Resolução do teste 1 de md de 2019

1

a)

```
MATCH (p:Person)-[:ACTED_IN|PRODUCED]-> (m:Movie)
WHERE p.name = 'Brad Pitt'
return m
union
MATCH (p:Person)-[:ACTED_IN|PRODUCED]-> (m:Movie)
Where p.name = 'Angelina Jolie'
return m

MATCH (p:Person)-[:ACTED_IN|PRODUCED]-> (m:Movie)
Where p.name = 'Angelina Jolie' or p.name = 'Brad Pitt'
return m
```

b)

```
Match (p1:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(p2:Person)
with p1,p2, count(m) as cnt_Movies
where cnt_Movies >= 4 and p1.name != p2.name
return p1.name ,p2.name
```

c)

```
MATCH (m:Movie)
OPTIONAL MATCH (a:Person)-[:ACTED_IN]->(m)
WITH m, collect(a.name) as actors
OPTIONAL MATCH (p:Person)-[:PRODUCED]->(m)
WITH m, actors, collect(p.name) as producers
RETURN m, actors, producers
```

2

a)

//desenhar grafo

b)

//desenhar grafos. A primeira justificação estará completa as restantes devem seguir essa

1

No,

since there is no subgraph of G (the graph in 2 a) that is an instance of the graph G1 (the graph in this exercise) so throught simple entailment G does not entail G1

The only node with a has-wife relationship is :c and it does not have any :is-mother-of relationship

2

no since neither _:b4 nor _:b3 have loops to themselves and are the only who have is brother relationships

3

yes

_:x mapped into _:b1

both _:y and _:z into <http://foo.ex/m> or <http://foo.ex/c>

and _:w mapped to either Manuel if _:y and _:z have the first mapping and "charles" if they have the second

4

no

no node is related to another node throught a is-mother-of relationships that is related to another through another is-mother-of.

c)

we need to infer the following 3 tuples (with URIs abbreviated)

_:x rdf:type fam:Person

_:x fam:is-parent-of _:y

_:y fam:is-parent-of _:z

_:z rdf:type fam:Male

the original triples are

13 _:b1 <http://www.family.org/onto#is-mother-of> <http://foo.ex/m>

14 _:b1 <http://www.family.org/onto#is-mother-of> <http://foo.ex/c>

15 <http://foo.ex/c> <http://www.w3.orf/2000/01/rdf-schema#label>"Charles"

16<http://foo.ex/c><http://www.family.org/onto#has-wife>_:b2

17 _:b2<http://www.family.org/onto#has-husband><http://foo.ex/c>

18 <http://foo.ex/c><http://www.family.org/onto#is-father-of> <http://foo.ex/r>

19 _:b2 <http://www.family.org/onto#is-mother-of> <http://foo.ex/r>

20 <http://foo.ex/r> <httop://www.w3.org/1999/02/22-rdf-syntax.ns#type><http://www.family.org/onto#female>

21 <http://foo.ex/m><http://www.w3.orf/2000/01/rdf-schema#label>"Manuel"

22 <http://foo.ex/m> <http://www.family.org/onto#is-father-of> _:b4

23 <http://foo.ex/m> <http://www.family.org/onto#is-father-of> _:b3

24 _:b3 <http://www.family.org/onto#brother-of> _:b4

25 _:b4 <http://www.family.org/onto#brother-of> _:b3

26 _:x fam:is-mother-of :m se2 (13) alocando _:x a _:b1

27 _:x rdf:type fam:Female rdfs2 (26,1)

28 _:x rdf:type fam:Person rdfs9 (27,12)

29 _:x fam:is-parent-of :m rdfs7 (26,7)

30 _:x fam:is-parent-of _:y se2 (29) alocando _:y a :m

31 :m fam:is-parent-of _:b4 rdfs7 (22,8)

32 _:y fam:is-parent-of _:b4 se2(31)

33 _:y fam:is-parent-of _:z se1(32) alocando _:z a _:b4

34 _:b4 rdf:type fam:Male rdfs2(25,4)

35 _:z rdf:type fam:Male se2(34)

3

a)

A = Bgp(?x?fam:is-father-of?y)

x,y

:c,:r

:m,_:i

:m,_:f

B = Bgp(?x?fam:is-mother-of?y)

x,y

_:b1,:c

_:b1,:m

_:b2,:r

C1 = Union(A,B)

x,y

:c,:r

:m,_:i

:m,_:f

_:b1,:c

_:b1,:m

_:b2,:r

C2 = Union(A,B)

x,z

:c,:r

:m,_:i

:m,_:f

_:b1,:c

_:b1,:m

_:b2,:r

D = InnerJoin(C1,C2)

x,y,z

:c,:r,:r

:m,_:i,_:i

:m,_:f,_:f

:m,_:i,_:f

:m,_:f,_:i

_:b1,:c,:m

_:b1,:m,:c

_:b1,:c,:c

_:b1,:m,:m

_:b2,:r,:r

E = Filter(D, y =z)

x,y,z

:m,:i,:f

:m,:f,:i

_:b1,:c,:m

_:b1,:m,:c

F = Bgp(?x?rdfs:label?n)

x,n

:c,"Charles"

:m,"Manuel"

Result = Leftjoin(E,F)

x,y,z,n

:m,:i,:f,"Manuel"

:m,:f,:i,"Manuel"

_:b1,:c,:m,null

_:b1,:m,:c,null

b)

select ?x ?sx ?y ?sy

WHERE{

{?z fam:is-parent ?x}

{?z fam:is-parent ?y}

Filter(?x != ?y)

} OPTIONAL

{?x a ?sx}

Filter(?sx = fam:Male || ?sx = fam:Female)

OPTIONAL

{?y a ?sy}

Filter(?sy = fam:Male || ?sy = fam:Female)

}.

c)

select ?x

Where{

```
?x a fam:Person.
{SELECT ?x Count(?y) as ?Count
 where{
        ?x fam:is-parent ?y.)}.}
Filter(?Count<2).}
```

d)

```
select ?x,?y
WHERE{
?x (^fam:is-parent-of)+ / ( fam:is-brother-of | fam:is-sister-of) / fam:is-parent-of+ ?y
.}
```