Exercises on SPARQL

1. SPARQL queries

An RDF graph has a schema that defines the classes s:Person, s:Woman, s:Man, s:Soldier and the property s:hasParent.

Write SPARQL queries to answer the following questions on this graph

a) Write SPARQL queries to answer the following questions on this graph

1. find the persons having at least one ancestor (from the grandparent generation) who was a soldier
2. find the men whose ancestors (from the grandparent generation) were all soldiers (no ancestor is not a soldier)
3. find the women with no male ancestor (from the parent generation) being a soldier
4. find the women having at least one ancestor who was a soldier or a clergyman

```
PREFIX s: <http://cui.unige.ch/>
```

Q1.

```
 select ?p
 where { ?p a s:Person. ?p s:hasParent ?pp. ?pp s:hasParent+ ?a. ?a a
s:Soldier}
```

Variant:

```
 select ?p
 where {?p a s:Person . ?p s:hasParent / s:hasParent+ ?a. ?a a s:Soldier}
```

Other possible variant :

```
 select ?p
 where {?p a s:Person . ?p s:hasParent+ / s:hasParent+ ?a. ?a a s:Soldier}
```

Not OK

```
 select ?p
 where {?p a s:Person . ?p (s:hasParent / s:hasParent)+ ?a. ?a a s:Soldier}
```

Other possible variant

```
 select ?p
 where {?p a s:Person . ?p s:hasParent / s:hasParent+ ?a. filter exists {?a   a
s:Soldier}
}
```

Q2.

```
select distinct ?p
where {?p a s:Man. ?p s:hasParent ?pp. ?pp s:hasParent+ ?aa.
minus {filter not exists {?p a s:Man. ?p s:hasParent ?pp. ?pp s:hasParent+
?a.?a a s:Soldier}}
}
```

Q3.

Not OK:

```
select distinct ?p
where {?p a s:Woman.?p s:hasParent+ ?a.
minus {?p a s:Woman. ?p s:hasParent+ ?a. ?a a s:Man. ?a a s:Soldier}
  }
```

OK:

```
select distinct ?p
where {?p a s:Woman. ?p s:hasParent+ ?b.
minus {?p a s:Woman. ?p s:hasParent+ ?a. ?a a s:Man. ?a a s:Soldier}
  }
```

Not OK:

```
select distinct ?p
where {?p a s:Woman. ?p s:hasParent+ ?a.
filter not exists {?a a s:Man. ?a a s:Soldier}
  }
```

Q4.

```
select distinct ?p
where {
    {?p a s:Woman. ?p s:hasParent+ ?a. ?a a s:Soldier}
union
    {?p a s:Woman. ?p s:hasParent+ ?a. ?a a s:Clergyman}
}
```

Variant:

```
   select distinct ?p  where {
    {?p a s:Woman. ?p s:hasParent+ ?a.}
    {{?a a s:Soldier} union {?a a s:Clergyman}}
}
```

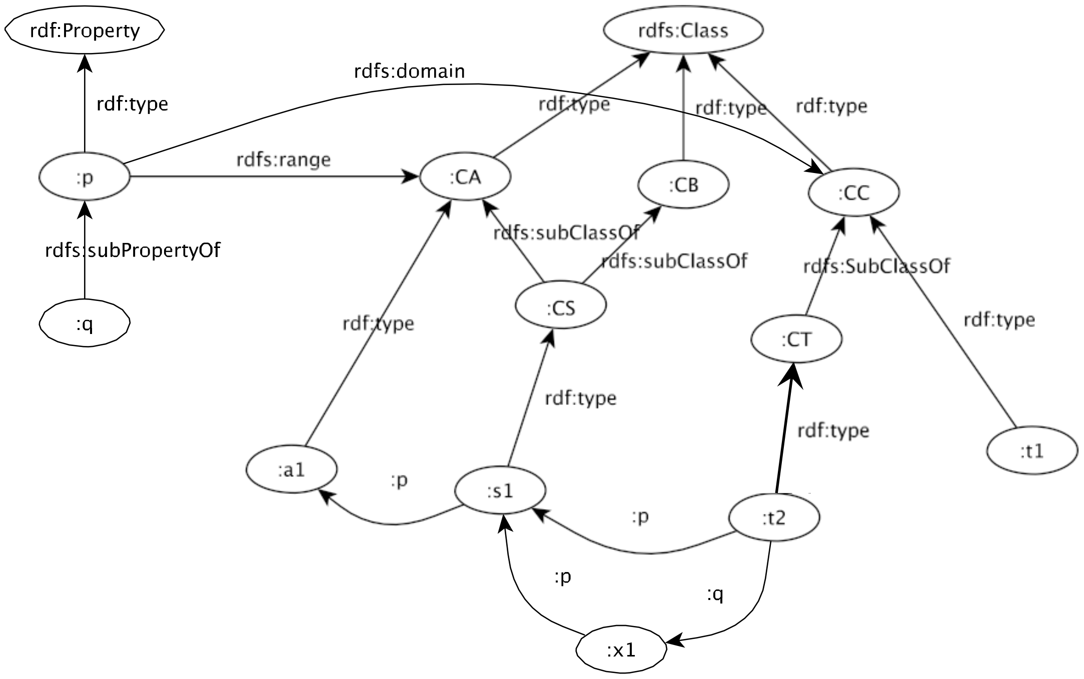
b) Write SPARQL queries to check if the property s:hasParent, considered as a relation, has no cycle.

Remark: To check if a condition holds find the entities that do not satisfy it

```
select ?a
  where {?a s:hasParent+ ?a}
If no answer -> no cycle
```

2. SPARQL queries with/without entailment

Consider the following graph RDF(S).



What will be the result of executing the following queries on
this graph a) without RDFS entailment
b) with RDFS entailment

1. select ?s where {?s :p ?o}
   :s1, :t2, :x1
   with RDFS entailment => nothing more
2. select ?w where {?w a :CA}
   :a1
   with RDFS entailment => adds :s1, :x1

3. select ?x where {?y :p ?x. filter not exists {?y a :CC}}

:a1, :s1, :x1

with RDFS entailment => result is empty

4. select ?x where {?x :p ?y. ?y a :CA}

:s1

with RDFS entailment => :t2. :x1.

3. SPARQL rewriting

A SPARQL endpoint *S* has an RDF schema that defines the classes *s:Person* and *s:Farmer* and the property *s:hasAncestor*.
For this endpoint a query to find all the ancestors of a person that are/were farmers can be expressed as:

```
Q: select ?a
   where {?a a s:Person. ?p a s:Person.
          ?p s:hasAncestor ?a. ?a a s:Farmer}
```

In another endpoint *T*, the schema has the classes: *t:LivingPerson*, *t:DeadPerson*, *t:Cultivator*, and the properties *t:hasFather* and *t:hasMother*.

Rewrite Q in order to obtain an (almost) equivalent query for the endpoint *T*.

```
Q: select ?a
   where {
     {?a a t:LivingPerson} union {?a a t:DeadPerson} .
     {?p a t:LivingPerson} union {?p a t:DeadPerson} .
     ?p (t:hasFather|t:hasMother)+ ?a .
     ?a a t:Cultivator
   }
```