Foundations of Semantic Web Technologies

Tutorial 7

Dörthe Arndt

WS 2023/24

Exercise 7.1. Today we will query an RDF graph about the Standard Model of Particle Physics using SPARQL. We will use RDF Playground (`http://rdfplayground.dcc.uchile.cl/`). Copy and paste the following RDF graph describing 19 sub-atomic particles into the text field on the left-hand side and view it as a graph.

```
@prefix : <http://ex.org/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

:Up a :Quark ; :spin 0.5 ; :generation 1 ; :interaction :Strong , :EM , :Weak .
:Down a :Quark ; :spin 0.5 ; :generation 1 ; :interaction :Strong , :EM , :Weak .
:Charm a :Quark ; :spin 0.5 ; :generation 2 ; :interaction :Strong , :EM , :Weak .
:Strange a :Quark ; :spin 0.5 ; :generation 2 ; :interaction :Strong , :EM , :Weak .
:Top a :Quark ; :spin 0.5 ; :generation 3 ; :interaction :Strong , :EM , :Weak .
:Bottom a :Quark ; :spin 0.5 ; :generation 3 ; :interaction :Strong , :EM , :Weak .
:Electron a :Lepton ; :spin 0.5 ; :generation 1 ; :interaction :Weak .
:ElectronNeutrino a :Lepton ; :spin 0.5 ; :generation 1 ; :interaction :Weak .
:Muon a :Lepton ; :spin 0.5 ; :generation 2 ; :interaction :Weak .
:MuonNeutrino a :Lepton ; :spin 0.5 ; :generation 2 ; :interaction :Weak .
:Tau a :Lepton ; :spin 0.5 ; :generation 3 ; :interaction :Weak .
:TauNeutrino a :Lepton ; :spin 0.5 ; :generation 3 ; :interaction :Weak .
:Gluon a :GaugeBoson ; :spin 1 ; :mediates :Strong .
:Photon a :GaugeBoson ; :spin 1 ; :mediates :EM  .
:ZBoson a :GaugeBoson ; :spin 1 ; :mediates :Weak  .
:WBoson a :GaugeBoson ; :spin 1 ; :mediates :Weak  .
:Higgs a :ScalarBoson ; :spin 0 .

:Neutron a :Baryon ;
:contains [ :component :Up ; :quantity 1 ] , [ :component :Down ; :quantity 2 ] .
:Proton a :Baryon ;
:contains [ :component :Up ; :quantity 2 ] , [ :component :Down ; :quantity 1 ] .
```

For each of the following questions your answer should be a SPARQL query. You can test these queries by using the SPARQL tab of the right-hand side. Be sure to add `PREFIX : <http://ex.org/>` to make your life easier. When you write a query select the operation type (unless otherwise stated use `SELECT` queries) and whatever result format you prefer (*Text* is best for `SELECT` and `ASK`, *TTL* for `CONSTRUCT`). The queries should not introduce more information than in the question; for example, if the question asks for the force mediated by gluons, you cannot simply insert `:Strong` in the query, but rather the query should find `:Strong`. Note that the term "particle" is used like "resource" or "entity" in order to phrase the questions more naturally; you do not need to explicitly check that something is of type particle in the query (though for more specific types like leptons, muons, etc., you should check the type). *Let's begin!*

(a) Find the spin of a photon.

(b) Find particles that mediate the weak force and their spin.

(c) Find particles that interact through the force mediated by gluons.

(d) Find the unique forces through which each baryon interacts based on its components.

(e) Find particles whose spin is a whole number.

(f) Find particles that interact through the force mediated by gluons or photons.

(g) Find particles with positive spin (> 0) and, if given, their generation (if not, still return the result).

(h) Find particles that interact through the weak force but not the strong force.

(i) Find if there are particles with lower (<) spin than a muon (use `ASK`).

(j) Create the following graph connecting quarks in the next or previous generation (use `CONSTRUCT`).

