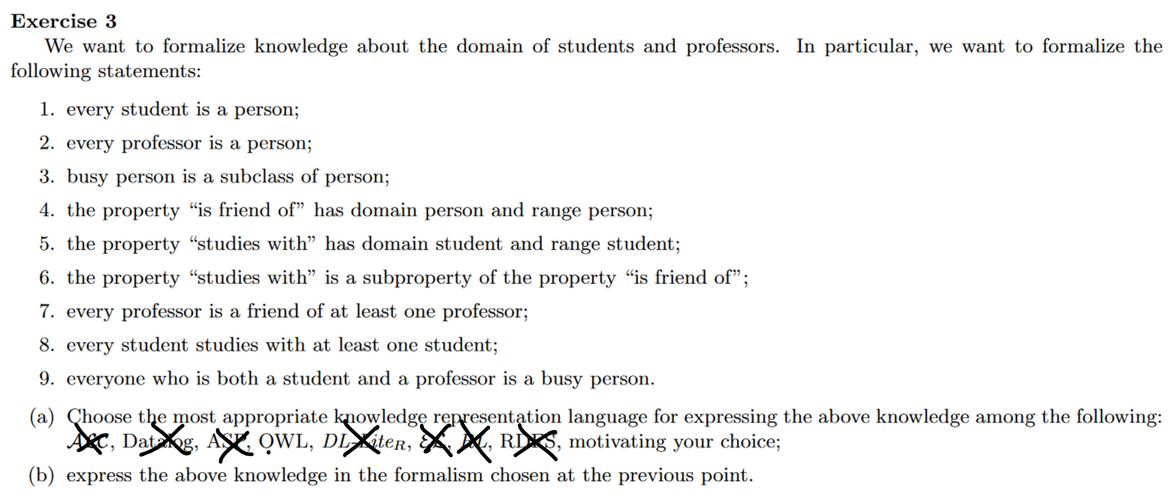
14/06/2017 EX3

lunedì 4 gennaio 2021 12:26



Exercise 3

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

1. every student is a person;
2. every professor is a person;
3. busy person is a subclass of person;
4. the property "is friend of" has domain person and range person;
5. the property "studies with" has domain student and range student;
6. the property "studies with" is a subproperty of the property "is friend of";
7. every professor is a friend of at least one professor;
8. every student studies with at least one student;
9. everyone who is both a student and a professor is a busy person.

(a) Choose the most appropriate knowledge representation language for expressing the above knowledge among the following: ~~$\mathcal{ALC}$~~, ~~Datalog~~, ~~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) All languages admitt it
4) Not admitted in EL because it cannot have inverse role
5) Not admitted in EL because it cannot have inverse role
6) Not admitted in EL because it cannot have subroles and also in ALC subroles are not present
7) Not possible in RDFS because it cannot admitt qualified existential restriction, Not possible in DL-LiteR because we cannot have qualified existential restriction, No in RL because we can have only atomic concept in the right side of the formula, No possible in Datalog and in ASP
8) Not possible in RDFS because it cannot admitt qualified existential restriction, Not possible in DL-LiteR because we cannot have qualified existential restriction, No in RL because we can have only atomic concept in the right side of the formula, No possible in Datalog and in ASP
9) Not possible in DL-Lite because it cannot admitt conjunction, No possible in RDFS because of conjunction

b)

We can express the knowledge in OWL

1) subClassOf(myns:Student myns:Person)
2) subClassOf(myns:Professor myns:Person)
3) subClassOf(myns:BusyPerson myns:Person)
4) subClassOf(ObjectSomeValuesFrom(myns:isFriendOf owl:Thing) myns:Person)
   subClassOf(ObjectSomeValuesFrom(ObjectInverse(myns:isFriendOf owl:Thing)) myns:Movie)
5) subClassOf(ObjectSomeValuesFrom(myns:studiesWith owl:Thing) myns:Student)
   subClassOf(ObjectSomeValuesFrom(ObjectInverse(myns:studiesWith owl:Thing)) myns:Student)
6) subObjectPropertyOf(myns:studiesWith myns:isFriendOf)
7) subClassOf(myns:Professor ObjectSomeValuesFrom(myns:isFriendOf myns:Professor)
8) subClassOf(myns:Student ObjectSomeValuesFrom(myns:studiesWith myns:Student)
9) subClassOf(ObjectIntersectionOf(myns:Student myns:Professor) myns:BusyPerson)