Foundations of Semantic Web Technologies
Solutions for Tutorial 3: OWL
Sebastian Rudolph
SS 2015

Solution (3.1).

```xml
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>
<rdf:RDF
  xmlns="http://example.org/
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <owl:Class rdf:about="Vegetable">
    <rdfs:subClassOf rdf:resource="PizzaTopping"/>
  </owl:Class>
  <owl:Class rdf:about="Pizza">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="hasTopping"/>
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
          2
        </owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="PizzaTopping"/>
  </owl:Class>
  <owl:NamedIndividual rdf:about="Aubergine">
    <rdf:type rdf:resource="Vegetable"/>
  </owl:NamedIndividual>
</rdf:RDF>
```
<Vegetable rdf:about="Aubergine"/>

<owl:ObjectProperty rdf:about="hasTopping">
  <rdfs:domain rdf:resource="Pizza"/>
  <rdfs:range rdf:resource="PizzaTopping"/>
</owl:ObjectProperty>

<owl:Class rdf:about = "PizzaMargaritta">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasTopping"/>
      <owl:someValuesFrom rdf:resource="Tomato"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="hasTopping"/>
          <owl:someValuesFrom rdf:resource="Meat"/>
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="VegetarianPizza">
  <rdfs:subClassOf rdf:resource="PizzaWithoutFish"/>
  <rdfs:subClassOf rdf:resource="PizzaWithoutMeat"/>
</owl:Class>

oder

<owl:Class rdf:about="VegetarianPizza">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:resource="PizzaWithoutFish"/>
    <owl:Class rdf:resource="PizzaWithoutMeat"/>
  </owl:intersectionOf>
</owl:Class>

</rdf:RDF>
In description logic syntax:

\[
\begin{align*}
\text{Vegetable} & \sqsubseteq \text{PizzaTopping} \\
\text{Pizza} & \sqsubseteq 2 \text{ hasTopping}, \top \\
\text{Pizza} \sqcap \text{PizzaTopping} & \sqsubseteq \bot
\end{align*}
\]

Alternatives

\[
\begin{align*}
\text{Pizza} & \sqsubseteq \neg \text{PizzaTopping} \\
\text{Vegetable}(\text{Aubergine}) & \\
\top & \sqsubseteq \forall \text{hasTopping}. \text{PizzaTopping} \\
\exists \text{hasTopping}. \top & \sqsubseteq \text{Pizza} \\
\text{PizzaMargarita} & \sqsubseteq \exists \text{hasTopping}. \text{Tomato}
\end{align*}
\]

or, in the case tomato is modeled as individual

\[
\begin{align*}
\text{PizzaMargarita} & \sqsubseteq \exists \text{hasTopping}. \{\text{Tomato}\} \\
\text{PizzaMargarita} & \sqsubseteq \neg \exists \text{hasTopping}. \text{Meat} \\
\text{VegetarianPizza} & \sqsubseteq \text{PizzaWithoutFish} \\
\text{VegetarianPizza} & \sqsubseteq \text{PizzaWithoutMeat}
\end{align*}
\]

Solution (3.2).

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

Solution (3.3).

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

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType = "Collection">
    <owl:Class rdf:resource = "Honest" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="commits"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
In DL syntax: Honest $\cap \exists$commits.Crime $\subseteq \exists$reports.Self

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

In DL syntax: Honest $\cap$ Wise $\subseteq \neg$$\exists$commits.Crime

In NNF: Honest $\cap$ Wise $\subseteq \forall$commits.($\neg$Crime)

(c) Bonnie does not report Clyde.

In DL syntax: Bonnie $\subseteq \neg$$\exists$reports.Clyde
In DL syntax: \{Bonnie\} \subseteq \neg \exists \ reports. \{Clyde\}

In OWL 2:

```owl
<owl:NegativePropertyAssertion>
  <owl:sourceIndividual rdf:about="Bonnie"/>
  <owl:assertionProperty rdf:about="reports"/>
  <owl:targeIndividual rdf:about="Clyde"/>
</owl:NegativePropertyAssertion>
```

In DL syntax: \neg \ reports(Bonnie, Clyde)

(d) Nobody reports a human, which whom he has committed a crime jointly.
    Not expressible in OWL.

(e) Clyde has committed at least 10 crimes.

```owl
<rdf:Description rdf:about="Clyde">
  <rdf:type>
    <owl:Restriction>
      <owl:onProperty rdf:resource="commits"/> 
      <owl:onClass rdf:resource="Crime"/>
      <owl:maxQualifiedCardinality xr:nonNegativeInteger>10</owl:maxQualifiedCardinality>
    </owl:Restriction>
  </rdf:type>
</rdf:Description>
```

In DL syntax: \( \geq 10 \ commits.\ Crime)(Clyde)

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

```owl
<owl:Class>
  <owl:oneOf rdf:parseType="Collection">
    <rdf:Description rdf:about="Bonnie"/>
  </owl:oneOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="commits"/>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
```
In DL Syntax: \{Bonnie\} \sqsubseteq \exists commits.(Crime \cap \exists commits^\sim .\{Clyde\})

What does that mean? We say that Bonnie is a Person who has committed a crime, which was also committed (inverse of has committed) by Clyde.

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

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

In DL syntax: suspects \circ knows^\sim \sqsubseteq \text{suspects}
(c) ...has no topping which is meat, and no topping which is fish.

(d) ...has a topping which is not meat, and a topping which is not fish.

(e) ...has only ingredients, which are neither meat nor fish.

Consequently, only (c) and (e) are reasonable definitions. Whereas (e) is more precise, since also meat in the ingredients of the base are excluded.

**Solution (3.5).**

We consider axiom (c) from the previous exercise. When using (e) we would have to modify the axioms further in order to not only ensure vegetarian toppings, but also vegetarian ingredients. Solution for (a) and (b):

- **CheesePizza ≡ Pizza ⊓ ∃ hasTopping.Cheese**  
  Not a subclass of VegetarianPizza. **Correction:**  
  CheesePizza ≡ Pizza ⊓ ∃ hasTopping.Cheese ⊓ ∀ hasTopping.Cheese

- **PizzaSpinach ≡ ∃ hasTopping.Spinach ⊓ ∃ hasTopping.Cheese ⊓ ∀ hasTopping.(Spinach ⊔ Cheese)**  
  Not a subclass of VegetarianPizza. **Correction:** add the axiom Spinach ⊑ Gemüse

- **PizzaCarnivorus ≡ Pizza ⊓ ∀ hasTopping.(Meat ⊓ Fish)**  
  Not a subclass of VegetarianPizza. **Correction:**  
  PizzaCarnivorus ≡ Pizza ⊓ ∀ hasTopping.(Meat ⊔ Fish)

- **EmptyPizza ≡ Pizza ⊓ ¬ ∃ hasTopping.⊤**  
  Subclass of VegetarianPizza (no correction needed).

(c) Changed axiom:

VegetarianPizza ⊑ Pizza ⊓ ∀ hasIngredient.(¬ Meat ⊓ ¬ Fish)

No class would be classified as subclass of VegetarianPizza anymore, e.g., it still holds:

CheesePizza ⊑ Pizza ⊓ ∀ hasIngredient.(¬ Meat ⊓ ¬ Fish)

which does not imply

CheesePizza ⊑ VegetarianPizza