Foundations of Semantic Web Technologies

Tutorial 10

Dörthe Arndt

WS 2023/24

Exercise 10.1. Reconsider the graph from Exercise 4.1 (Tutorial 5[1]). We want to solve the same exercise with N3 rules. To test your rules, go to `https://editor.notation3.org/`.

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

ex:Japan ex:name "Japan"@en , "Japón"@es ;
ex:capitalCity ex:TokyoCity ;
ex:alpha2code "JP" .

ex:Kanto a ex:Region ;
ex:country ex:Japan .

ex:Tokyo a ex:Prefecture ;
ex:region ex:Kanto ;
ex:capitalCity ex:TokyoCity .

ex:TokyoCity a ex:City ;
ex:prefecture ex:Tokyo ;
ex:replaced ex:Edo .

ex:Edo a ex:Prefecture , ex:FormerPrefecture ;
ex:name "Edo" ;
ex:region ex:Kanto ;
ex:country ex:Japan .

ex:Saitama a ex:Prefecture ;
ex:sharesBorderWith ex:Tokyo ;
ex:region ex:Kanto .

ex:Musashimurayama a ex:City ;
ex:prefecture ex:Tokyo .

ex:MasahiroSakurai ex:name "Masahiro Sakurai" ;
ex:knownAs "Masa Sakurai" , "Sakurai" ;
ex:placeOfBirth ex:Musashimurayama .

ex:Nippon ex:alpha2code "JP" ;
ex:capitalCity ex:Tokyo .

ex:SupremeCourtOfJapan a ex:SupremeCourt ;
ex:country ex:Japan .

ex:SaikoSai a ex:SupremeCourt ;
ex:country ex:Japan .
```

[1] confusing, I know, sorry for that.

(a) Write four N3 rules to derive from the existing data:

```
ex:Japan ex:hasPart ex:TokyoCity .
ex:Tokyo ex:hasPart ex:TokyoCity .
ex:Nippon ex:hasPart ex:Tokyo .

ex:Kanto ex:isPartOf ex:Japan .
ex:Edo ex:isPartOf ex:Japan .
ex:SupremeCourtOfJapan ex:isPartOf ex:Japan .
ex:SaikoSai ex:isPartOf ex:Japan .

ex:Tokyo ex:isPartOf ex:Kanto .
ex:Edo ex:isPartOf ex:Kanto .
ex:Saitama ex:isPartOf ex:Kanto .

ex:TokyoCity ex:isPartOf ex:Tokyo .
ex:Musashimurayama ex:isPartOf ex:Tokyo .
```

(b) Add two N3-rules to infer the triples:

```
ex:TokyoCity ex:isPartOf ex:Japan .
ex:Tokyo ex:isPartOf ex:Nippon .

ex:Japan ex:hasPart ex:Kanto , ex:Edo , ex:SupremeCourtOfJapan , ex:SaikoSai .

ex:Kanto ex:hasPart ex:Tokyo , ex:Edo , ex:Saitama .

ex:Tokyo ex:hasPart ex:Musashimurayama .
```

(c) Add one N3 rule to infer triples of the following form:

```
ex:Japan ex:hasPart ex:Tokyo , ex:Saitama , ex:Musashimurayama .
ex:Nippon ex:hasPart ex:Musashimurayama , ex:TokyoCity .

ex:Tokyo ex:isPartOf ex:Japan .
ex:Saitama ex:isPartOf ex:Japan .
ex:Musashimurayama ex:isPartOf ex:Japan .
ex:Musashimurayama ex:isPartOf ex:Nippon .
ex:TokyoCity ex:isPartOf ex:Nippon .
```

(d) Add one N3-rule to infer the following triple:

```
ex:Tokyo ex:sharesBorderWith ex:Saitama .
```

(e) Add one N3 rule to infer the following triple:

```
ex:TokyoCity ex:formerlyKnownAs "Edo" .
```

(f) Add one N3-rule to infer the following triple:

```
ex:MasahiroSakurai ex:countryOfBirth ex:Japan .
```

(g) Add three N3-rules to infer the following triples:

```
ex:Japan owl:sameAs ex:Nippon .
ex:Nippon owl:sameAs ex:Japan .

ex:Japan ex:capitalCity ex:Tokyo .

ex:Nippon ex:name "Japan"@en .
```

```
# and so forth, duplicating all
# triples for ex:Japan to ex:Nippon
# and vice versa
```

(h) Add three N3-rules to infer the following triples:

```
ex:Tokyo owl:sameAs ex:TokyoCity .
ex:TokyoCity owl:sameAs ex:Tokyo .

ex:Tokyo ex:capitalCity ex:Tokyo ;
ex:replaced ex:Edo .

ex:TokyoCity a ex:Prefecture .

# and so forth, duplicating all
# triples for ex:Tokyo to ex:TokyoCity
# and vice versa
```

(i) Add one N3-rule to infer the following triples:

```
ex:SupremeCourtOfJapan owl:sameAs ex:SaikoSai .
ex:SaikoSai owl:sameAs ex:SupremeCourtOfJapan .
```

(j) Add one N3-rule to state that something cannot have itself as a capital city. This should give two inconsistencies (look for `err:ErrorMessage`): one for `ex:Tokyo` and another for `ex:TokyoCity`.

Exercise 10.2. The solution of exercise 4.1 is given in the following RDF-graph:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex: <http://ex.org/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

# a
ex:capitalCity rdfs:subPropertyOf ex:hasPart .
ex:country rdfs:subPropertyOf ex:isPartOf .
ex:region rdfs:subPropertyOf ex:isPartOf .
ex:prefecture rdfs:subPropertyOf ex:isPartOf .

#b
ex:hasPart owl:inverseOf ex:isPartOf .

# c
ex:hasPart a owl:TransitiveProperty .

# d
ex:sharesBorderWith a ex:SymmetricProperty .

#e
ex:formerlyKnownAs owl:propertyChainAxiom ( ex:replaced ex:name ) .

# f
ex:countryOfBirth owl:propertyChainAxiom ( ex:placeOfBirth ex:isPartOf ex:country ).

# g
ex:alpha2code a owl:InverseFunctionalProperty .

#h
ex:capitalCity a owl:FunctionalProperty .

# i
ex:SupremeCourt owl:hasKey ( ex:country ) .
```

```
# j
ex:capitalCity a owl:IrreflexiveProperty .
```

Write rules which –based on these triples– automatically produce the rules you found as a solution for exercise 10.1. For `owl:propertyChainAxion` it suffices, if you explicitely write rules for lists of length 2 and 3. The same holds for `owl:hasKey` where it enough to write the rules for lists with length 1.