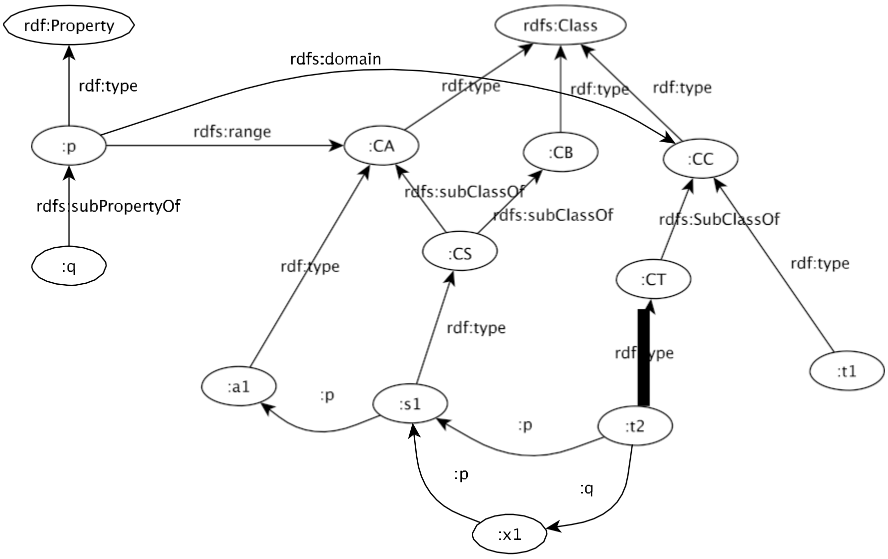
WRITTEN EXAM: SOME SAMPLE QUESTIONS

Possible answers in blue

Q: SPARQL

Consider the following graph RDF/S // *On considère le graphe RDF suivant*



1. What will be the result of executing the following queries on this graph

   a) without RDFS entailment

   b) with RDFS entailment

Quel sera le résultat de l'exécution des requêtes suivantes sur ce graphe

a) sans inférence RDFS

b) avec inférence RDFS

```
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.
```

2. Write SPARQL queries that correspond to the following questions:

   1. find all the classes *C* that have at least one member that is connected through :p to a member of a subclass of *C*

      trouver toutes les classes C dont au moins un membre est connecté par la propriété :p à un membre d'une sous-classe de C

      select ?c where ?c a rdfs:Class . ?x a ?c . ?x :p ?y . ?y a ?s . ?s rdfs:subClassOf ?c

   2. find all the members of the class :CC that are connected to at most one node through property :p.

      trouver tous les membres de la classe :CC qui sont connecté à au plus un noeud par la propriété :p.

      select ?c {?c a :CC. {filter not exists{?c :p ?n}} union {?c :p ?d . filter not exists {?c :p ?e. filter(?e != ?d)}}}

Q: 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*.

```
select distinct ?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
      }
```

Q: DL modeling

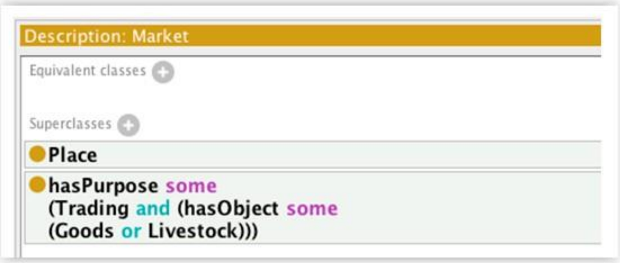
An OWL ontology contains the following class hierarchy, properties and individuals:
Classe hierarhy:

- Place
  - Castle
    - HauntedCastle
  - BedAndBreakfast
  - GuestHouse
  - PerchedHut
- Entity
  - Ghost
  - Tree
- Purpose
  - Providing
- Object
  - Accomodation
  - Breakfast
- Country

Properties: locatedIn
frequentedBy
hasPurpose
hasObject

Individuals:
Scotland (instance of Country)

Hint: Here is the description of a Market with a similar vocabulary:



Description: Market

Equivalent classes

Superclasses

Place

hasPurpose some
(Trading and (hasObject some
(Goods or Livestock)))

Write axioms (in DL or Manchester syntax) to express the following elements of domain knowledge:

1. A haunted castle is a castle frequented by ghosts
   Un château hanté est un château fréquenté par des fantômes

   HC ≡ Castle and frequentedBy min 2 Ghost

2. Every castle located in Scotland is frequented by at least 2 ghosts
   Tout château situé en Ecosse est fréquenté par au moins 2 fantômes

   Castle and locatedIn value Scotland ⊑ frequentedBy min 2 Ghost

3. A bed and breakfast is a place whose purpose is providing accomodation and breakfast and which is located in a guest house
   Un bed and breakfast est un endroit qui a pour but de fournir hébergement et petit déjeuner et qui est situé dans une maison d'hôtes

