RDFS and reasoning

Read

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

1 From the lecture

a) Why can we not implement the model-theoretic semantic directly?

b) What 3 types of reasoning does RDFS support?

c) Give examples of some axiomatic triples in RDFS?

d) What are some limitations of RDFS reasoning?

e) What is forward chaining?

f) In which cases is it best to use forward chaining?

1.1 solution

a) We would have to consider all possible interpretations. This is not possible in practice.

b) i Type propagation
   ii Property inheritance
   iii Domain and range reasoning

c)
- rdf:type rdfs:domain rdfs:Resource .
- rdfs:domain rdfs:domain rdf:Property .
- rdfs:range rdfs:domain rdf:Property .
- ...
- For a full list see: `https://www.w3.org/TR/rdf11-mt/#rdfs-interpretations`

d) RDFS cannot express inconsistencies, so any RDFS graph is consistent

e) Forward chaining is reasoning from premises to conclusions of rules, adding the facts corresponding to the conclusions and storing the conclusions in the datastore. Reasoning is done up front.

f) Forward chaining is best for data that is relatively static, expensive to compute, frequently accessed and small enough to store efficiently.

2 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.

```
:Person    a                 rdfs:Class .
:Man       a                 rdfs:Class ;
    rdfs:subClassOf    :Person .
:Parent    a                 rdfs:Class ;
    rdfs:subClassOf    :Person .
:Father    a                 rdfs:Class ;
    rdfs:subClassOf    :Parent ;
    rdfs:subClassOf    :Man .
: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 .
:isChildOf a                 rdf:Property ;
    rdfs:domain        :Child ;
    rdfs:range         :Parent .
:Ann       a                  :Person ;
    :hasFather         :Carl .
:Carl      a                  :Man .
```

:CODE: :END:

2.1 Exercise

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

2.1.1 solution

Yes.

2.2 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"? ("the real world": more in the following lecture)

2.3 Exercise

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

2.3.1 solution

Quoting RDF semantics:

> Entailment is the key idea which connects model-theoretic semantics to real-world applications. As noted earlier, making an assertion amounts to claiming that the world is an interpretation which assigns the value true to the assertion. If A entails B, then any interpretation that makes A true also makes B true, so that an assertion of A already contains the same "meaning" as an assertion of B; one could say that the meaning of B is somehow contained in, or subsumed by, that of A.

3 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 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.

3.1 Exercise

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

3.1.1 solution

Legend: Nodes are represented in the diagram as nodes. A resource of type `rdfs:Class` is depicted with a circle. A resource of type `rdf:Property` is depicted with a diamond. A box is a resource which is not of type `rdfs:Class`

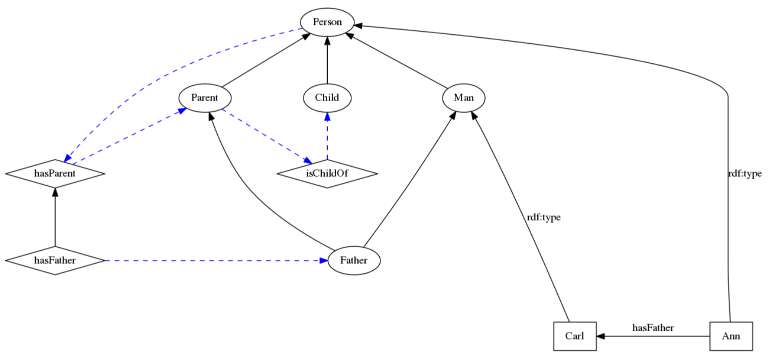

Figure 1: A diagram of the entailments.n3 graph.

or `rdf:Resource`. The resources `rdfs:Class`, `rdf:Property`, `rdf:type`, `rdfs:subClassOf` and `rdfs:subPropertyOf` are not marked by nodes in the diagram.

Triples are illustrated with edges, where the node which the edge points from is the subject, the edge itself is the predicate and the object is the node which the edge points to. A solid unlabelled edge represent an `rdfs:subClassOf` or a `rdfs:subPropertyOf` predicate. Blue, dashed edges show domain and range restrictions for properties. An edge pointing to a property represents the domain restriction for the property, a range restriction is marked with an outwards pointing edge.

3.2 Exercise

`:Father rdfs:subClassOf :Person .`

3.2.1 solution

True. `:Father` is (transitively) a subclass of `:Person`. Rule rdfs11.

In the proof below each line is marked with "P" if the statement is a premise, i.e., exists in `entailments.n3`, or with the rule and the input statements to this rule by which the line in question is concluded.

Proof:

1. `:Father rdfs:subClassOf :Parent` — P

2. `:Parent rdfs:subClassOf :Person` — P

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

Statements 1. and 2. are found in `entailments.n3` and are premises to the application of the entailment rule rdfs11 on line 3, which yields the statement we're after.

3.3 Exercise

`:Man rdfs:subClassOf :Person .`

3.3.1 solution

True. This is explicitly stated in `entailments.n3`, so the entailment is trivial.