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

:Up :nextGeneration :Charm , :Strange .
:Down :nextGeneration :Charm , :Strange .
:Charm :nextGeneration :Top , :Bottom ; :prevGeneration :Up , :Down .
:Strange :nextGeneration :Top , :Bottom ; :prevGeneration :Up , :Down .
:Top :prevGeneration :Charm , :Strange .
:Bottom :prevGeneration :Charm , :Strange .
```

Exercise 7.2. Wikidata is a dataset proposed to better organise the data of Wikipedia (used for lists, tables, infoboxes, etc.) in a central, language-agnostic, structured repository. Wikidata is used not only by Wikipedia, but also Google[1], Alexa[2], Siri[3], etc. The dataset is published using Semantic Web standards and provides a SPARQL endpoint that we will use for today's lab to try out some SPARQL queries.

Navigate to `http://query.wikidata.org/`. Here you will find the SPARQL interface for Wikidata. Try this query, which returns the Spanish names of Chilean cities:

```
SELECT ?cityLabel
WHERE {
  ?city wdt:P31/wdt:P279* wd:Q515 ;
  wdt:P17 wd:Q298 ;
  rdfs:label ?cityLabel .
  FILTER (lang(?cityLabel)="es")
}
```

Note the following:

- Rather than use identifiers like `ex:Bejing`, which are biased to a particular language , Wikidata uses numeric identifiers like `wd:Q956`; as an additional benefit, when a city changes its name from Peking to Bejing, for example, it is not necessary to change the identifier or maintain multiple identifiers. The downside is that the query is more difficult to read but you can hover over the IRI term in the query interface to get an explanation.
- In order to find identifiers for entities (like `wd:Q515`) you can write `wd:` in the interface, hold Ctrl and press Spacebar; this opens a search box were you can type (e.g.) `chile` and get suggested identifiers. On the other hand, for properties, you can type `wdt:` and then hold Ctrl and press Spacebar.
- Wikidata uses its own properties for some properties already defined by the standards. The property `wdt:P31` denotes INSTANCE-OF, which serves the same role as `rdf:type`. The property `wdt:P279` denotes SUBCLASS-OF which serves the same role as `rdfs:subClassOf`. The expression `wdt:P31/wdt:P279*` is a property path that matches nodes connected to `?city` by one `wdt:P31` edge and zero-or-many `wdt:P279` edges; this pattern matches instances of classes and their transitive sub-classes.

[1] `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44818.pdf`
[2] `https://www.wired.com/story/inside-the-alexa-friendly-world-of-wikidata/`
[3] `https://io9.gizmodo.com/siri-erroneously-told-people-stan-lee-was-dead-1827322243`

- Wikidata provides human-readable labels for entities in many languages; the filter asks for Spanish labels.

In the following queries you should return names in English where applicable (you'll find more results for English).[4] You should return distinct results in each case. You can assume that each entity has one English label (using `rdfs:label`), but you should not assume that each English label is unique to that entity. Note that if your favourite entity of type x provides empty results, you should select your other favourite entity of type x.

(a) Find the names of video games set on Mars.

(b) Find the names of the *other* videogames in the same series as your favourite video game (the results should not include your favourite video game).

(c) Find the name of the highest grossing video game and its revenue.

(d) Find the names of video games directed by women and optionally their direct sequel; return the name of the video game, the director, and the sequel (if any).

(e) Find the names of video game series with at least 25 video game characters; return also the number of characters (descending order) and the name of a random character.

(f) Find the names of exclusive video games for the Sega Dreamcast (only released on that platform) and how many *transitive* sequels they had (returning 0 if they had none[5]).

(g) Find the names of video game characters that appear in more than one video game series and how many different video games they have appeared in.

[4] Use `FILTER(lang(?someLabel)="en")`

[5] Some sequels might be missing ... you could add them to Wikidata if you wished!