23/01/2019 EX3

mercoledì 6 gennaio 2021 11:29

Exercise 3

We want to formalize knowledge about the domain of movies. In particular, we want to formalize the following statements:

1. every director is a person;
2. every actor is a person;
3. the property "worked in" has domain person and range movie;
4. the property "acted in" has domain actor and range movie;
5. the property "directed by" has domain movie and range director;
6. the property "acted in" is a subproperty of "worked in";
7. if a director was born in a European country, she/he is a European director;
8. an actor-director is a director who is also an actor;
9. every movie directed by an actor-director is a special movie;
10. every movie directed by a European director is a European movie.

(a) Choose the most appropriate knowledge representation language for expressing the above knowledge among the following ones: $\mathcal{ALC}$, Datalog, Datalog with constraints, ASP, OWL, $DL\text{-}Lite_R$, $\mathcal{EL}$, RL, RDFS, motivating your choice;

(b) express the above knowledge in the formalism chosen at the previous point.

a)

1) All languages admitt it
2) All languages admitt it
3) Not possible in EL because it does not have inverse role
4) Same as previous
5) Same as previous
6) Not possible in ALC because it does not have subroles, Not possible in EL for the same reason
7) Not possible in DL-Lite because we don't have qualified existential restriction and conjunction, Not possible in RDFS for the same reason
8) No possible in Datalog because we cannot have conjunction of heads, Not possible in Datalog with constraints for the same reason, Not possible in RL because can admitt only atomic concept in the right side, Not possible in ASP because we cannot have conjunction of head, Not in DL-Lite because we don't have conjunction
9) Not possible in DL-Lite because we cannot have qualified existential restriction and conjuction, same for RDFS.
10) Same as previous

b)

The only language is OWL

1) subClassOf (myns:Director myns:Person)
2) subClassOf (myns:Actor myns:Person)
3) subClassOf (ObjectSomeValues(myns:workedIn owl:Thing) myns:Person)
   subClassOf (ObjectSomeValues(ObjectInverseOf(myns:workedIn) owl:Thing) myns:Movie)
4) subClassOf (ObjectSomeValues(myns:actedIn owl:Thing) myns:Actor)
   subClassOf (ObjectSomeValues(ObjectInverseOf(myns:actedIn) owl:Thing) myns:Movie)
5) subClassOf (ObjectSomeValues(myns:directedBy owl:Thing) myns:Movie)
   subClassOf (ObjectSomeValues(ObjectInverseOf(myns:directedBy) owl:Thing) myns:Director)
6) subObjectPropertyOf (myns:actedIn myns:workedIn)
7) subClassOf(ObjectIntersectOf(myns:Director ObjectSomeValuesFrom(myns:bornIn

myns:EuropeanCountry)) myns:EuropenDirector)

8) EquivalentClass(myns:Actor-Director ObjectIntersectOf(myns:Director myns:Actor))
9) subClassOf(ObjectIntersectOf(myns:Movie ObjectSomeValuesFrom(myns:directedBy myns:Actor-Director)) myns:SpecialMovie)
10) subClassOf(ObjectIntersectOf(myns:Movie ObjectSomeValuesFrom(myns:directedBy myns:EuropeanDirector)) myns:EuropeanMovie)