Day 04: questions from the course on RDFS.

Q4.1 Choose among the following assertions one or more you consider to be true:

1. an ontology is necessarily formalized in first-order logic
2. an ontology may allow inferences on data that uses it
3. conceptual graphs can represent an ontology
4. a shared ontology promotes interoperability
5. description logics can represent an ontology (and biggest)

Answer

2, 3, 4, 5

Q4.2 RDFS contains primitives to (several answers possible)...

- describe classes of resources
- describe formulas of calculation for values of properties
- describe types of properties of resources
- document definitions in natural language
- sign and authenticate the authors of the definitions of classes and properties

Answer

1 3(subPropertyOf) 4(label, comment)

Q4.3. What is defined and derived from these definitions?

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@prefix : <http://inria.fr/devices#>
 :Phone rdfs:subClassOf  :Device .
 :Computer rdfs:subClassOf  :Device .
 :Smartphone rdfs:subClassOf :Computer .
 :Smartphone rdfs:subClassOf :Phone .
```

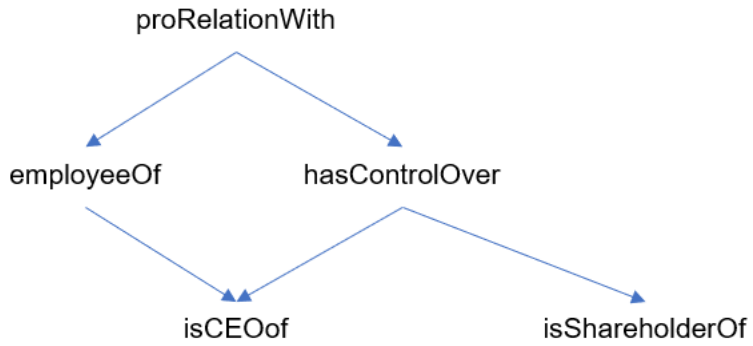
Answer

Phone and computer are a subclass of device
smartphone is the subclass of phone and computer
therefore we know that the smartphone is also a subclass of device

Q4.4. What is defined and derived from these definitions?

```
@prefix rdfs:  < http://www.w3.org/2000/01/rdf-schema# >
@prefix : <http://inria.fr/member#>
:employeeOf rdfs\:subPropertyOf  :proRelationWith .
:hasControlOver rdfs:subPropertyOf :proRelationWith .
:isShareholderOf rdfs:subPropertyOf :hasControlOver .
:isCEOof rdfs:subPropertyOf :employeeOf, :hasControlOver .
```

Answer

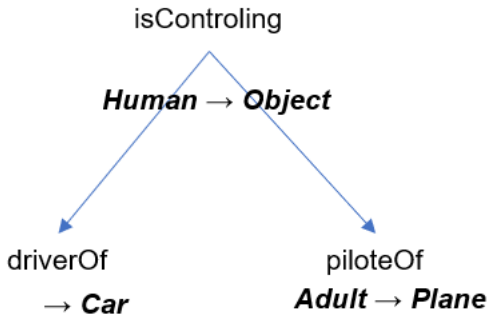


Q4.5. What can be said about the types of the resources that will be linked by the properties defined below?

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
@prefix : <http://inria.fr/humans#>
:driverOf  rdfs:subPropertyOf  :isControling .
:piloteOf  rdfs:subPropertyOf  :isControling .
:isControling  rdfs:domain  :Human ;  rdfs:range :Object .
:driverOf rdfs:range  :Car .
:piloteOf rdfs:domain  :Adult ; rdfs:range  :Plane .
```

Answer



so if we say Fabien pilote Airbus
then we can know that Fabien is a human and Adult
and Airbus is a object and a plane

Q4.6. What could we add to this schema (several answers are possible)?

