5/02/2016 EX5

martedì 5 gennaio 2021 22:00

(a) Write an OWL ontology that formalizes the domain described at point (a) of Exercise 4.

(b) Add to the above ontology the axioms formalizing the following statements:

1. add a new property `isWrittenBy` and state that it is the inverse of `isWriterOf`;
2. add a new class `WrittenByMultipleAuthors` and state that it corresponds to the class of movies written by at least two writers;
3. add a new class `hasLargeCast` and state that it corresponds to the class of movies played by at least 10 actors;
4. add the new class `allFemaleCast` and state that such a class corresponds to the class consisting of every movie whose writers, directors and actors are all women;
5. `bornIn` and `actsIn` are disjoint properties.

Then, tell whether the resulting OWL ontology is redundant, i.e.: can some of the axioms constituting the ontology be deleted without changing the meaning (that is, the models) of the ontology? if so, identify and list such axioms.

a)

1) Declaration(Class(myns:Person))
   Declaration(Class(myns:Director))
   Declaration(Class(myns:Writer))
   Declaration(Class(myns:Actor))
   Declaration(Class(myns:Country))
   Declaration(Class(myns:Movie))
   Declaration(Class(myns:Comedy))
   Declaration(Class(myns:Drama))
   Declaration(Class(myns:Man))
   Declaration(Class(myns:Woman))
2) subClassOf(myns:Man myns:Person)
   subClassOf(myns:Woman myns:Person)
3) subClassOf(myns:Comedy myns:Movie)
   subClassOf(myns:Drama myns:Movie)
4) Declaration(ObjectProperty(myns:actsIn))
   Declaration(ObjectProperty(myns:bornIn))
   Declaration(ObjectProperty(myns:filmedIn))
   Declaration(ObjectProperty(myns:isDirectorOf))
   Declaration(ObjectProperty(myns:isWriterOf))
5) subClassOf(ObjectSomeValuesFrom(myns:isDirectorOf owl:Thing) myns:Director)
   subClassOf(ObjectSomeValuesFrom(ObjectInverseOf(myns:isDirectorOf) owl:Thing) myns:Movie)
6) subClassOf(ObjectSomeValuesFrom(myns:filmedIn owl:Thing) myns:Movie)
   subClassOf(ObjectSomeValuesFrom(ObjectInverseOf(myns:filmedIn) owl:Thing) myns:Country)
7) subClassOf(ObjectSomeValuesFrom(myns:bornIn owl:Thing) myns:Person)
   subClassOf(ObjectSomeValuesFrom(ObjectInverseOf(myns:bornIn) owl:Thing) myns:Country)

8) subClassOf(ObjectSomeValuesFrom(myns:actsIn owl:Thing) myns:Actor)
   subClassOf(ObjectSomeValuesFrom(ObjectInverseOf(myns:actsIn) owl:Thing) myns:Movie)
9) subClassOf(DataSomeValuesFrom(myns:hasBoxOfficeGross owl:Thing) myns:Movie)
   subClassOf(owl:Thing DataAllValuesFrom(myns:hasBoxOfficeGross xsd:integer))
10) ObjectPropertyAssertion(myns:isDirectorOf myns:Ann myns:XYZ)
    ObjectPropertyAssertion(myns:isWriterOf myns:Ann myns:XYZ)
11) ObjectPropertyAssertion(myns:actsIn myns:Joe myns:ABC)
    ObjectPropertyAssertion(myns:actsIn myns:Paul myns:ABC)
12) ObjectPropertyAssertion(myns:filmedIn myns:ABC myns:France)
13) ClassAssertion(myns:Woman myns:Ann)
14) ClassAssertion(myns:Man myns:Paul)

b)

1) InverseObjectProperty(myns:isWrittenBy myns:isWriterOf)
2) Declaration(Class(myns:WrittenByMultipleAuthors))
   EquivalentClasses(myns:WrittenByMultipleAuthors ObjectIntersectionOf(myns: Movie ObjectMinCardinality(2 myns:isWrittenBy myns:Writers))
3) Declaration(Class(myns:hasLargeCast))
   EquivalentClasses(myns:hasLargeCast ObjectIntersectionOf(myns:Movie ObjectMinCardinality(10 ObjectInverseOf(myns:actsIn) myns:Actor))
4) Declaration(Class(myns:AllFemaleCast))
   EquivlentClasses(myns:AllFemaleCast ObjectIntersectionOf(myns:Movie ObjectAllValuesFrom(myns:isWrittenBy myns:Woman) ObjectAllValuesFrom(ObjectInverseOf(myns:isDirectorOf) myns:Woman) ObjectAllValuesFrom(ObjectInverseOf(myns:actsIn) myns:Woman)))
5) DisjointObjectProperties(myns:bornIn myns:actsIn)