IN3067/INM713 Semantic Web Technologies and Knowledge Graphs

Laboratory 6: Reasoning with RDFS Semantics and OWL 2 RL

Ernesto Jiménez-Ruiz

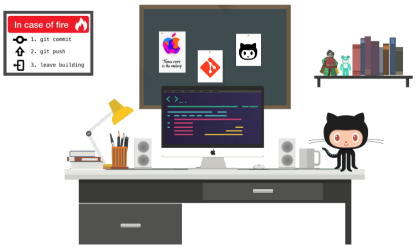
Academic course: 2023-2024
Updated: March 18, 2024

Contents

1 Git Repositories and dependencies

Support codes for the laboratory sessions are available in *GitHub*. There are two repositories, one in Python and another in Java:

`https://github.com/city-knowledge-graphs`



For Java developers we use maven to deal with dependencies (see pom file). For Python developers there is always a `requirements.txt` within each lab folder with possibly new dependencies. It is recommended to use environments, but it is not strictly necessary.

2 Inference rules (cheatsheet)

As seen in the lecture, Figure 1 summarizes the necessary inference rules we need for the lab.

Rule	If	Then add
rdf1	(x p y)	(p rdf:type rdf:Property)
rdfs2	(p rdfs:domain c) (x p y)	(x rdf:type c)
rdfs3	(p rdfs:range c) (x p y)	(y rdf:type c)
rdfs4a	(x p y)	(x rdf:type rdfs:Resource)
rdfs4b	(x p y)	(y rdf:type rdfs:Resource)
rdfs5	(p rdfs:subPropertyOf q) (q rdfs:subPropertyOf r)	(p rdfs:subPropertyOf r)
rdfs6	(p rdf:type rdf:Property)	(P rdfs:subPropertyOf p)
rdfs7	(p rdfs:subPropertyOf q) (x p y)	(x q y)
rdfs8	(c rdf:type rdfs:Class)	(c rdfs:subClassOf rdfs:Resource)
rdfs9	(c rdfs:subClassOf d) (x rdf:type c)	(x rdf:type d)
rdfs10	(c rdf:type rdfs:Class)	(c rdfs:subClassOf c)
rdfs11	(c rdfs:subClassOf d) (d rdfs:subClassOf e)	(c rdfs:subClassOf e)
rdfs12	(p rdf:type rdfs:ContainerMembershipProperty)	(p rdfs:subPropertyOf rdfs:Member)
rdfs13	(x rdf:type rdfs:Datatype)	(x rdfs:subClassOf rdfs:Literal)
	(p owl:inverseOf q)	(q owl:inverseOf p)
	(p owl:inverseOf q) (x p y)	(y q x)
	(p rdf:type owl:SymmetricProperty) (x p y)	(y p x)

schema only

data + schema

not relevant

Figure 1: Figure adapted from "Towards Efficient Schema-Enhanced Pattern Matching over RDF Data Streams". International Semantic Web Conference (ISWC) 2011. Slides.

3 RDFS inference

Consider the following set of triples (we will refer to them as the graph $\mathcal{G}_{\text{rdfs}}$).

```
@PREFIX : <http://city.ac.uk/kg/lab4/>
@PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
:Person     a                   rdfs:Class .
:Man        a                   rdfs:Class ;
            rdfs:subClassOf     :Person .
:Woman      a                   rdfs:Class ;
            rdfs:subClassOf     :Person .
:Parent     a                   rdfs:Class ;
            rdfs:subClassOf     :Person .
:Father     a                   rdfs:Class ;
            rdfs:subClassOf     :Parent ;
            rdfs:subClassOf     :Man .
:Mother     a                   rdfs:Class;
            rdfs:subClassOf     :Parent ;
            rdfs:subClassOf     :Woman .
:Child      a                   rdfs:Class ;
            rdfs:subClassOf     :Person .
:hasParent  a                   rdf:Property ;
            rdfs:domain         :Person ;
            rdfs:range          :Parent .
:hasFather  a                   rdf:Property ;
            rdfs:subPropertyOf :hasParent ;
            rdfs:range          :Father .
:hasMother  a                   rdf:Property ;
            rdfs:subPropertyOf :hasParent ;
            rdfs:range          :Mother .
:isChildOf  a                   rdf:Property ;
            rdfs:domain         :Child ;
            rdfs:range          :Parent .
:Ann        a                   :Person ;
             :hasFather         :Carl ;
             :hasMother         :Juliet .
```

3.1 Manual inference

Task 6.1 Decide if $\mathcal{G}_{\text{rdfs}}$ derives the following statement(s) and explain *why* or *why not*. In the positive case, then indicate the RDFS inference rules from the lecture (also found at `http://www.w3.org/TR/rdf-mt/`) to prove your answer. If the statement is not derived, then explain, informally or formally, why this is so. Formally can be done via a counterexample, *i.e.*, with an interpretation that entails $\mathcal{G}_{\text{rdfs}}$, but it does not the statement.

