

Advanced Database Systems, solutions of second intermediate test – 31 May 2017 – V1.2

Please feel free to answer this test in English, Italian, or any mixture

1. Let \$doc be bound to an XML document with the following schema:

```
movies
  movie*
    @idmovie
    title
    year
    director (@idperson)
    actor* (@idperson)
persons
  persona*
    @idperson
    name
    birthyear
cinemas
  cinema*
    @idcinema
    name
    town
    screen*
      @idscreen
      nameOfScreen
      screening*
        @idmovie
        date
        starttime
        price
```

- a. Write a query that returns, for each date, the list of all movies that have been screened in that date, listing, for each date and movie, the number of cinemas that project that movie, the number of screenings, and the list of all of the screenings (for the movie and for the date), with the following format

```
date
  movie*
    title
    numberOfCinemas
    numberOfScreenings
    screening*
      nameOfCinema
      starttime
```

```
for $d in fn:distinct-values($doc//date)
return <date>
```

```

{ $d,
  for $m in fn:distinct-values($doc//screening[date=$d]/@idmovie
  let $ss := $doc/screening[@idmovie=$m and date=$d]
  <movie>
    { $doc//movie[@idmovie=$m]/title,
      <numberOfCinemas> fn:count( $ss/../../..) </numberOfCinemas>
      <numberOfScreenings > fn:count( $ss) </numberOfScreenings >,
      for $s in $ss
      return <screening> <nameOfCinema> $s/../../name/text() </nameOfCinema>
        { $s/starttime }
      </screening>
    }
  </movie> }
</date>

```

- b. Write a query that returns the list of all directors that only directed movies with no actor, or where the only actor is the director herself/himself

```

for $d in $doc //director
where each $m in $doc/movie[director/@idperson = $d/idperson]
  satisfies each $a in $m/actor
    satisfies $a/@idperson = $d/idperson
return $d

```

2. Consider an RDF graph with classes Person and Movie, and with the following declarations of classes and predicates

$$\begin{aligned}
&\text{HasActor} \subseteq \text{Movie} \times \text{Actor} \\
&\text{HasDirector} \subseteq \text{Movie} \times \text{Director} \\
&\text{Directed} \subseteq \text{Director} \times \text{Actor} \\
&\text{Actor, Director, SelfDirector} \subseteq \text{Person} \\
&\text{ManyDirectorsMovie, ActorLessOne, ActorLessTwo} \subseteq \text{Movie}
\end{aligned}$$

Formalize the following statements in OWL, paying extreme attention to the direction of the implication:

- a. If X has been the director of a movie where Y was among the actors, then X Directed Y

SubObjectPropertyOf(ObjectPropertyChain (InverseOf (HasDirector) HasActor) Directed)

- b. X is a SelfDirector if and only if X Directed X

EquivalentClasses(ObjectHasSelf (Directed) SelfDirector)

c. Is (b) equivalent to say that X is a SelfDirector if and only if X has been the director of a movie where X was among the actors?

No – the statement of (c) makes SelfDirector equivalent to being the director of a movie where one acts, while (b) makes that equivalent to X Directed X which is only implied by being the director of a movie where one acts, but is not equivalent to that condition. In other terms, the two are not equivalent since (a) only gives a lower bound for the Directed relation, but does not give an iff condition. Hence, in a model, every individual X may satisfy X Directed X without contradicting (a).

d. A Movie is a ‘ManyDirectorsMovie’ if and only if this movie has at least two directors

```
EquivalentClasses( ManyDirectorsMovie
                   ObjectMinCardinality(2 HasDirector)
                   )
```

e. If a movie has no actor then it is an ActorLessOne movie

```
SubClassOf ( IntersectionOf ( Movie ObjectMaxCardinality(0 HasActor) )
            ActorLessOne )
```

f. A movie is ActorLessTwo if and only if it has no Actor

```
SubClassOf ( IntersectionOf ( Movie ObjectMaxCardinality(0 HasActor) )
            ActorLessOne )
```

(*) (Literally, the statement (d) starts with ‘A Movie is....’, and hence does not, literally, say anything about an individual which is not a movie. In our formalization, we decided to interpret it as: “Something is a ManyDirectorsMovie iff it is a movie and has at least two directors”. We did the same for (f). The statement (e) does not have this kind of ambiguity.)

Consider now a knowledge base that consists of the above declarations, of the formalization of a-f, and of a set of RDF triples talking about those predicates. Remembering the open world assumption and considering the directions of the implications, answer the following questions.

g. In this knowledge base, may it be possible, or is it possible, to deduce that ActorLessOne is equivalent to ActorLessTwo, or that one is a subclass of the other?

From (e) and (f) we may deduce that ActorLessTwo is a subclass of ActorLessOne. The other inclusion cannot be deduced since we have no upper bound for ActorLessOne.

h. In such a knowledge base, may it be possible to prove that a movie is an ActorLessOne movie?

No, because of the Open World Assumption

i. In such a knowledge base, may it be possible to prove that a movie is not an ActorLessOne movie?

No, because of the direction of the implication: we have no upper bound for ActorLessOne, every element may potentially belong to that class

j. In such a knowledge base, may it be possible to prove that a movie is an ActorLessTwo movie?

No, because of the Open World Assumption

k. In such a knowledge base, may it be possible to prove that a movie is not an ActorLessTwo movie?

Yes. If the RDF graph contains a triple “M HasActor A”, then M does not belong to ObjectMinCardinality (0 HasActor), and we can deduce that it does not belong to ActorLessTwo since the two classes are equivalent.