3.4 Exercise

`:Carl a :Person .`

3.4.1 solution

True. `:Carl` is a `:Man`. `:Man` is a subclass of `:Person`. Thus, `:Carl` is a `:Person`. Rule rdfs9. Proof:

1. `:Man rdfs:subclassOf :Person` — P
2. `:Carl rdf:type :Man` — P
3. `:Carl rdf:type :Person` — 1, 2, rdfs9

3.5 Exercise

`:Carl a :Parent .`

3.5.1 solution

True. Carl is in the range of `hasFather` (`:Ann :hasFather :Carl`). The range of `hasFather` is `:Father`. Thus, `:Carl` is a `:Father`. `:Father` is a subclass of `:Parent`, which makes `:Carl` a `:Parent`. Rule rdfs3 and rdfs9. Proof:

1. `:hasFather rdfs:range :Father` — P
2. `:Ann :hasFather :Carl` — P
3. `:Carl rdf:type :Father` — 1, 2, rdfs3
4. `:Father rdfs:subClassOf :Parent` — P
5. `:Carl rdf:type :Parent` — 4, 3, rdfs9

3.6 Exercise

`:Carl :hasChild :Ann .`

3.6.1 solution

False. The predicate `:hasChild` does not exist in `entailments.n3`.

3.7 Exercise

`:Carl a :Man .`

3.7.1 solution

True. Trivial. Statement is in `entailments.n3`.

3.8 Exercise

`:Carl a :Father .`

3.8.1 solution

True. See `:Carl a :Parent`.

3.9 Exercise

`:Child rdf:type rdfs:Resource .`

3.9.1 solution

True. Proof:

1. `:Child rdf:type rdfs:Class` — P
2. `:Child rdf:type rdfs:Resource` 1, rdfs4a

3.10 Exercise

`:Ann a :Child .`

3.10.1 solution

False. The statements about Ann in `entailments.n3` are `:Ann a Person` and `:Ann :hasFather :Carl`. None of these statements force us to conclude that `:Ann` is a `:Child`.

3.11 Exercise

`:Ann :isChildOf :Carl .`

3.11.1 solution

False. See `:Ann a :Child`.

3.12 Exercise

`:Ann :hasParent :Carl .`

3.12.1 solution

True. `:Ann :hasFather :Carl` is a statement in `entailments.n3`. `:hasFather` is a subproperty of `:hasParent`. This means that all pairs related by the `:hasFather` property are also related by the `:hasParent` property. Rule rdfs7. Proof:

1. `:Ann :hasFather :Carl` — P
2. `:hasFather rdfs:subPropertyOf :hasParent` — P
3. `:Ann :hasParent :Carl` — 2, 1, rdfs7

3.13 Exercise

`:Ann :hasParent _:x .`

3.13.1 solution

True. This follows from the results from the previous exercise. Since it is true that "Ann has a parent Carl", then it is true that "Ann has *some* parent". This is a conclusion by the application of a simple entailment rule. Proof:

1. `:Ann :hasParent :Carl` — P (from previous exercise)
2. `:Ann :hasParent _:x` — 1, se1

3.14 Exercise

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

3.14.1 solution

True. This follows from the results of the previous exercise and the fact that the fact that the range of the property `:hasParent` is `:Person`.

1. `:Ann :hasParent _:x` — P (from previous exercise)
2. `:hasParent rdfs:range :Parent` — P
3. `_:x rdf:type :Parent` — 2, 1, rdfs3
4. `:Parent rdfs:subClassOf :Person` — P
5. `_:x rdf:type :Person` — 4, 3, rdfs9

`:Ann :hasParent [ rdf:type :Person ]` is just an other form of writing the statements 1. and 5.

3.15 Exercise

`:hasFather rdfs:domain :Person .`

3.15.1 solution

False. No RDFS entailment rule lets one derive a statement (see right-most column in the RDFS entailment rules table) about `rdfs:domain`.

With the "Extensional Entailment Rules" (`ext1`, ..., `ext9`) the entainment would be true, however, there rules are not included in the rdfs-rules, quoting the RDF Semantics document:

> The semantics given [above] is deliberately chosen to be the weakest 'reasonable' interpretation of the RDFS vocabulary. Semantic extensions MAY strengthen the range, domain, subclass and subproperty semantic conditions [...].

3.16 Exercise

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

3.16.1 solution

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

3.17 Exercise

`:hasFather rdfs:range :Father .`

3.17.1 solution

True. Trivial. Statement is in `entailments.n3`.

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

3.17.2 Solution

False. See `:hasFather rdfs:domain :Person .` exercise.

3.18 Exercise

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

3.18.1 solution

True. Proof:

1. `:Father rdfs:subClassOf :Parent` — P
2. `:Father rdfs:subClassOf _:n` — 1, se1
3. `:Parent rdfs:subClassOf :Person` — P
4. `_:n rdfs:subClassOf :Person` — 3, se2

Combine 2. and 4. to get `:Father rdfs:subClassOf [ rdfs:subClassOf :Person ].`