   BB ≡ Place
   and hasPurpose some (Providing and hasObject some Accomodation)
   and hasPurpose some (Providing and hasObject some Breakfast)
   and locatedIn some GuestHouse

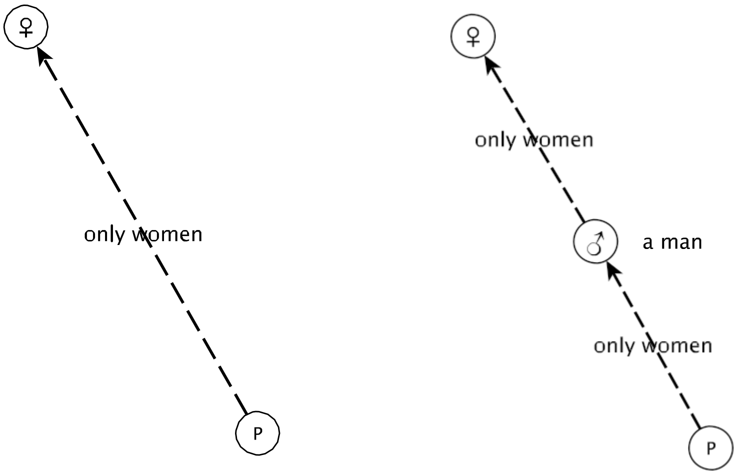
4. A perched hut is a place located in a tree whose purpose is providing accomodation
   Une cabane perchée est un endroit situé dans un arbre qui a pour but de fournir un hébergement

   PH ≡ Place and locatedIn some Tree
   and hasPurpose some (Providing and hasObject some Accomodation)

Q: SWRL modeling

Define a vocabulary (classes and properties) and SWRL rules for representing the following definitions:

1. The uncle of a person is a brother of the mother or a brother of the father of this person
   L'oncle d'une personne est un frère de la mère ou un frère du père de cette personne
2. The women-only-ancestors of a person are his/her mother, mother of the mother, mother of the mother of the mother, etc. (see figure, left)
   Les ancêtres-femmes-seulement d'une personne sont sa mère, la mère de sa mère, la mère de la mère de sa mère, etc. (à gauche sur la figure)
3. The almost-women-only-ancestors of a person are his/her ancestors connected to him/her through a chain containing only women except for one man (see figure right)
   Les ancêtres presque-seulement-femmes d'une personne sont ses ancêtres femmes liées à elle par une chaine ne comprenant que des femmes, à l'exception d'un homme (à droite sur la figure).



```
mother(?X, ?M), brother(?M, ?Y) -> uncle(?X, ?Y)
father(?X, ?F), brother(?F, ?Y) -> uncle(?X, ?Y)

mother(?X, ?M) -> woAncestor(?X, ?M)
mother(?X, ?M), woAncestor(?M, ?A) -> woAncestor(?X, ?A)

woAncestor(?X, ?W), father(?W, ?F), woAncestor(?F, ?Y) -> awoAncestor(?X, ?Y)
father(?X, ?F), woAncestor(?F, ?Y) -> awoAncestor(?X, ?Y)
woAncestor(?X, ?F), father(?F, ?Y) -> awoAncestor(?X, ?Y)
```

Q: SWRL to DL

An ontology contains the following SWRL rules:
Une ontologie contient les règles SWRL suivantes:

```
p(?X, ?Y) -> q(?Y, ?X)
q(?X, ?Y) -> p(?Y, ?X)
```

```
p isInverseOf q
```

```
Car(?X), hasMaker(?X, ?M), inCountry(?M, Italy) -> ItalianCar(?X) .
```

```
Car and hasMaker some (inCountry value Italy) subClassOf ItalianCar
```

```
Person(?X), hasChild(?X, ?Y), Person(?Y), hasChild(?X, ?Z),
                                                             Person(?Z)
, differentFrom(?Y, ?Z) -> PW2C(?X) .
```

```
Person and hasChild min 2 Person subClassOf PW2C
```

Your goal is to replace these rules with equivalent OWL axioms (that produce the same inferences). These axioms can be of the form <expression> subClassOf <expression>, <expression> subPropertyOf <expression>, <property> isInverseOf <property>, <property> isTransitive, etc.
Votre but est de remplacer ces règles par des axiomes OWL équivalents (qui produisent les mêmes inférences). Ces axiomes peuvent être de la forme <expression> subClassOf <expression>, <expression> subPropertyOf <expression>, <property> isInverseOf <property>, <property> isTransitive, etc.

Example

```
Car(?X), driver(?X, ?D) -> Person(?D) .
```

can be replaced by // *peut être remplacé par*

```
Car subClassOf driver only Person
```