Foundations of Semantic Web Technologies

Tutorial 4

Dörthe Arndt

WS 2022/23

Exercise 4.1. Model the following statements in OWL DL by giving the corrsponding OWL/RDF snippets:

- The class `Vegetable` is a subclass of `PizzaTopping`.
- The class `PizzaTopping` does not have common elements with the class `Pizza`.
- The individual `aubergine` is an element of the class `Vegetable`.
- The abstract role `hasTopping` connects only elements of the class `Pizza` with elements of the class `PizzaTopping`.
- Pizzas always have at least two toppings.
- Every Pizza from the class `PizzaMargarita` has `Tomato` as topping.
- The class `VegetarianPizza` consists of those individuals that are both in the class `PizzaWithoutMeat` and in the class `PizzaWithoutFish`.
- No Pizza from the class `PizzaMargarita` has a topping from the class `Meat`.

Exercise 4.2. Decide, if the following statements would make sense in the context of the pizza ontology from Exercise 4.1:

- The role `hasIngredient` is transitive.
- The role `hasTopping` is functional.
- The role `hasTopping` is inverse functional.
- The role `hasIngredient` is asymmetric.

Exercise 4.3. Assume a vocabulary with the individual names `bonny` and `clyde`, the class names `Honest`, `Wise`, `Crime` and `Human` as well as the role names `commits`, `marriedWith`, `suspects`, `report` and `know`.

Which of the following propositions can be made in OWL 1, which in OWL 2 and which ones not at all? In the positive case, provide the corresponding axioms.

(a) Everybody, who is honest and who commits a crime, reports himself.

(b) Who is wise and honest, doesn't commit crimes.

(c) Bonnie does not report Clyde.

(d) Nobody reports a human, which whom he has committed a crime jointly.

(e) Clyde has committed at least 10 crimes.

(f) Bonnie and Clyde have committed at least one crime together.

(g) Who committed a crime together with his/her spouse is not honest.

(h) Everybody knowing a suspect, is a suspect himself.

Exercise 4.4. We will use OWL to define some semantics for properties relating to places and use these definitions to infer new data in the RDF graph. Navigate to `http://rdfplayground.dcc.uchile.cl`. Copy and paste the following data into the text field on the left side of the page:

```
@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 .
```

Select the *OWL* tab on the right side and copy over the following to that side:

```
@prefix ex:  <http://ex.org/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

ex:knownAs owl:equivalentProperty ex:alias .
```

Select the *OWL* reasoner. Append the necessary RDFS/OWL axioms in the right-hand side text field to answer the following questions (note that you cannot add the required data explicitly; it must be inferred through the requested

OWL axioms; also note that OWL includes the most important features of RDFS). Press the *INFER OVER GRAPH* button to verify that the desired triples are correctly inferred. For the reasoner to work, the Turtle syntax be valid. The answer to each question should be kept and extended (some questions depend on previous answers). In the case of an inconsistency ~~the world will implode~~ the reasoner will add one or more messages of the following form to the output data (this is specific to this reasoner implementation).

```
[] a err:ErrorMessage ; err:error "..." .
```

(a) Add four RDFS axioms to infer the triples:

```
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 one OWL axiom 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 OWL axiom 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 OWL axiom (not using inverse-of) to infer the following triples:

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

(e) Add one OWL axiom to infer the following triple:

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

(f) Add an OWL axiom to infer the following triple:

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

(g) Add an OWL axiom 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 an OWL axiom 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 an OWL axiom to infer the following triples:

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

(j) Add one OWL axiom 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`.