RDFS and reasoning

Read

- Semantic Web Programming: chapter 4, 5.
- Foundations of Semantic Web Technologies: chapter 2, 3.

1 Entailment

In these exercises we will learn about entailment and decide the logical consequences of RDFS statements.

Let `entailments.n3` be the file listed below, where `rdf` and `rdfs` are the usual namespaces.

```
 1  :Person    a                 rdfs:Class .
 2  :Man       a                 rdfs:Class ;
 3      rdfs:subClassOf    :Person .
 4  :Parent    a                 rdfs:Class ;
 5      rdfs:subClassOf    :Person .
 6  :Father    a                 rdfs:Class ;
 7      rdfs:subClassOf    :Parent ;
 8      rdfs:subClassOf    :Man .
 9  :Child     a                 rdfs:Class ;
10      rdfs:subClassOf    :Person .
11  :hasParent a                 rdf:Property ;
12      rdfs:domain        :Person ;
13      rdfs:range         :Parent .
14  :hasFather a                 rdf:Property ;
15      rdfs:subPropertyOf :hasParent ;
16      rdfs:range         :Father .
17  :isChildOf a                 rdf:Property ;
18      rdfs:domain        :Child ;
19      rdfs:range         :Parent .
20  :Ann       a                 :Person ;
21      :hasFather         :Carl .
22  :Carl      a                 :Man .
```

1.1 Exercise

Is `entailments.n3` syntactically correct RDF(S)?

1.2 Exercise

Assuming the RDFS statements in `entailments.n3` are syntactically correct, are they semantically correct, i.e., do they give an accurate description of "the real world"?

1.3 Exercise

Explain what it means for one set of statements to entail a (different) set of statements.

2 Manual entailment calculation

In the following exercises decide if `entailment.n3` entails the statement(s) given and explain why/why not? If the answer is "yes, the statement(s) is entailed by `entailments.n3`", then use the simple entailment rules (`se1`, `se2`) and the rdfs entailment rules (`rdfs1`, `...`, `rdfs12`) found at RDFS entailment rules[1] to prove your answer. If the answer is "no", then explain, informally or formally, why this is so.

There are quite a few of these exercises, but many of them are quite easy so they should be quick to do. If they are too easy, then skip to the last ones, which are perhaps a bit harder.

2.1 Exercise

First, to get the an overview of the statements in `entailments.n3`, draw a diagram.

2.2 Exercise

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

2.3 Exercise

```
:Man rdfs:subClassOf :Person .
```

2.4 Exercise

```
:Carl a :Person .
```

2.5 Exercise

```
:Carl a :Parent .
```

2.6 Exercise

```
:Carl :hasChild :Ann .
```

[1] http://www.w3.org/TR/rdf-mt/

2.7 Exercise

```
:Carl a :Man .
```

2.8 Exercise

```
:Carl a :Father .
```

2.9 Exercise

```
:Child rdf:type rdfs:Resource .
```

2.10 Exercise

```
:Ann a :Child .
```

2.11 Exercise

```
:Ann :isChildOf :Carl .
```

2.12 Exercise

```
:Ann :hasParent :Carl .
```

2.13 Exercise

```
:Ann :hasParent _:x .
```

2.14 Exercise

```
:Ann :hasParent [ rdf:type :Person ] .
```

2.15 Exercise

```
:hasFather rdfs:domain :Person .
```

2.16 Exercise

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

2.17 Exercise

```
:hasFather rdfs:range :Father .
```

2.18 Exercise

```
:hasFather rdfs:domain [ rdfs:subClassOf :Person ] .
```

2.19 Exercise

```
:Father rdfs:subClassOf [ rdfs:subClassOf :Person ] .
```

3 The Simpson family

Now we will use the family vocabulary as a schema for the Simpsons data we have produced earlier, by opening both files in Protégé. Even though Protégé is an OWL editor it is quite safe to also load RDFS models as Protégé interprets them as OWL ontologies. This is not always the case, so it is best to use OWL if you do not need the meta-modelling capabilities of RDFS.

3.1 Exercise

Open Protégé and choose create a new OWL ontology. Give it the ontology URI

`http://www.ifi.uio.no/INF3580/v16/simpsons.owl`

Save it to a file of your choice, and choose the format you would like to use. What format you choose will not be directly visible in the user interface of Protégé, but is used when saving the ontology to file.

3.2 Exercise

Import the Simpsons RDF file you wrote in the RDF week exercises and the Family RDFS file you have written in this week's exercises. (This is done by clicking the plus sign in the "Ontology Imports" pane on the starting page of Protégé and importing "an ontology contained in a specific file" for each of the RDF files.) Note that Protégé seems to have a problem *importing* files which are not in RDF/XML format, while *opening* files in different formats works better. If you have written your files in a format different from RDF/XML and you have problems with this exercise, try converting your files to RDF/XML with, e.g., RDF Validator and Converter[2] .

Then both files are successfully imported you should see the hierarchy of classes, properties and individuals under the tabs Classes, Object Properties and Data Properties respectively and Individuals. See also if your domain and range assertions look correct.

3.3 Exercise

Find the class Person in the Classes pane and see that it has quite a few members, while the classes Man and Woman have no members.

Now apply reasoning by choosing a reasoner, e.g., Pellet, in the Reasoner menu, and click Classify... in the same menu. Record any error messages that appear, you should get none.

[2] http://www.easyrdf.org/converter

If reasoning was successful, and assuming you have modelled classes and properties the same way as I have, you should see that the classes Man and Woman now have members.

Questions:

- What is the added statements about Bart after reasoning?
- In the Individuals list there might appear new individuals labelled genid1, genid2,... Explain what they are.
- Which named individuals do not get any added information after reasoning?