# Foundations of Semantic Web Technologies

Solutions for Tutorial 3: OWL

Sebastian Rudolph

SS 2015

**Solution** (3.1)**.**

```
<?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>
or
```

```
  <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:

$$\begin{aligned} \text{Vegetable} &\sqsubseteq \text{PizzaTopping} \\ \text{Pizza} &\sqsubseteq \geqslant 2\ \text{hasTopping}.\top \\ \text{Pizza} \sqcap \text{PizzaTopping} &\sqsubseteq \bot \end{aligned}$$

Alternatives

$$\begin{aligned} \text{Pizza} &\sqsubseteq \neg\text{PizzaTopping} \\ &\text{Vegetable}(\text{Aubergine}) \\ \top &\sqsubseteq \forall\text{hasTopping.PizzaTopping} \\ \exists\text{hasTopping}.\top &\sqsubseteq \text{Pizza} \\ \text{PizzaMargaritta} &\sqsubseteq \exists\text{hasTopping.Tomato} \end{aligned}$$

or, in the case tomato is modeled as individual

$$\begin{aligned} \text{PizzaMargaritta} &\sqsubseteq \exists\text{hasTopping}.\{\text{Tomato}\} \\ \text{PizzaMargaritta} &\sqsubseteq \neg\exists\text{hasTopping.Meat} \\ \text{VegetarianPizza} &\sqsubseteq \text{PizzaWithoutFish} \\ \text{VegetarianPizza} &\sqsubseteq \text{PizzaWithoutMeat} \end{aligned}$$

**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.

```
<owl:Class>
   <owl:intersectionOf rdf:parseType = "Collection">
      <owl:Class rdf:resource = "Honest" />
      <owl:Restriction>
         <owl:onProperty rdf:resource="commits"/>
         <owl:someValuesFrom rdf:resource="Crime"/>
      </owl:Restriction>
   </owl:intersectionOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="reports"/>
         <owl:hasSelf rdf:datatype="&xsd;boolean">true</owl:hasSelf>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

In DL syntax: $\text{Honest} \sqcap \exists\text{commits.Crime} \sqsubseteq \exists\text{reports.}\mathsf{Self}$

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

```
<owl:Class>
   <owl:intersectionOf rdf:parseType = "Collection">
      <owl:Class rdf:resource = "Honest" />
      <owl:Class rdf:resource = "Wise" />
   </owl:intersectionOf>
   <rdfs:subClassOf>
      <owl:complementOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="commits"/>
            <owl:someValuesFrom rdf:resource="Crime"/>
         </owl:Restriction>
      </owl:complementOf>
   </rdfs:subClassOf>
</owl:Class>
```

In DL syntax: $\text{Honest} \sqcap \text{Wise} \sqsubseteq \neg\exists\text{commits.Crime}$
In NNF: $\text{Honest} \sqcap \text{Wise} \sqsubseteq \forall\text{commits.}(\neg\text{Crime})$

(c) Bonnie does not report Clyde.

```
<owl:Class>
   <owl:oneOf rdf:parseType="Collection">
      <rdf:Description rdf:about="Bonnie"/>
   </owl:oneOf>
   <rdfs:subClassOf>
      <owl:Class>
         <owl:complementOf>
            <owl:Restriction>
               <owl:onProperty rdf:resource="reports"/>
               <owl:someValuesFrom>
                  <owl:Class>
                     <owl:oneOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="Clyde"/>
                     </owl:oneOf>
                  </owl:Class>
               </owl:someValuesFrom>
            </owl:Restriction>
         </owl:complementOf>
      </owl:Class>
   </rdfs:subClassOf>
</owl:Class>
```

In DL syntax: $\{\text{Bonnie}\} \sqsubseteq \neg\exists\text{reports}.\{\text{Clyde}\}$
In OWL 2:

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

In DL syntax: $\neg\text{reports}(\text{Bonnie}, \text{Clyde})$

(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.

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

In DL syntax: $(\geqslant 10\ \text{commits}.\text{Crime})(\text{Clyde})$

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

```
<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">
```

```
                  <rdf:Description rdf:about="Crime"/>
                  <owl:Restriction>
                     <owl:onProperty>
                        <rdf:Description>
                           <owl:inverseOf rdf:resource="commits"/>
                        </rdf:Description>
                     </owl:onProperty>
                     <owl:someValuesFrom>
                         <owl:Class>
                            <owl:oneOf rdf:parseType="Collection">
                               <rdf:Description rdf:about="Clyde"/>
                            </owl:oneOf>
                         </owl:Class>
                     </owl:someValuesFrom>
                  </owl:Restriction>
               </owl:intersectionOf>
            </owl:Class>
         </owl:someValuesFrom>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

In DL Syntax: $\{\text{Bonnie}\} \sqsubseteq \exists\text{commits}.(\text{Crime} \sqcap \exists\text{commits}^{-}.\{\text{Clyde}\}$

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

```
<owl:ObjectProperty rdf:about="suspects">
   <owl:propertyChainAxiom rdf:parseType="Collection">
      <owl:ObjectProperty rdf:resource="suspects"/>
      <owl:ObjectProperty>
         <owl:inverseOf rdf:resource="knows"/>
      </owl:ObjectProperty>
   </owl:propertyChainAxiom>
</owl:ObjectProperty>
```

In DL syntax: $\text{suspects} \circ \text{knows}^{-} \sqsubseteq \text{suspects}$

**Solution** (3.4)**.**
A vegetarian pizza, is a pizza which . . .

(a) . . . does not have an ingredient, which is meat and fish at the same time.

(b) . . . has only toppings, which are either not meat or not fish.

(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):

- $\texttt{CheesePizza} \equiv \texttt{Pizza} \sqcap \exists \texttt{hasTopping.Cheese}$
  Not a subclass of $\texttt{VegetarianPizza}$. Correction:
  $\texttt{CheesePizza} \equiv \texttt{Pizza} \sqcap \exists \texttt{hasTopping.Cheese} \sqcap \forall \texttt{hasTopping.Cheese}$
- $\texttt{PizzaSpinach} \equiv \exists \texttt{hasTopping.Spinach} \sqcap \exists \texttt{hasTopping.Cheese} \sqcap \forall \texttt{hasTopping.}(\texttt{Spinach} \sqcup \texttt{Cheese})$
  Not a subclass of $\texttt{VegetarianPizza}$. Correction: add the axiom
  $\texttt{Spinach} \sqsubseteq \texttt{Gemüse}$
- $\texttt{PizzaCarnivorus} \equiv \texttt{Pizza} \sqcap \forall \texttt{hasTopping.}(\texttt{Meat} \sqcap \texttt{Fish})$
  Not a subclass of $\texttt{VegetarianPizza}$. Correction:
  $\texttt{PizzaCarnivorus} \equiv \texttt{Pizza} \sqcap \forall \texttt{hasTopping.}(\texttt{Meat} \sqcup \texttt{Fish})$
- $\texttt{EmptyPizza} \equiv \texttt{Pizza} \sqcap \neg \exists \texttt{hasTopping.}\top$
  Subclass of $\texttt{VegetarianPizza}$ (no correction needed).

(c) Changed axiom:

$$\texttt{VegetarianPizza} \sqsubseteq \texttt{Pizza} \sqcap \forall \texttt{hasIngredient.}(\neg \texttt{Meat} \sqcap \neg \texttt{Fish})$$

No class would be classified as subclass of $\texttt{VegetarianPizza}$ anymore, e.g., it still holds:

$$\texttt{CheesePizza} \sqsubseteq \texttt{Pizza} \sqcap \forall \texttt{hasIngredient.}(\neg \texttt{Meat} \sqcap \neg \texttt{Fish})$$

which does not imply

$$\texttt{CheesePizza} \sqsubseteq \texttt{VegetarianPizza}$$