Foundations of Semantic Web Technologies

Tutorial 9

Dörthe Arndt

WS 2023/24

Exercise 9.1. Consider the following N3-graph:

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

# characters
:vito a :Person, :Male.
:carmela a :Person, :Female.
:conny a :Person, :Female.
:micheal a :Person, :Male.
:fredo a :Person, :Male.
:sonny a :Person, :Male.
:vincent a :Person, :Male.
:marry a :Person, :Female.

#relations
:vito :spouse :carmela.
:carmela :hasChild :michael, :fredo, :conny, :vincent.
:michael :hasChild :marry.
:sonny :hasChild :vincent.
```

This graph ia also preloaded at: `https://editor.notation3.org/s/1xzClC24`.

- Find a rule to express that all children of Camela are also children of Vito.
- Define one rule to state that the `:spouse`-relation is symmetric.
- Add the triple `:spouse a owl:SymmetricProperty.` and add a rule to handle symmetric properties in general (do not forget to uncomment the previous rule if you test.).
- Define rules for the relations `:isWifeOf`, `:isHusbandOf` and `:isGrandchildOf`.

Exercise 9.2. Define a rule for the graph in the previous exercise to define the concept of `:hasMother`. You should be able to derive that, e.g., `:carmela :motherOf :michael.` Now add a second rule saying that every person has a mother. What do you observe if you change the order of these two rules? How would you explain that difference?

Exercise 9.3. Consider the set-up as in exercise 8.5: The game *Sudoku* is about completing incomplete tables with numbers while respecting certain rules. We consider the following simple 4×4 Sudoku:

			3
			4
2			
3			

You have to fill in numbers with values 1 to 4 in the empty slots in the table so that no number occurs twice in any row or any column, and so that no number is duplicated within any of the marked 2×2 squares.

Instead of a SPARQL query, we would like to use N3 to solve the game, i.e. we want to obtain all possible solutions by means of answers to N3 rules. In order to do this, set up a suitable RDF document (you can also re-use the document from exercise 8.4) and one or more N3 rules. You find a list of built-ins at `https://w3c.github.io/N3/reports/20230703/builtins.html`. Try your solution at: `https://editor.notation3.org/`. If your solution contains built-in functions, try different orders of built-ins.

Exercise 9.4. Recall the concept of property paths in SPARQL. To retrieve people and their descandants from the graph in exercise 9.1, you could for example write:

```
Select  ?x ?y
Where {
?x :hasChild+ ?y.
}
```

Write rules to define the concept `:hasDescandent` which leads to the same data as the query above.

Exercise 9.5. In lecture 8, slide 63 we had the following graph:

```
:MysticRiver :actor :KBacon, :TRobbins.
:TopGun :actor :TRobbins, :MRyan.
:CityOfAngles :actor :MRyan, :NCage.
```

There we had a query to find persons with finite Bacon numbers.

- Write rules to calculate the bacon numbers for all actors present. For simplicity, you can assume that every person only has one bacon number.
- How could you modify your rules to cope with multiple bacon numbers. In that case we would like to find and output the smallest (it is enough to explain an idea).