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@base  <http://inria.fr/2005/humans.rdfs>
<p1>  a rdf:Property ;  rdfs:label  "age"@fr .
<c1>  a rdfs:Class; rdfs:comment "un être humain"@fr .
```

1. <p1> rdfs:label "prénom"@fr .
2. <c1> rdfs:comment "a human being"@fr .
3. <c1> rdfs:label "personne"@fr .
4. <p1> rdfs:label "age"@en .
5. <c1> rdfs:label "woman"@en .
6. <c1> rdfs:label "persona"@es .

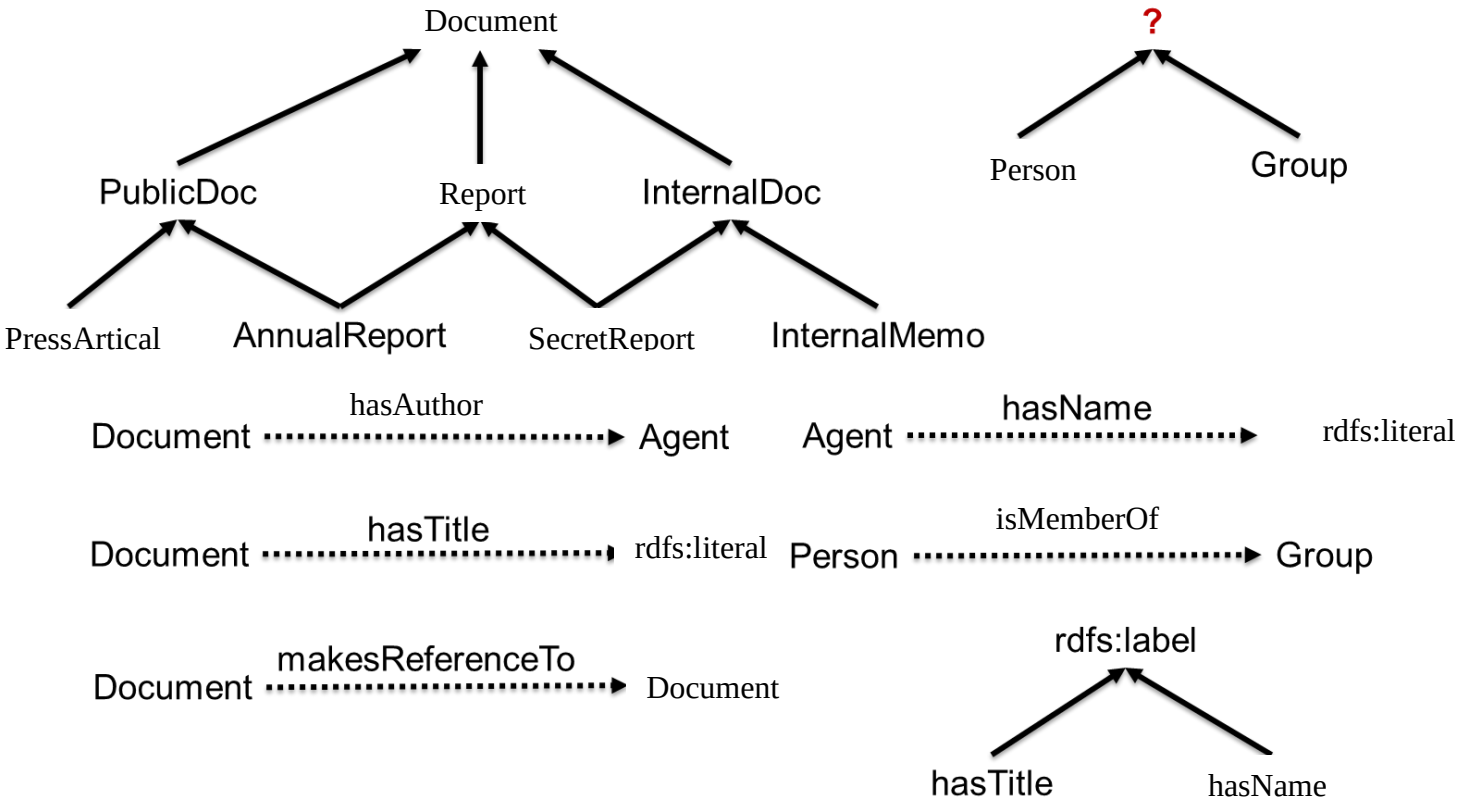
PublicDoc

Answer 3 4 6

Q4.7.

(a) Fill the blanks with: Document, PublicDoc, PressArticle, Report, AnnualReport, InternalDoc, SecretReport, InternalMemo, Agent, Person, Group, hasTitle, hasAuthor, makesReferenceTo, hasName, isMemberOf + rdf / rdfs primitives.
(b) Write it in RDFS and validate the RDF.

Agent


Document
PublicDoc
Report
InternalDoc
PressArtical
AnnualReport
SecretReport
InternalMemo
?
Person
Group
Document
hasAuthor
Agent
Agent
hasName
rdfs:literal
Document
hasTitle
rdfs:literal
Person
isMemberOf
Group
Document
makesReferenceTo
Document
rdfs:label
hasTitle
hasName

Day 04: Answers to the practical session on RDFS.

Software requirements

- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The RDF online translator: http://rdf-translator.appspot.com/
- The SPARQL Corese engine: http://wimmics.inria.fr/corese

Create your own schema Family.rdfs

- Write the the RDF schema that you used in the description of Jen in a RDF/XML (or in turtle and then translate it) and save the RDF/XML in a file called "Family.rdfs". Of course, this assumes that the URIs for the classes and properties declared/used must match in both files. You mays have to update the files Jen.rdf and Jen.ttl to use your ontology.

Your schema:

```
@prefix voc: <http://www.unice.fr/voc#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

#class
voc:Man a rdfs:Class;
    rdfs:subClassOf voc:Person ;
    rdfs:seeAlso voc:Women ;
    rdfs:comment "a male adult"@en.

voc:Women a rdfs:Class;
    rdfs:subClassOf voc:Person ;
    rdfs:seeAlso voc:Man ;
    rdfs:comment "a female adult"@en.

voc:Enginneer a rdfs:Class;
    rdfs:subClassOf voc:Women, voc:Man.

#Property
voc:hasParent a rdf:Property;
    rdfs:label "父母"@zh.

voc:hasMother a rdf:Property;
    rdfs:subProertyOf voc:hasParent;
    rdfs:domain voc:Women ;
    rdfs:range voc:Person ;
    rdfs:label "母親"@zh.

voc:hasFather a rdf:Property;
    rdfs:subProertyOf voc:hasParent;
    rdfs:domain voc:Man ;
    rdfs:range voc:Person ;
    rdfs:label "父親"@zh.

voc:hasChild a rdf:Property;
    rdfs:label "孩子"@zh.

voc:hasSpouse a rdf:Property;
    rdfs:label "伴侶"@zh.
```

```
voc:hasColleague a rdf:Property;
    rdfs:label "同事"@zh.

#label
voc:name a rdf:Property;
    rdfs:label "nom"@fr,"name"@en.
voc:age a rdf:Property.
voc:shoesSize a rdf:Property.
voc:trouserSize a rdf:Property.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
  <rdf:Description rdf:about="http://www.unice.fr/voc#Women">
    <rdfs:comment xml:lang="en">a female adult</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:seeAlso rdf:resource="http://www.unice.fr/voc#Man"/>
    <rdfs:subClassOf rdf:resource="http://www.unice.fr/voc#Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#hasSpouse">
    <rdfs:label xml:lang="zh">伴侣</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#Man">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="http://www.unice.fr/voc#Person"/>
    <rdfs:comment xml:lang="en">a male adult</rdfs:comment>
    <rdfs:seeAlso rdf:resource="http://www.unice.fr/voc#Women"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#name">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
    <rdfs:label xml:lang="fr">nom</rdfs:label>
    <rdfs:label xml:lang="en">name</rdfs:label>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#shoesSize">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#hasColleague">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
    <rdfs:label xml:lang="zh">同事</rdfs:label>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#hasChild">
    <rdfs:label xml:lang="zh">孩子</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#hasParent">
```

```
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
    <rdfs:label xml:lang="zh">父母</rdfs:label>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#hasFather">
    <rdfs:domain rdf:resource="http://www.unice.fr/voc#Man"/>
    <rdfs:subProertyOf rdf:resource="http://www.unice.fr/voc#hasParent"/>
    <rdfs:range rdf:resource="http://www.unice.fr/voc#Person"/>
    <rdfs:label xml:lang="zh">父親</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#hasMother">
    <rdfs:range rdf:resource="http://www.unice.fr/voc#Person"/>
    <rdfs:label xml:lang="zh">母親</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
    <rdfs:domain rdf:resource="http://www.unice.fr/voc#Women"/>
    <rdfs:subProertyOf rdf:resource="http://www.unice.fr/voc#hasParent"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#Enginneer">
    <rdfs:subClassOf rdf:resource="http://www.unice.fr/voc#Man"/>
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="http://www.unice.fr/voc#Women"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#trouserSize">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.unice.fr/voc#age">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
  </rdf:Description>
</rdf:RDF>
```

- Check that your RDF schema and RDF files are valid using the W3C's RDF validation service.
- Launch the standalone interface of Corese and load your files Family.rdfs and Jen.rdf
- The interface contains a default SPARQL query:

```
Select ?x ?t where {?x rdf:type ?t}
```

Launch the query and look at the results.

Screenshot:

	?x	?y	?p
	rdfs:comment	rdf:Property	rdf:type
	rdfs:seeAlso	rdf:Property	rdf:type
	<http://www.unice.fr/voc#hasFather>	rdf:Property	rdf:type
	rdfs:domain	rdf:Property	rdf:type
	rdfs:subProertyOf	rdf:Property	rdf:type
	rdfs:range	rdf:Property	rdf:type
	<http://www.unice.fr/voc#hasMother>	rdf:Property	rdf:type
	<http://www.unice.fr/voc#Enginneer>	rdfs:Class	rdf:type
	<http://www.unice.fr/voc#Man>	"a male adult"@en	rdfs:comment
	<http://www.unice.fr/voc#Women>	"a female adult"@en	rdfs:comment
	<http://www.unice.fr/voc#hasFather>	<http://www.unice.fr/voc#Man>	rdfs:domain
	<http://www.unice.fr/voc#hasMother>	<http://www.unice.fr/voc#Women>	rdfs:domain
	<http://www.unice.fr/voc#name>	"nom"@fr	rdfs:label
	<http://www.unice.fr/voc#name>	"name"@en	rdfs:label

- Modify your ontology to declare the classes of Man and Woman as sub classes of Human (don't change the data), reload the schemas and data and search for the humans to see the results

Screenshot:

```
1  prefix voc: <http://www.unice.fr/voc#> .
2  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
3  prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
4
5  select * where{?x ?z voc:Human}
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://www.unice.fr/voc#hasFather>
2	<http://www.unice.fr/voc#hasMother>
3	<http://www.unice.fr/voc#Man>
4	<http://www.unice.fr/voc#Women>

```
5  select * where{?x a voc:Human}
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://www.unice.fr/data#Steffen>
2	<http://www.unice.fr/data#Jen>
3	<http://www.unice.fr/data#Anny>
4	<http://www.unice.fr/data#Thomas>
5	<http://www.unice.fr/data#Catherine>
6	<http://www.unice.fr/data#Fabien>
7	<http://www.unice.fr/data#Seb>

Explanation:

We can see that now Man and Women is subclass of Human, and if I search for the person all the people belong to man and woman is belong to human.

- Modify your ontology to declare the properties hasChild and hasSpouse as sub properties of familyLink (don't change the data), reload the schemas and data and search for the family links to see the results.

Screenshot:

```
5  select * where{?x ?y voc:familyLink}
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.unice.fr/voc#hasChild>	rdfs:subPropertyOf
2	<http://www.unice.fr/voc#hasSpouse>	rdfs:subPropertyOf
3	<http://www.unice.fr/voc#hasMother>	rdfs:subPropertyOf
4	<http://www.unice.fr/voc#hasFather>	rdfs:subPropertyOf

```
5  select * where{?x voc:familyLink ?z}
```

Graph | XML/RDF | Table | Validate

num	?x	?z
1	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/data#Steffen>
2	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/data#Anny>
3	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/data#Seb>
4	<http://www.unice.fr/data#Seb>	<http://www.unice.fr/data#Steffen>
5	<http://www.unice.fr/data#Seb>	<http://www.unice.fr/data#Anny>

Explanation:

Here I added hasChild hasSpouse hasMother and has Father to be the subproperty of familyLink, so when I query for subproperty it would give me all the people who has those link.

- Modify your ontology to declare the class FamilyMember and use it to specify the signature of the property familyLink (don't change the data) then reload the schemas and data and search for the family members.

Screenshot:

```
1  prefix voc: <http://www.unice.fr/voc#> .
2  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
3  prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
4  select * where {
5    ?x a voc:FamilyMember
6  }
```

Graph | XML/RDF | Table | Validate

num	
1	<http://www.unice.fr/data#Steffen>
2	<http://www.unice.fr/data#Jen>
3	<http://www.unice.fr/data#Anny>
4	<http://www.unice.fr/data#Seb>

Explanation:
When we put the class FamilyMember. Then define the domain and range to FamilyMember of Family link. Finally we query the Family Member, we got all the family members who was linked with Family link.
Code:

About the human.rdfs schema

1. If you don't have the human schema file yet, download the RDF schema available at this address and save it as "human.rdfs":
   http://wimmics.inria.fr/doc/tutorial/human_2013.rdfs
2. What is the namespace associated with this ontology? How was it associated?

