Coursework C4:
All about RDF(S)

Intention

The idea of this coursework is to deepen your understanding about RDF(S), more specifically, to

- build an RDFS ontology and to make a simple RDFS inference engine,
- learn the semantics of RDF and
- the semantics of RDFS.

1 Building an RDFS Ontology

Write an RDFS ontology `$yourname$-tourism.ttl` in Turtle syntax that represents the following:

- Each museum is a tourism attraction.
- Each art museum is a museum.
- Each modern art museum is an art museum.
- Each history museum is a museum.
- Each landmark is a tourism attraction.
- Each outdoor spot is a tourism attraction.
- Each amusement park is a tourism attraction.
- Each restaurant is a tourism object.
- Each hotel is a tourism object.
- Each tourism attraction is a tourism object.
- Only tourism objects can have a TripAdvisor rating.

- Each ‘has TripAdvisor rating’ is a ‘has rating’.
- Only hotels can have a star.
- Each ‘has star’ is a ‘has rating’.
- Only museums can have curators.
- Only people can be a curator of a museum.
- Only tourism attractions can have ticket price.
- Only restaurants can have a head chef.
- Each ‘has head chef’ is a ‘has chef’.
- ‘has chef’ has range chef.
- Each chef is a skilled worker.
- Only restaurants can have signature dishes.
- Only restaurants can have the information whether they are a Michelin-starred restaurant.
- Only hotels can have a number of rooms.
- Each hotel chain is a `rdfs:Container`.
- ‘has hotel’ is a `rdfs:ContainerMembershipProperty`.
- Only hotel chains can have hotels.
- ‘has hotel’ has range hotel.

2 Simple RDFS Inference Engine using Jena

The goal of this task is to learn how an inference engine may work by creating a simple RDFS inference engine in Jena using the forward-chaining strategy.

Forward-chaining is a strategy for rule-based inference.[1] Forward-chaining involves applying the inference rules to the known facts (explicit statements) to generate new facts. The rules can then be re-applied to the combination of original facts and inferred facts to produce yet more new facts. The process is iterative and continues until no new facts can be generated. The advantage of this approach is that when all inferences have been computed, query answering over inferred facts can proceed extremely quickly. The disadvantages come from greater initialization costs (inference computed at load time) and space/memory usage (especially when the number of inferred facts is very large).

Your inference engine should meet the following requirements:

[1] `http://owlim.ontotext.com/display/OWLIMv43/OWLIM-SE+Reasoner`

- It is not allowed to use the built-in RDFS inference engine in Jena.
- The engine takes two input parameters: an ontology that stores RDFS axioms and an RDF file that stores RDF data.
- The output will be a Turtle file containing all the inferred data. In your Turtle output, separate the original and new facts using the separator:

  `### NEW FACTS ###`
- You should cover all RDFS entailment rules, except the rule `rdfs1`, at:

  `http://www.w3.org/TR/rdf11-mt/#rdfs_patterns`

  Hint: Use `CONSTRUCT` queries representing the rules to infer new triples.
- Package your engine into a runnable JAR file called `$yourname$-fcRDFS.jar`.

Using your inference engine and your tourism ontology, list new RDFS-inferred facts for each of the following:

- Museion is a modern art museum. The curator of Museion is Francesco Pavarotti.
- The number of rooms in the Greif is 20. The hotel chain BZ-Hotel has two hotels: the Greif and Laurin.
- The TripAdvisor rating of Chicken Hut is 3.5. It doesn't have any Michelin's star. Del Piero is the head chef of Chicken Hut.

Put the new RDF facts of each of the facts above into separate Turtle files `$yourname$-inf1.ttl`, `$yourname$-inf2.ttl` and `$yourname$-inf3.ttl`.

3 Semantics of RDF

Write up all the answers of the following.

3.1 Simple Interpretations

1. Provide a model for this graph G:

```
@prefix : <http://example.org/> .

:laurin :type :Hotel .
:laurin :star "4" .
_:someHotel :star "1" .
```

2. Does the graph G entail the following graph? Explain why?

```
@prefix : <http://example.org/> .

_:b1 :type _:b2 .
```

3.2 RDF Interpretations

1. Recall the graph G. Provide 10 RDF-inferred facts of G.

2. Decide if the following propositions are true or false:

   (a) Blank nodes can stand for arbitrary resources.

   (b) URIs can stand for arbitrary resources.

   (c) Every blank node has an ID.

   (d) Two blank nodes with different IDs can stand for the same resource.

   (e) Two different URIs can stand for the same resource.

   (f) Blank nodes carrying the same ID that occur in several RDF documents must stand for the same resource.

   (g) URIs that occur in several RDF documents must stand for the same resource.

   (h) Two different Literals can never stand for the same value.

   (i) Two Literals with different datatype can never stand for the same value.

   (j) A URI can never stand for a datatype value.

   (k) Blank nodes cannot occur in the predicate position of triples.

   (l) Blank nodes cannot stand for properties (that is, resources that belong to the class `rdf:Property`).

3.3 First-order Logic and SPARQL

Suppose that a vocabulary consists of the IRIs `:hasFriend`, `:likes` and `:Person`.[2]

1. Provide a first-order logic representation of the query "Give me all people who are liked by all their friends".

2. Translate the FOL representation into a SPARQL 1.1 query that has the same intended semantics.

4 Semantics of RDFS

Write up all the answers of the following.

4.1 RDFS Interpretations

1. Represent the rules `rdfs5` and `rdfs9` at `http://www.w3.org/TR/rdf11-mt/#rdfs_patterns` using first-order logic (FOL).

2. Recall the tourism ontology above and the facts "Museion is a modern art museum" and "The curator of Museion is Pavarotti". Derive three new RDFS-inferred facts for each of the facts. In the derivations, mention which RDFS inference rules at `http://www.w3.org/TR/rdf11-mt/#rdfs_patterns` you are using.

[2]Suppose the default prefix is `http://example.org/`.

4.2 RDFS-aware SPARQL Queries

In this task, you are asked to provide SPARQL queries incorporating the RDFS semantics of the following natural-language queries:

1. Give all subclasses of the class `Person`.
2. Give all superclasses of the class `Actor`.
3. Give all subproperties and superproperties of the property `writer`.
4. Is it true that `In_The_Park` co-participated with `Charlie_Chaplin`?

Observe the following query "All classes of Chaplin":

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?supclass WHERE {
dbr:Charlie_Chaplin rdf:type ?typeOfChaplin .
?typeOfChaplin rdfs:subClassOf* ?supclass }
```

Is the query correct, that is, the query will return all classes of Chaplin per the RDFS semantics? If not, please give an explanation and a fix to the query to return all classes of Chaplin correctly and completely?

5 Deliverables

Submit your solutions to `fariz.darari@stud-inf.unibz.it` by 16 January 2015. Your solutions should contain:

- The tourism ontology `$yourname$-tourism.ttl`
- The inference engine `$yourname$-fcRDFS.jar` with the files `$yourname$-inf1.ttl`, `$yourname$-inf2.ttl` and `$yourname$-inf3.ttl` containing new RDF facts.
- Your answers to Section 3 and 4, in a PDF file called `$yourname$-CW4.pdf`.