Statement 1 `:Father rdfs:subClassOf :Person .`

Statement 2 `:Woman rdfs:subClassOf :Person .`

Statement 3 `:Juliet a :Person .`

Statement 4 `:Ann a :Child .`

Statement 5 `:Ann :isChildOf :Carl .`

Statement 6 `:Ann :hasParent :Juliet .`

Statement 7 `rdfs:range rdf:type rdfs:Resource .`

Statement 8 `:Mother rdfs:subClassOf :Person .`

3.1.1 Example solution

Example solution for Statement 1:

```
:Father rdfs:subClassOf :Person .
```

True, the statements is derived by $\mathcal{G}_{\text{rdfs}}$. `:Father` is (transitively) a subclass of `:Person`. Rule **rdfs11**. Statements 1 and 2 below are found in $\mathcal{G}_{\text{rdfs}}$ and are premises to the application of the inference rule **rdfs11**, which yields the statement we're after (Statement 3).

Proof:

1. `:Father rdfs:subClassOf :Parent` — P
2. `:Parent rdfs:subClassOf :Person` — P
3. `:Father rdfs:subClassOf :Person` — 1, 2, rdfs11

In the proof above each line is marked with "P" if the statement is a premise, *i.e.*, exits in $\mathcal{G}$, or with the rdfs rule and the line identification of the input statements.

3.2 RDFS inference programmatically

Python. We are using the OWL-RL python library which builds on top of RDFLib and has a RDFS reasoning component.[1] The file in GitHub `RDFSReasoning.py` expands our example graph $\mathcal{G}$ using the RDFS inference rules. A Jupyter notebook is also provided.

Java. We are using the Jena API. The file in GitHub `RDFSReasoning.java` provides an example to set up the reasoner and extend the model with the new triples according to the RDFS inference rules.

Task 6.2: Check if the statements in **Task 6.1** are True or False via SPARQL queries over the extended graph or model (*i.e.*, the graph after applying reasoning). The graph $\mathcal{G}_{\text{rdfs}}$ is provided within the file `lab6-rdfs.ttl`.

[1] Documentation: `https://github.com/RDFLib/OWL-RL`

4 OWL 2 RL entailment

Consider the following set of triples (we will refer to them as the graph $\mathcal{G}_{\text{owl2rl}}$).

```
@PREFIX : <http://city.ac.uk/kg/lab4/>
@PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@PREFIX owl: <http://www.w3.org/2002/07/owl#> .
:Person     a                  owl:Class .
:Man        a                  owl:Class ;
            rdfs:subClassOf    :Person .
:Woman      a                  owl:Class ;
            rdfs:subClassOf    :Person .
:Parent     a                  owl:Class ;
            rdfs:subClassOf    :Person .
:Father     a                  owl:Class ;
            rdfs:subClassOf    :Parent ;
            rdfs:subClassOf    :Man .
:Mother     a                  owl:Class;
            rdfs:subClassOf    :Parent ;
            rdfs:subClassOf    :Woman .
:hasChild   a                  owl:ObjectProperty ;
            owl:inverseOf      :hasParent .
:hasParent  a                  owl:ObjectProperty ;
            rdfs:domain        :Person ;
            rdfs:range         :Parent .
:hasFather  a                  owl:ObjectProperty ;
            rdfs:subPropertyOf :hasParent ;
            rdfs:range         :Father .
:hasMother  a                  owl:ObjectProperty ;
            rdfs:subPropertyOf :hasParent ;
            rdfs:range         :Mother .
:Ann        a                  :Person ;
            :hasFather         :Carl ;
            :hasMother         :Juliet .
```

4.1 Manual entailment

Task 6.3. Indicate if the following statements are derived by $\mathcal{G}_{\text{owl2rl}}$. $\mathcal{G}_{\text{owl2rl}}$ is within the OWL 2 RL profile so one could apply, among many others, similar inference rules to those for RDFS.[2] Indicate in your proof which are the involved triples from $\mathcal{G}_{\text{owl2rl}}$.

