obvious" interpretation, e.g., that the class `fam:Man` is the class of all men and `fam:hasBrother` is the relationship from a person to its brother.

2 Entailment

In this exercise you shall use the RDFS entailment rules to derive new facts from the RDFS statements you have created in the RDFS modelling exercise above and the statements in the RDF file given below. The RDFS entailment rules are found at `http://www.w3.org/TR/rdf11-mt/#patterns-of-rdfs-entailment-informative`, e.g.,

	If S contains:	**then S RDFS entails recognizing D:**
rdfs2	`aaa rdfs:domain xxx .` `yyy aaa zzz .`	`yyy rdf:type xxx .`

specifies the RDFS inference rule `rdfs2`, with two premisses `aaa rdfs:domain xxx .` and `yyy aaa zzz .` and the conclusion `yyy rdf:type xxx`.

RDF graph:

```
1  @prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2  @prefix sim:     <http://www.ifi.uio.no/IN3060/simpsons#> .
3  @prefix fam:     <http://www.ifi.uio.no/IN3060/family#> .
4  @prefix foaf:    <http://xmlns.com/foaf/0.1/> .
5
6  sim:Homer
7        rdf:type   foaf:Person ;
8        fam:hasSpouse sim:Marge .
9
10 sim:Marge
11       fam:hasSpouse sim:Homer .
12
13 sim:Lisa
14       rdf:type   foaf:Person ;
15       fam:hasFather sim:Homer ;
16       fam:hasParent _:1, _:2 .
17 _:1
18       rdf:type   foaf:Person ;
19       fam:hasSister sim:Patty , sim:Selma .
20 _:2
21       rdf:type   foaf:Person ;
22       fam:hasBrother sim:Herb .
```

2.1 Exercise

For each of the statements below, decide whether it follows from the RDFS statements from the previous exercise and the RDF file above. If yes, then show the derivation by indicating the rdfs rules used and on which premisses (see the tip below). If no, then, with a sentence or two, explain why.

Deliver this exercise in a pdf file named `oblig4.pdf`.

1. `sim:Marge rdf:type foaf:Person`
2. `fam:hasSister rdfs:subPropertyOf fam:isRelativeOf`
3. `sim:Marge rdf:type fam:Woman`
4. `sim:Herb rdf:type fam:Man`
5. `sim:Lisa fam:isRelativeOf sim:Homer`
6. `sim:Lisa fam:hasMother sim:Marge`
7. `sim:Patty rdf:type foaf:Person`
8. Both of these triples *together*, i.e. the blank node `_:b` is the same in both triples:
   ```
   _:a fam:hasParent _:b .
   _:b fam:hasSister sim:Patty .
   ```
9. Both of these triples *together*, i.e. the blank node `_:d` is the same in both triples:
   ```
   _:d fam:hasBrother _:e .
   _:d fam:hasBrother _:f .
   ```

2.1.1 Tip

The answer to the first statement should look like this:

> Yes, and this is the derivation:
>
> 1. `sim:Marge fam:hasSpouse sim:Homer` — P
> 2. `fam:hasSpouse rdfs:domain foaf:Person` — P
> 3. `sim:Marge rdf:type foaf:Person` — rdfs*n*, 1, 2

P indicates that the statement is a premiss, thus an already existing fact; the first premiss is found in the RDF listing in this section, the second you should have created in the exercise in the first section. "rdfs*n*, 1,2" means that the statement on this line is the result of applying the rule rdfs*n* (where *n* is a number that you have to find out) with the statements on lines 1 and 2 as premisses.

3 Reasoning

In this exercise you shall write a java or python program and a SPARQL CONSTRUCT query. The SPARQL query is similar to a query you made for the previous mandatory exercise set: it shall produce a FOAF file for Homer Simpson where he `foaf:knows` all `foaf:Person`-s he has a family relation to (`fam:isRelativeOf`).

The program shall read three arguments:

- the first argument shall be the path to the file your have written in the first exercise, RDFS modelling.
- the second argument shall be the path to your SPARQL construct query.
- the third argument shall be the file name to where the results of the SPARQL query shall be written.

The program shall:

- read the file given by the first argument,
- read the file at `https://www.uio.no/studier/emner/matnat/ifi/IN3060/v23/obliger/simpsons.ttl`
- apply RDFS reasoning to the combined model of these two files,
- and finally, execute the SPARQL CONSTRUCT query located in the second argument on this model, and output the results of the query to the file given in the third argument.

3.1 Jena

To apply RDFS reasoning in Jena you simply create a RDFS model. Use the method `ModelFactory.createRDFSModel(Model schema, Model model)`. The first argument should be a model with the RDFS schema data from the first exercise. The second argument should be a model containing instance data assertions read from the URL above.

3.2 rdflib

To apply RDFS reasoning in rdflib you need to use a new module called `owlrl`. Make sure this module is installed in your environment with `pip install owlrl`. In your file you need to import `DeductiveClosure` and `RDFS_Semantics` from the `owlrl` module with

```
from owlrl import DeductiveClosure, RDFS_Semantics
```

Then you can extend the graph using RDFS reasoning with

```
DeductiveClosure(RDFS_Semantics).expand(g)
```

Put your program in a file named `Oblig4.java` or `Oblig4.py` and your SPARQL query in a file called `oblig4.rq`.

4 Delivery, Devilry

Mandatory exercises are to be handed in using Devilry. Make sure that your are registered in the system by logging on and finding that an `oblig4` is available as an assignment in IN3060 or IN4060. *Check this before you start solving the exercises!* If you are not registered in the system, give notice to `martingi@ifi.uio.no`.

Good Luck!