`Namespace` http://www.inria.fr/2007/09/11/humans.rdfs
So everytime we declare a new variable it is associated to this namespace

3. Look at the XML structure of this file and locate different syntactic properties: the different possible uses of the markup (ex: opening tag and closing, single tag), the use of namespaces for qualified names, the use of entities, etc.

< and > for declaration is using the namespace. When we declare a new class it would be located to the base namespaces. It would transfer to http://www.inria.fr/2007/09/11/humans.rdfs +# +"ID" where ID would be replace by the defined name

4. Locate the use of the terms of the RDF (S) language: Class, Property, label, comment, range, domain, subClassOf, subPropertyOf, etc. To what namespaces are they associated?

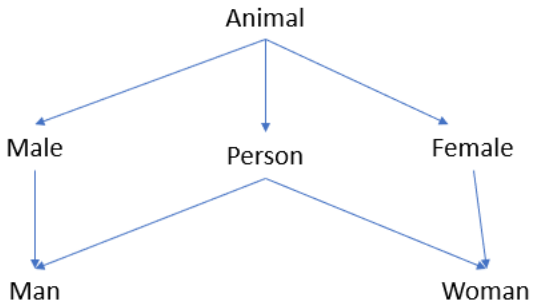
Class, Property	Name space of xml:base
Label, comment, range, domain, subClassOf, subPropertyOf	Namespace of xmlns:rdfs

5. What are the classes of resources that can have the age property? Explain

Person also (Man, Woman, Lecturer, Researcher). People should have age and all the other 4 belong to person.

6. Look at the beginning of the file and draw the subgraph of the hierarchy containing the classes `Animal`, `Man` and `Woman`.

Drawing of hierarchy:



Query the schema itself

Reset or relaunch the standalone Corese search engine interface and load the file `human.rdfs` (and only this one).

1. Write a query to find all the classes of the ontology.

query:

```
select distinct ?x where {
  ?x ?p rdfs:Class
}
```

2. Write a query to find all the links subClassOf in the ontology.

query:

```
select distinct ?x where {
  ?x rdfs:subClassOf ?y
}
```

3. Write a query to find the definitions and translations of "shoe size" (*other* labels and comments in different languages for the resource labeled "shoe size").

query:

```
select * where {
  <http://www.inria.fr/2007/09/11/humans.rdfs#shoesize> rdfs:label ?y;
rdfs:comment ?z.
filter(?y!="shoe size"@en && (lang(?y)!=lang(?z))).
}
```

answers:

Graph | XML/RDF | Table | Validate

num	?y	?z	?
1	"size"@en	"\ntaille, exprimÃ©e en points...	fr
2	"pointure"@fr	"\nexpress in some way the a...	en

4. Write a query to find the synonyms in French of the word 'personne' in French (*other* labels in the same language for the same resource/class/property). What are the answers?

query:

```
select ?c where
{{ select ?x where {?x ?type "personne"@fr}}
?a  rdfs:label ?c.
filter(?a=?x&& lang(?c)='fr' &&?c!= "personne"@fr)}
```

answers:

?c
"homme"@fr
"Ãªtre humain"@fr
"humain"@fr

5. Write a query to find the different meaning of the term "size" (disambiguation using the different comments attached to different resources/classes/properties having the label "size"). What are the answers?

query:

```
select ?c where {?x ?p "size"@en; rdfs:comment ?c}
```

answers:

?c
"\nexpress in some way the approximate length of the shoes for a person.\n"@en
"\ntaille, exprimÃ©e en points, des chaussures d'une personne.\n"@fr
"\nexpress in some way the approximate dimensions of the shirts of a person.\n"@en
"\ndimensions approximatives des chemises portÃ©es par une personne.\n"@fr
"\nexpress in some way the approximate dimensions of the trousers of a person.\n"@en
"\ndimensions approximatives des pantalons portÃ©s par une personne.\n"@fr

6. Write a query to find the properties that use the class Person in their signatures?

query:

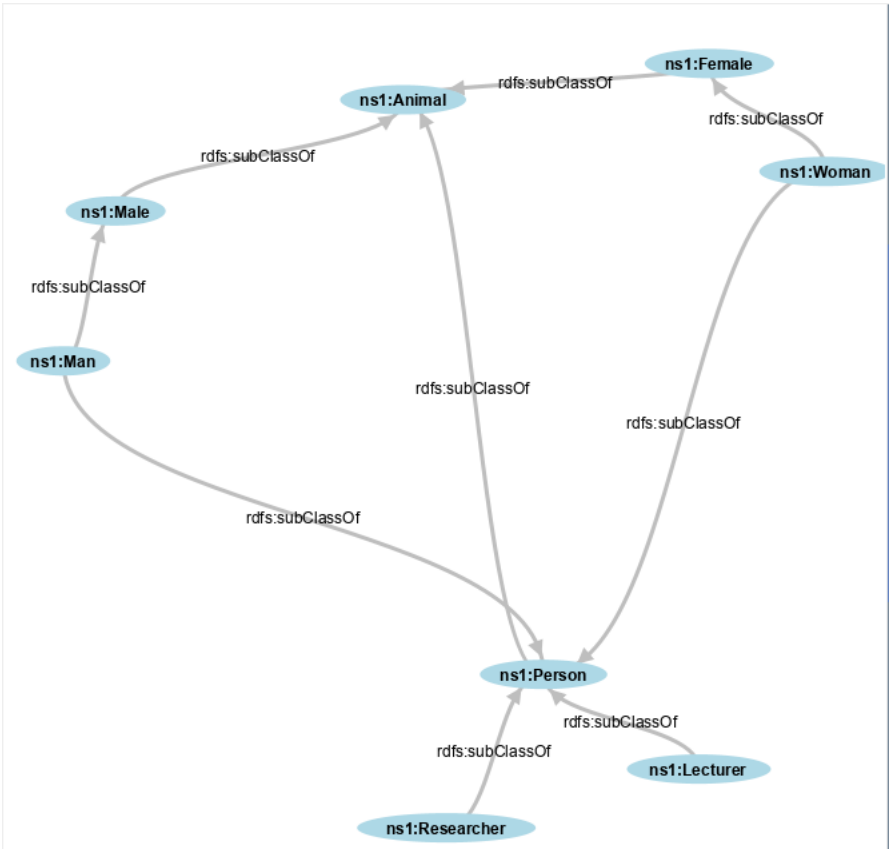
```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#>
select distinct ?x where {
  ?x ?p :Person; rdf:type rdf:Property.}
```

7. Rebuild the hierarchy of Classes (CONSTRUCT) considering only the classes in the humans.rdfs schema

query:

```
construct {?x rdfs:subClassOf ?y}
where {?x rdfs:subClassOf ?y}
```

screenshot:

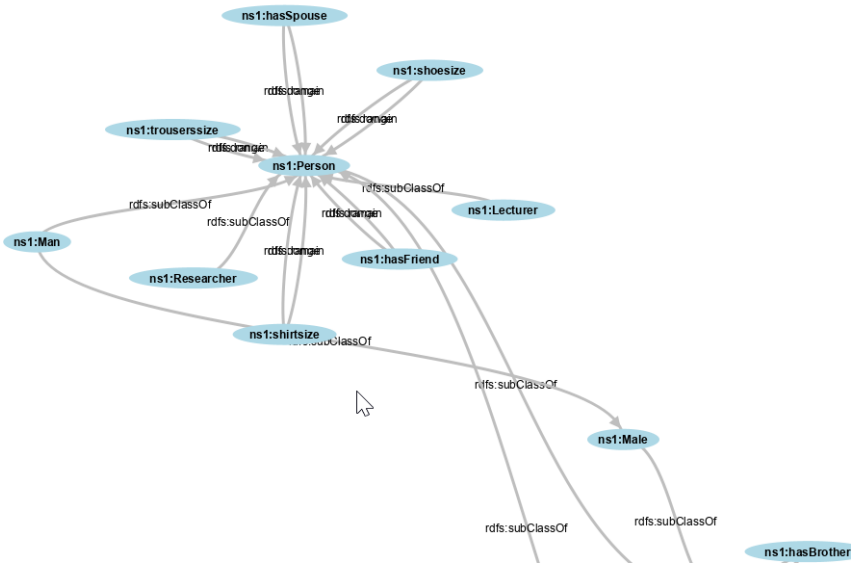


8. To the previous CONSTRUCT add the signatures of the relations.

query:

```
construct {?x rdfs:subClassOf ?y.
?p rdfs:domain ?c1.
?p rdfs:range ?c1.
}
where{ {?x rdfs:subClassOf ?y} union {?p rdfs:domain ?c1} union {?p
rdfs:range ?c2}}
```

screenshot:



You now know how to query schemas on the semantic Web!

Query data augmented by an RDFS schema

Question 1

1. Reset the Corese engine and load only the annotations (.rdf)
2. Write a query to find the Persons.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select * where {?x a :Person}
```

Number of results before: 7

3. Load the schema (.rdfs)
4. Rerun the query to find the Persons and explain the result.

New number of results after and your explanation: 17

Because that some person was under class Lecturer, Man, Woman and Reasearcher instead of a Person. When we load our predefined Schema in, it automatically add in the class person to those objects who is under the class of Person.

Question 2

1. Write a query to find Males and their wives. How many answers do you get? Explain this result.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select * where {?x a :Male; :hasSpouse ?y.}
```

Number of results and explanation: only 1

Harry is a man which make him a person and a male

He has Spouse of Sophie

2. In the data declare that Lucas has to father Karl. Reset Corese, reload the ontology and the data, and then rerun the query to find Males and their wives. Explain the new result.

Line added in RDF:

```
<hasFather rdf:resource="#Karl"/>
```

Number of results before and after and explanation:

Now 2 Karl is added. He was a Person, but by adding it him as a father of Lucas. It's the same as it's a parent. Also it give Karl a Man class, which due to "Man" class definition, he is also in the class "Male". As Karl already have "Catherine" as his wife, he is now shown to the answer

Question 3

1. Write a query to find the Lecturers and their types. How many answers do you get? See how this typing is declared in the data and explain the result.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select * where {?x a :Lecturer; a ?y. filter(?y!=:Lecturer)}
```

?x	?y
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	<http://www.inria.fr/2007/09/11/humans.rdfs#Researcher>
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	<http://www.inria.fr/2007/09/11/humans.rdfs#Female>

Number of results and your explanation:

I have got 6 answers (filtered out the Lecturer class). Eve was just a lecture, but since lecture is person and person is animal too so that it would have all the type in her class. Laura is the same principle.

2. Write a query to find common instances of the classes Person and Male. See how this typing is declared in the data and explain the presence of Jack.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select * where {?x a :Person; a :Male}
```

Your explanation of the result:

In the data Jack was shown as a Man, while man is subclass of both person and male it also inherit those classes

Question 4

Write a query to find the hasAncestor relations. Explain the result after checking where this property is used in the data.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select * where {?x :hasAncestor ?y}
```

Your explanation of the result:

hasAncestor is declare in the schema, from an animal to an animal. And hasParent is a subProperty of has Ancestor therefore when we have the relationship has parent the relationship has ancestor would be inherit.

Question 5

1. Write a query to find the family cores (couples and their children) using a SELECT

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select * where {?x :hasSpouse ?y; :hasChild ?z}
```

2. Modify it to display the result with a CONSTRUCT query

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
construct {?x :hasSpouse ?y; :hasChild ?z}
where {?x :hasSpouse ?y; :hasChild ?z}
```

Question 6

1. Declare the olderThan relationship in the schema to indicate between two people which is eldest and construct the arcs between peoples with a SPARQL query

Addition to schema:

```
<rdf:Property rdf:ID="olderThan">
<domain rdf:resource="#Animal"/>
<range rdf:resource="#Animal"/>
</rdf:Property>
```

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
Insert {?x :olderThan ?y} where {?x :age ?a1. ?y:age ?a2. filter(?a1>?a2)}
```

2. Find a query that generates only the minimum number of links without redundancy with olderThan transitivity.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select ?x ?y (min(?aged) as ?min) where {
  ?x :olderThan ?y; :age ?a1.
  ?y :age ?a2.
  bind((?a1-?a2) as ?aged)}
group by ?x
```

Question 7

Write a query to find for John the properties which label contains the string "size" and the value of these properties.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select ?y ?z where{
?x :name "John"; ?y ?z.
filter(contains(?y,"size"))}
```

Question 8

Use the ontology to document your answers in natural language: write a query to find the types and properties of Laura in French.

Query:

```
prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
select ?y
where {
    ?x :name "Laura";?property ?type.
    {
    {?property rdfs:label ?y.}
    UNION
    {?type rdfs:label ?y.}
    }
    filter(lang(?y)="fr").}
```