Statement 1 `:Juliet :hasChild :Ann .`

Statement 2 `:Ann a :Child .`

[2] https://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules

4.2 OWL2 RL entailment programmatically

Python. We are using the OWL-RL python library. The file `OWLReasoning.py` in GitHub expands our example graph $\mathcal{G}_{\text{owl2rl}}$ using OWL 2 RL reasoning. A Jupyter notebook is also provided.

Java. We are using the Jena API. The file `OWLReasoning.java` in GitHub provides an example to set up the reasoner and to extend the model with the new triples. Jena does not exactly support the OWL 2 RL profile, but includes a set of predefined reasoners. *OWLMiniReasoner* and *OWLMicroReasoner* are the closest to OWL 2 RL.[3]. *OWLMicroReasoner* is less expressive than *OWLMiniReasoner*, but it can be a good alternative if reasoning gets slower with the dataset in the coursework.

Task 6.4. Check programmatically if the above statements (Task 6.3) are True or False via SPARQL queries over the extended graph (*i.e.*, after applying reasoning). The graph $\mathcal{G}_{\text{owl2rl}}$ is provided within the file `lab6-owl2rl.ttl` in the corresponding `lab6` folders.

5 Solutions

5.1 Task 6.1

Statement 2

```
:Woman rdfs:subClassOf :Person .
```

True. This triple is explicitly stated in $\mathcal{G}$.

Statement 3

```
:Juliet a :Person .
```

True. Proof:

1. `:Ann :hasMother :Juliet .` — P
2. `:hasMother rdfs:range :Mother .` — P
3. `:Juliet rdf:type :Mother` — 1, 2, rdfs3
4. `:Mother rdfs:subclassOf :Woman` — P
5. `:Woman rdfs:subclassOf :Person` — P
6. `:Mother rdfs:subclassOf :Person` — 4, 5, rdfs11
7. `:Juliet rdf:type :Person` — 3, 6, rdfs9

[3]Inference in Jena: `https://jena.apache.org/documentation/inference/`

Statement 4

`:Ann a :Child .`

False. It seems intuitive that `:Ann` is a child, but there is not connection between `:Ann` and `:Child` that can be entailed. To get this inference, `:hasFather` and/or `:hasMother` should be declared as sub-property of `:isChildOf`.

Statement 5

`:Ann :isChildOf :Carl .`

False. Similarly to Statement 4.

Statement 6

`:Ann :hasParent :Juliet .`

True. Proof:

1. `:Ann :hasMother :Juliet` — P
2. `:hasMother rdfs:subPropertyOf :hasParent` — P
3. `:Ann :hasParent :Juliet` — 1, 2, rdfs7

Statement 7

`rdfs:range rdf:type rdfs:Resource .`

True. This statement is an axiomatic triple and is always satisfied in RDFS models.

Statement 8

`:Mother rdfs:subClassOf :Person .`

True. Similarly to Statement 1.

5.2 Task 6.2

Solutions added to GitHub. The solutions use `ASK` queries over the extended graph. Check the solution files in the Java and Python repositories, respectively.

5.3 Task 6.3

Statement 1

`:Juliet :hasChild :Ann .`

True. Proof:

1. `:Ann :hasMother :Juliet` — P
2. `:hasMother rdfs:subPropertyOf :hasParent` — P

3. `:hasChild owl:inverseOf :hasParent` — P
4. `:Ann :hasParent :Juliet` — 1, 2, rdfs7[4]
5. `:Juliet :hasChild :Ann` — 3, 4, prp-inv2[5]

Statement 2

`:Ann a :Child .`

False. Ann is the child/daughter of Carl and Juliet but there is not a class `:Child`. For this statement to hold, the range of `:hasChild` should be `:Child`.

5.4 Task 6.4

Solutions added to GitHub. As for the RDFS case, the solutions use `ASK` queries over the extended graph. Check the solution files in the Java and Python repositories, respectively.

[4]Or *prp-spo1* with respect to OWL 2 RL reasoning.

[5]OWL 2 RL reasoning: `https://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules`