Pizza Ontology

Tiago Lubiana Alves São Paulo, 11/11/2020

Exercise 1.2

Question

Find Margherita and see how it is defined as a pizza with only cheese and tomato topping. Look at the definition of VegetarianPizza. Is a Margherita pizza a vegetarian pizza? Why / why not?

Answer

- VegetarianPizza is defined as: `Pizza and (not (hasTopping some SeafoodTopping)) and (not (hasTopping some MeatTopping))`

as neither cheese or tomato toppings are subclasses of either SeafoodTopping or MeatTopping, then Margherita pizza is a vegetarian pizza.

Exercise 1.3

Question

Find hasIngredient. What is the domain and range of this property? What are the subproperties of hasIngredient? What is the inverse property of hasIngredient? What property characteristics does hasIngredient have?

Answer

- Domain:
  - Food
- Ranges:
  - Food
- SubProperties:
  - hasBase
  - hasTopping
- Inverse Property:
  - isIngredientOf

Property characteristics: rdfs:comment "NB Transitive - the ingredients of ingredients are ingredients of the whole"

Exercise 1.4

Question

Classify the ontology by choosing a reasoner and then "classify" in the reasoner menu. In the "Inferred class hierarchy" two classes show up as subclasses of owl:Nothing. Answer the following questions:

- In general, what is the difference between the asserted class hierarchy and the inferred class hierarchy?
- What does it mean for a class to be a subclass of owl:Nothing?
- Explain why these two classes appear as subclasses of owl:Nothing.
- Find Margherita in the inferred class hierarchy and see which classes are inferred as superclasses of Margherita.

Answer A

Asserted class hierarchies are explicitly added by editors. Inferred hierarchies stem from logical entailments derived from the asserted statements and the ontology constraints.

Answer B

According to W3C, the class extension of owl:Nothing is the empty set. Consequently, every OWL class is a subclass of owl:Thing and owl:Nothing is a subclass of every class. A subclass of owl:Nothing means that it is unsatisfiable, as owl:Nothing represents the empty set, and thus its subclasses cannot have any instances.

Answer C

They do not have an assertion of subclass in the asserted hierarchy, they are not stated as subclasses of anything.

Answer D

The HermiT reasoner inferred that Margherita is a subclass of CheesyPizza, VegetarianPizza1 and VegetarianPizza2

Exercise 1.5

Question

Add a new class Grandiosa as a subclass of NamedPizza. Define "Grandiosa" as something which

- hasTopping some HamTopping,
- hasTopping some TomatoTopping and
- hasTopping some CheeseTopping.

Classify the ontology. What superclasses are inferred as superclass of Grandiosa? Explain why.

Answer

The Elk reasoner inferred MeatyPizza and CheesyPizza as superclasses, due to "hasTopping some HamTopping" and "hasTopping some CheeseTopping" respectively. The HermiT reasoner inferred both previous superclasses and also InterestingPizza, as it has 3+ toppings.

Exercise 1.6

Question

State in the ontology that a Grandiosa pizza comes from Norway, and that Norway is different from the other countries already present in the pizza ontology. Apply reasoning and explain the results.

Answer

The reasoner Hermit complained that the Individual Norway was outside the domain of the class country. The Elk reasoner did not show anything new. I changed the domain of Country and the HermiT reasoner did not show anything new.

Family relations in OWL

For each of the modelling exercises below express the exercise text as a set of description logic (DL) axioms.

Exercise 2.2

Question

State that a person has at least one father and one mother.

Answer:

In OWL

```
    _:x1 rdf:type owl:Restriction ;
            owl:onProperty fam:hasMother ;
            owl:minCardinalityQ "1"^^xsd:nonNegativeInteger .

    _:x2 rdf:type owl:Restriction ;
            owl:onProperty fam:hasFather ;
            owl:minCardinalityQ "1"^^xsd:nonNegativeInteger .
```

Exercise 2.3

Question

State that a person can only have one mother and only one father.

Answer:

In OWL

```
    _:x4 rdf:type owl:Restriction ;
            owl:onProperty fam:hasMother ;
            owl:cardinality "1"^^xsd:nonNegativeInteger .

    _:x5 rdf:type owl:Restriction ;
```

```
            owl:onProperty fam:hasFather ;
            owl:cardinality "1"^^xsd:nonNegativeInteger .
```

Exercise 2.4

Question

State that a woman can only have female as gender, and a man can only have male as gender.

Answer:

In OWL

```
fam:Woman rdf:type owl:Class .

_:x7 rdf:type owl:Restriction ;
        owl:onProperty fam:hasGender ;
        owl:hasValue fam:Female .

fam:Woman rdfs:subClassOf _:x7 .

fam:Man rdf:type owl:Class .

_:x17 rdf:type owl:Restriction ;
        owl:onProperty fam:hasGender ;
        owl:hasValue fam:Male .

fam:Man rdfs:subClassOf _:x17 .
```

Exercise 2.5

Question

State that nothing can be both male and female.

Answer:

Note: current gender theory disagrees with such assertion. See https://www.nature.com/articles/s12276-019-0341-0 for reference.

In OWL

` fam:Female owl:disjointWith fam:Male.

`

Exercise 2.6

Question

Define the gender so that there can only be the genders man and woman.

Answer:

Note: current gender theory disagrees with such assertion. See https://www.nature.com/articles/s12276-019-0341-0 for reference.

In OWL

```
fam:Gender rdf:type owl:Class .

_:x8 rdf:type owl:Class ;
owl:oneOf (fam:Male fam:Female)

fam:Gender rdfs:subClassOf _:x8 .
```

Exercise 2.7

Question

Explain what disjointness is. For all pair of classes in the family ontology, add the correct disjoint axioms.

Answer

Disjointness means that there are no instances shared by both classes, i.e there is not any individual that is an instance of ("belongs") to both classes.

In OWL

```
fam:Family owl:disjointWith fam:Gender,
                    fam:Man,
                    fam:Woman,
                    fam:Male,
                    fam:Female.


fam:Gender  owl:disjointWith fam:Family,
                    fam:Man,
                    fam:Woman.



fam:Woman  owl:disjointWith fam:Man.
```

Exercise 2.8

Question

State that a person is either a man or a woman, but not both.

Answer

In OWL

```
foaf:Person rdf:type owl:Class .

_:x9 rdf:type owl:Class ;
        owl:oneOf (fam:Man fam:Woman)

foaf:Person rdfs:subClassOf _:x9 .
```

Exercise 2.9

Question

Explain what inverse properties are. For all the properties that exist in our ontology, add the correct inverse property axioms. You are not supposed to add new properties, only state that a property is the inverse of an other property if they already exist in the ontology.

Answer

Inverse properties are logically linked in a way that subject-object pairs are reversed, i. e. they represent inverse relations. As per the w3c example, " persons own cars, cars are owned by persons. The owl:inverseOf construct can be used to define such an inverse relation between properties"

In OWL

There are no inverse relations for the properties in the family.owl file provided.

Exercise 2.10

Explain what it means for a property to be transitive or symmetric. For all the properties in our ontology, if it is natural, state that they are transitive and/or symmetric.

Answer

Transitive properties "travel" down the hierarchy, and can associate pairs along the chain. More precisely, as per w3c: "When one defines a property P to be a transitive property, this means that if a pair (x,y) is an instance of P, and the pair (y,z) is also instance of P, then we can infer the the pair (x,z) is also an instance of P."

Symmetric properties go both ways, and are symmetric in the subject-object axis. As per W3C: "A symmetric property is a property for which holds that if the pair (x,y) is an instance of P, then the pair (y,x) is also an instance of P."

In OWL

```
fam:hasSibling rdf:type owl:SymmetricProperty .

fam:hasFamilyMember rdf:type owl:SymmetricProperty .

fam:hasSpouse rdf:type owl:SymmetricProperty .
```

Exercise 2.11

Question

Is a subproperty of a transitive property necessarily also transitive? Explain why / why not?

Answer

Not necessarily. A subproperty might be more specific, and this specificity hampers transitivity. For example, a "hasAncestor" property could be transitive, while "hasMother", a subproperty of hasAncestor, would not be.

Exercise 2.12

Question

Is a subproperty of a symmetric property necessarily also symmetric? Explain why / why not?

Answer

Not necessarily. A subproperty might be more specific, and this specificity hampers symmetry. For example, hasSibling is symmetric, but hasBrother is not.

Exercise 2.13

Question

Explain what it means for a property to be inverse functional. For all properties in our ontology, state that they are inverse functional if you believe that is correct.

Answer

An inverse functional property is any property for which the value is unique, allowing us to unambiguously identify the subject. In the words of the W3C: "If a property is declared to be inverse-functional, then the object of a property statement uniquely determines the subject (some individual). More formally, if we state that P is an owl:InverseFunctionalProperty, then this asserts that a value y can only be the value of P for a single instance x, i.e. there cannot be two distinct instances x1 and x2 such that both pairs (x1,y) and (x2,y) are instances of P."

For all the properties in the ontology, defined in the way they currently are, there is at least one culture in which the inverse-functionality assumption would break.