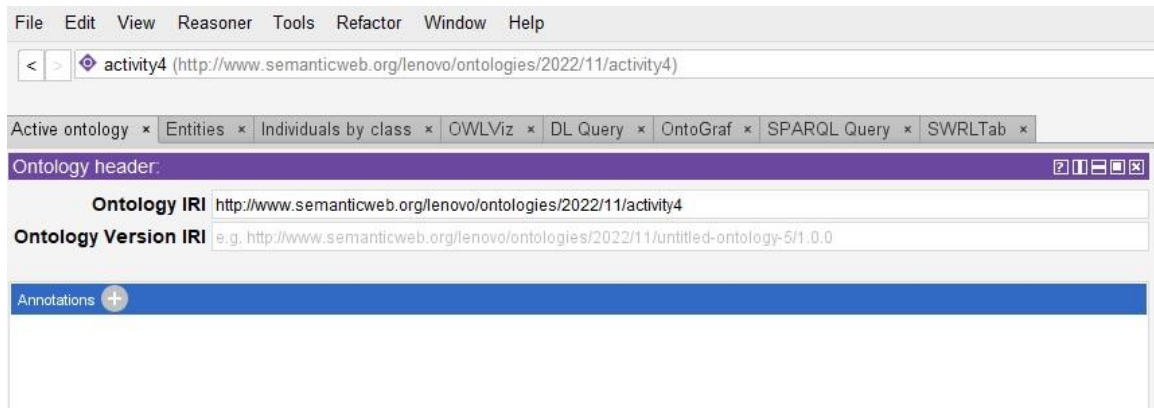


Nom et Prénom :	Nasri Jouhaina
Groupe :	2 Mastère Génie Logiciel
DM4 :	ONTOLOGIE AVEC OWL

Travail à faire :

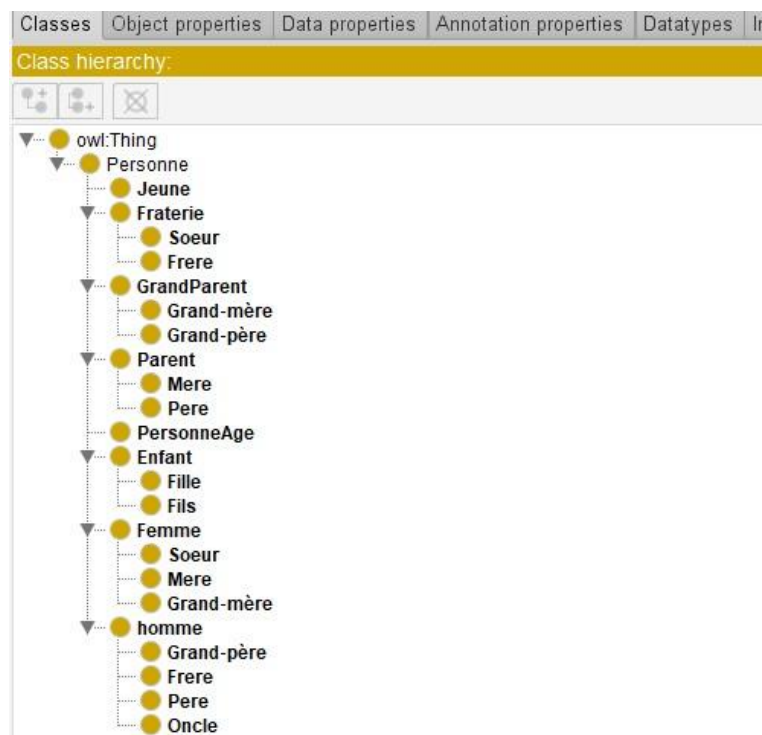
1. Créer une nouvelle ontologie en Protégé et donner lui un URI de votre choix :

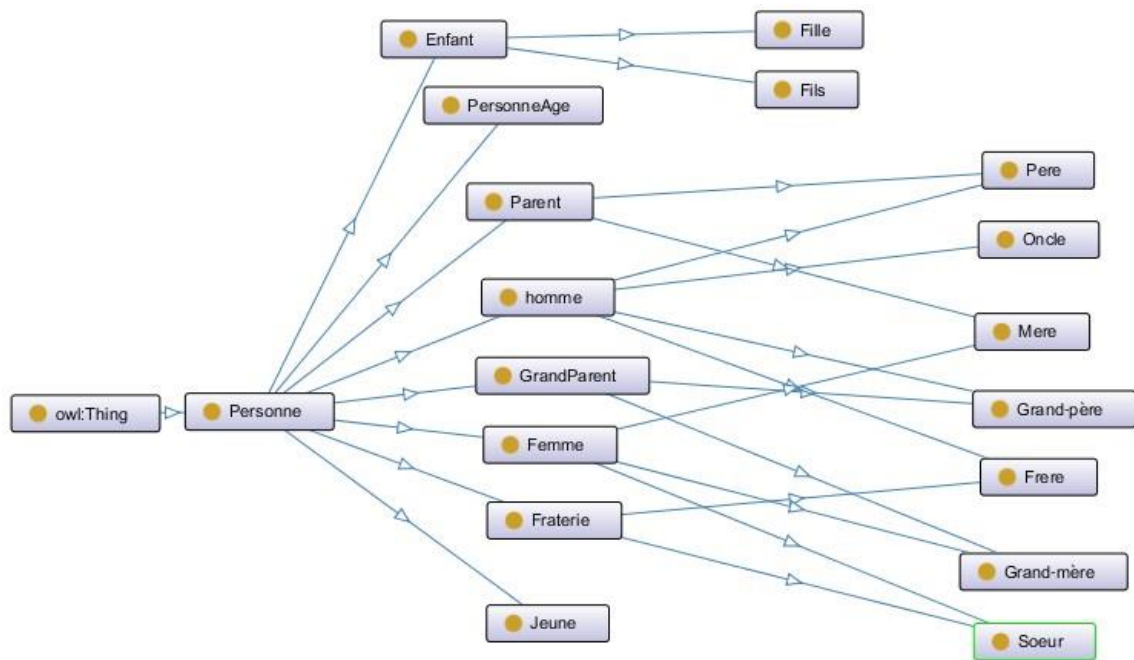


2. Enregistrer l'ontologie avec la syntaxe RDF/XML et l'extension .owl



3. Créer la hiérarchie des classes suivante :





4. Création des propriétés pour les classes :

- Une personne possède un nom, un âge et une nationalité.

Classes	Object properties	Data properties
Data property hierarchy: nationalite		
<div> <div>owl:topDataProperty</div> <div> <div>nationalite</div> <div>age</div> <div>name</div> </div> </div>		

Description: nationalite

Equivalent To +

SubProperty Of +

Domains (intersection) +

Personne

Ranges +

xsd:string

Description: age

Equivalent To +

SubProperty Of +

Domains (intersection) +

Personne

Ranges +

xsd:int

Description: name

Equivalent To +

SubProperty Of +

Domains (intersection) +

Personne

Ranges +

xsd:string

- Deux personnes peuvent se marier

Description: se_marier_avec

Equivalent To

SubProperty Of

Inverse Of

Domains (intersection)

Personne

Ranges (intersection)

Personne

Disjoint With

SuperProperty Of (Chain)

- Une personne est le parent d'une autre personne

Description: estParentDe

Equivalent To

SubProperty Of

Inverse Of

estEnfantDe

Domains (intersection)

Personne

Ranges (intersection)

Personne

Disjoint With

SuperProperty Of (Chain)

- Un homme est le père d'une personne

Description: estPereDe

Equivalent To

SubProperty Of

estParentDe

Inverse Of

Domains (intersection)

homme
 Personne

Ranges (intersection)

Personne

Disjoint With

SuperProperty Of (Chain)

- Une femme est la mère d'une personne

Description: estMereDe

Equivalent To

SubProperty Of

estParentDe

Inverse Of

Domains (intersection)

Femme
 Personne

Ranges (intersection)

Personne

Disjoint With

SuperProperty Of (Chain)

- Une personne appartient à la fraterie d'une autre personne

Description: estEnRelationDeFraterieAvec

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

● **Personne**

Ranges (intersection) +

● **Personne**

Disjoint With +

SuperProperty Of (Chain) +

- Un homme est le frère d'une personne

Description: estFrereDe

Equivalent To +

SubProperty Of +

■ **estEnRelationDeFraterieAvec**

Inverse Of +

Domains (intersection) +

● **homme**

● **Personne**

Ranges (intersection) +

● **Personne**

Disjoint With +

SuperProperty Of (Chain) +

- Une femme est la soeur d'une personne

Description: estSoeurDe

Equivalent To +

SubProperty Of +

■ **estEnRelationDeFraterieAvec**

Inverse Of +

Domains (intersection) +

● **Femme**

● **Personne**

Ranges (intersection) +

● **Personne**

Disjoint With +

SuperProperty Of (Chain) +

- Une personne est un enfant d'une autre personne

Description: estEnfantDe

Equivalent To +

SubProperty Of +

Inverse Of +

■ **estParentDe**

Domains (intersection) +

● **Personne**

Ranges (intersection) +

● **Personne**

Disjoint With +

SuperProperty Of (Chain) +

- Un homme est le fils d'une personne

Description: estFilsDe

Equivalent To +

SubProperty Of +
estEnfantDe

Inverse Of +

Domains (intersection) +
 ● **Personne**
 ● **homme**

Ranges (intersection) +
 ● **Personne**

Disjoint With +

SuperProperty Of (Chain) +

- Une femme est la fille d'une personne

Description: estFilleDe

Equivalent To +

SubProperty Of +
estEnfantDe

Inverse Of +

Domains (intersection) +
 ● **Femme**
 ● **Personne**

Ranges (intersection) +
 ● **Personne**

Disjoint With +

SuperProperty Of (Chain) +

5. Création des restrictions sur les classes et propriétés (conditions nécessaires et suffisantes) :

- La classe Oncle a la restriction : est frère d'un parent

Found 6 uses of **Oncle**

- **Oncle** SubClassOf **homme**
- **Oncle** SubClassOf **estFrereDe exactly 1 Parent**
- **Class: Oncle**

Description: Oncle

Equivalent To +

SubClass Of +
 ● **estFrereDe exactly 1 Parent**
 ● **homme**

- La classe Grand-père a la restriction : est père d'un parent

Description: Grand-père

Equivalent To +

SubClass Of +
 ● **estPereDe exactly 1 Parent**
 ● **GrandParent**
 ● **homme**

- La classe Grand-mère a la restriction : est mère d'un parent

Description: Grand-mère

Equivalent To 

SubClass Of 

-  **estMereDe exactly 1 Parent**
-  **Femme**
-  **GrandParent**

- La classe Pere a la restriction : la propriété estPereDe a au moins une instance

Description: Pere

Equivalent To 

SubClass Of 

-  **estPereDe min 1 Enfant**
-  **homme**
-  **Parent**

- La classe Mere a la restriction : la propriété estMereDe a au moins une instance

Description: Mere

Equivalent To 

SubClass Of 

-  **estMereDe min 1 Enfant**
-  **Femme**
-  **Parent**

- La classe Fils a la restriction : la propriété estFilsDe a au moins une instance

Description: Fils

Equivalent To 

SubClass Of 

-  **Enfant**
-  **estFilsDe min 1 Parent**

- La classe Fille a la restriction : la propriété estFilleDe a au moins une instance

Description: Fille

Equivalent To 

SubClass Of 

-  **Enfant**
-  **estFilleDe min 1 Parent**

- La classe Frere a la restriction : la propriété estFrereDe a au moins une instance

Description: Frere

Equivalent To 

SubClass Of 

-  **estFrereDe min 1 Enfant**
-  **Fraterie**
-  **homme**

- La classe Sœur a la restriction : la propriété estSœurDe a au moins une instance

Description: Soeur

Equivalent To

SubClass Of

- estSoeurDe** **min 1** **Enfant**
- Femme**
- Fraterie**

6. Classes disjointes :
- Homme et Femme sont disjointes

Description: Femme

Target for key

Disjoint With

- homme**

- Père et Mère sont disjointes

Description: Mere

Target for key

Disjoint With

- Pere**

Disjoint Union Of

- Fils et Fille sont disjointes

Description: Fille

Target for key

Disjoint With

- Fils**

Disjoint Union Of

- Grand père et Grand mère sont disjointes

Description: Grand-mère

Target for key

Disjoint With

- Grand-père**

Disjoint Union Of

7. Assigner les types pour les propriétés

- La propriété se_marier_avec et estEnRelationDeFraterieAvec sont symétriques
- La propriété estEnRelationDeFraterieAvec est transitive

Characteristics: se_marier_avec

- ☐ Functional
- ☐ Inverse functional
- ☐ Transitive
- ☒ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☐ Irreflexive

Characteristics: estEnRelationDeFraterieAvec

- ☐ Functional
- ☐ Inverse functional
- ☒ Transitive
- ☒ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☐ Irreflexive

- La propriété estEnfantDe est la propriété inverse de la propriété estParentDe

Characteristics: estEnfantDe	Description: estEnfantDe
<input type="checkbox"/> Functional <input checked="" type="checkbox"/> Inverse functional <input type="checkbox"/> Transitive <input type="checkbox"/> Symmetric <input type="checkbox"/> Asymmetric	Equivalent To + SubProperty Of + Inverse Of + estParentDe

- La propriété age est fonctionnelle

Characteristics: age
<input checked="" type="checkbox"/> Functional

8. Création des instances

- Créer des instances pour la classe Homme :

Active ontology	Entities	Individuals by class	OWLViz
Class hierarchy: homme			
owl:Thing Personne Jeune Fraterie GrandParent			
Direct instances:			
For: homme			
Jean John Micheal Paul Peter Thomas Tom			

- Peter, 70, se_marier_avec Marie, de nationalité française

Property assertions: Peter
Object property assertions
estParentDe Paul
se_marier_avec Marie
estParentDe Chloé
estParentDe Sylvie
estParentDe Thomas
Data property assertions
nationalite "française"
name "Peter"
age 70

- Thomas, 40, estFilsDe Peter, de nationalité française

- Paul, 38 estFilsDe Peter

- John, 45, de nationalité anglaise

- Jean, 10, estFilsDe John, anglaise

- Tom, 10, estFilsDe Thomas et Alex

- Micheal, 5, estFilsDe Thomas et Alex

Property assertions: Thomas	
Object property assertions +	
estParentDe	Micheal
estEnfantDe	Peter
estFilsDe	Peter
estParentDe	Tom
Data property assertions +	
nationalite	"française"
age	40
name	"Thomas"

Property assertions: Paul	
Object property assertions +	
estEnfantDe	Peter
estFilsDe	Peter
Data property assertions +	
name	"estFilsDe"
age	38
Negative object property assertions +	
Negative data property assertions +	

Property assertions: John	
Object property assertions +	
estParentDe	Jean
Data property assertions +	
name	"John"
age	45
nationalite	"anglaise"
Negative object property assertions +	
Negative data property assertions +	

Property assertions: Jean	
Object property assertions +	
estEnfantDe	John
estFilsDe	John
Data property assertions +	
name	"Jean"
nationalite	"anglaise"
age	10
Negative object property assertions +	
Negative data property assertions +	

Property assertions: Tom	
Object property assertions +	
estEnfantDe	Thomas
estFilsDe	Thomas
Data property assertions +	
name	"Tom"
age	10

Property assertions: Micheal	
Object property assertions +	
estFilsDe	Thomas
estEnfantDe	Thomas
Data property assertions +	
name	"Micheal"
age	5
Negative object property assertions +	
Negative data property assertions +	

- Créer des instances pour la classe Femme :

- Marie, 69, de nationalité française

- Sylvie, 30, estFilleDe Marie et Peter
- Sylvie se_marier_avec John

- Chloé, 18, estFilleDe Marie et Peter

- Claude, 5, estFilleDe Sylvie, de nationalité française

- Alex, 25, se_marier_avec Thomas


For:  Femme

 Alex
 Chloé
 Claude
 Marie
 Sylvie

Property assertions: Marie

Object property assertions 

 **estParentDe** Chloé
 **estParentDe** Sylvie

Data property assertions 



 **name** "Marie"
 **nationalite** "française"
 **age** 69

Property assertions: Sylvie

Object property assertions 

 **estEnfantDe** Marie
 **estFilleDe** Peter
 **estParentDe** Claude
 **estEnfantDe** Peter
 **estFilleDe** Marie
 **se_marier_avec** John


Data property assertions 

 **name** "Sylvie"
 **age** 30

Property assertions: Chloé

Object property assertions 

 **estFilleDe** Peter
 **estEnfantDe** Peter
 **estFilleDe** Marie
 **estEnfantDe** Marie


Data property assertions 

 **age** 18
 **name** "Chloé"

Property assertions: Claude

Object property assertions 


 **estEnfantDe** Sylvie
 **estFilleDe** Sylvie

Data property assertions 



 **age** 5
 **nationalite** "française"
 **name** "Claude"

Property assertions: Alex

Object property assertions 

 **se_marier_avec** Thomas

Data property assertions 

 **age** 25
 **name** "Alex"

9. Vérification de la cohérence de l'ontologie

Property assertions: Thomas

Object property assertions +

- estParentDe Micheal
- estEnfantDe Peter
- estFilsDe Peter
- estParentDe Tom
- se_marier_avec Alex

SubClass Of +

- Enfant
- estFilleDe min 1 Parent
- Femme

Property assertions: Marie

Object property assertions +

- estParentDe Chloé
- estParentDe Sylvie
- se_marier_avec Peter

Data property assertions +

- name "Marie"
- nationalite "française"
- age 69

SubClass Of +

- Enfant
- estFilsDe min 1 Parent
- homme

Property assertions: John

Object property assertions +

- estParentDe Jean
- se_marier_avec Sylvie

Data property assertions +

- name "John"
- age 45
- nationalite "anglaise"

- Le raisonneur sélectionné effectue un certain nombre de tâches de raisonnement en fonction des types d'inférences que Protégé est configuré pour afficher.
- On declare que Petar se_marier_avec Marie, mais on ne declare pas que Marie se_marie_avec Petar mais on mettre que se_marier_avec est symetrique, et pour cela et grace à le raisonneur on peut voir la relation symetrique.
- Aussi, pour les classes homme et femme sont générer automatiquement.
- Mais il y a des autres relations comme estFilsDe, estFilleDe, estSoeurDe qui ne sont pas générer automatiquement, aussi les individus ne sont pas generer automatiquement comme Enfant, Sœur, Jeune, Père etc...
- Il n'y a aucun moyen de corriger cette insuffisance sans utilisation de règles SWRL.

10. Règles SWRL

Oncle :

```
homme(?x) ^ estEnRelationDeFraterieAvec(?x, ?p)^:estParentDe(?p, ?e) -> Oncle(?x)
```

Description: Paul

Types +

- homme
- Personne
- Oncle

Description: Thomas

Types +

- homme
- Personne
- Oncle

Sœur :

$\text{estFilleDe}(?x, ?p) \wedge \text{estParentDe}(?p, ?y) \rightarrow \text{estSoeurDe}(?x, ?y) \wedge \text{Soeur}(?x)$

Description: Sylvie

Types

- Femme
- Personne
- Soeur

Same Individual As

Different Individuals

Property assertions: Sylvie

Object property assertions

- estEnfantDe Marie
- estFilleDe Peter
- estParentDe Claude
- estEnfantDe Peter
- estFilleDe Marie
- se_marier_avec John
- estEnRelationDeFraterieAvec Chloé
- estEnRelationDeFraterieAvec Paul
- estEnRelationDeFraterieAvec Thomas
- estEnRelationDeFraterieAvec Sylvie
- estParentDe Jean
- estSoeurDe Chloé
- estSoeurDe Paul
- estSoeurDe Thomas
- estSoeurDe Sylvie

Description: Claude

Types

- Femme
- Personne
- Enfant
- Soeur

Same Individual As

Different Individuals

Property assertions: Claude

Object property assertions

- estEnfantDe Sylvie
- estFilleDe Sylvie
- estEnRelationDeFraterieAvec Claude
- estEnRelationDeFraterieAvec Jean
- estEnfantDe John
- estSoeurDe Claude
- estSoeurDe Jean

Description: Chloé

Types

- Femme
- Personne
- Soeur

Same Individual As

Different Individuals

Property assertions: Chloé

Object property assertions

- estFilleDe Peter
- estEnfantDe Peter
- estFilleDe Marie
- estEnfantDe Marie
- estEnRelationDeFraterieAvec Chloé
- estEnRelationDeFraterieAvec Paul
- estEnRelationDeFraterieAvec Thomas
- estEnRelationDeFraterieAvec Sylvie
- estSoeurDe Chloé
- estSoeurDe Paul
- estSoeurDe Thomas
- estSoeurDe Sylvie

L'enfant d'une personne mariée est aussi l'enfant de son époux(se).

$\text{estEnfantDe}(?x, ?p) \wedge \text{se_marier_avec}(?p, ?y) \rightarrow \text{estEnfantDe}(?x, ?y)$

Property assertions: Alex

Object property assertions

- se_marier_avec Thomas
- estParentDe Micheal
- estParentDe Tom

Property assertions: Claude

Object property assertions +

- estEnfantDe Sylvie
- estFilleDe Sylvie
- estEnfantDe John

Property assertions: Jean

Object property assertions +

- estEnfantDe John
- estFilsDe John
- estEnfantDe Sylvie

Property assertions: Marie

Object property assertions +

- estParentDe Chloé
- estParentDe Sylvie
- estParentDe Paul
- estParentDe Thomas
- se_marier_avec Peter

Property assertions: John

Object property assertions +

- estParentDe Jean
- estParentDe Claude
- se_marier_avec Sylvie

Property assertions: Micheal

Object property assertions +

- estFilsDe Thomas
- estEnfantDe Thomas
- estEnfantDe Alex

Property assertions: Sylvie

Object property assertions +

- estEnfantDe Marie
- estFilleDe Peter
- estParentDe Claude
- estEnfantDe Peter
- estFilleDe Marie
- se_marier_avec John
- estParentDe Jean

Property assertions: Tom

Object property assertions +

- estEnfantDe Thomas
- estFilsDe Thomas
- estEnfantDe Alex

Property assertions: Thomas

Object property assertions +

- estParentDe Micheal
- estEnfantDe Peter
- estFilsDe Peter
- estParentDe Tom
- estEnfantDe Marie
- se_marier_avec Alex

Property assertions: Chloé

Object property assertions +

- estFilleDe Peter
- estEnfantDe Peter
- estFilleDe Marie
- estEnfantDe Marie

Data property assertions +

- age 18
- name "Chloé"

Les personnes qui ont un parent en commun sont en relation de fraterie

`estParentDe(?p, ?x) ^ estParentDe(?p, ?y) ->estEnRelationDeFraterieAvec(?x, ?y)`

	Name	Rule
✓	S1	autogen0:estEnfantDe(?x, ?p) ^ autogen0:se_marier_avec(?p, ?y) -> autogen0:estEnfantDe(?x, ?y)
✓	S2	autogen0:estParentDe(?p, ?x) ^ autogen0:estParentDe(?p, ?y) -> autogen0:estEnRelationDeFraterieAvec(?x, ?y)

Property assertions: Chloé

Object property assertions

estFilleDe Peter

estEnfantDe Peter

estFilleDe Marie

estEnfantDe Marie

estEnRelationDeFraterieAvec Chloé

estEnRelationDeFraterieAvec Paul

estEnRelationDeFraterieAvec Thomas

estEnRelationDeFraterieAvec Sylvie

Property assertions: Jean

Object property assertions

estEnfantDe John

estFilsDe John

estEnRelationDeFraterieAvec Claude

estEnRelationDeFraterieAvec Jean

estEnfantDe Sylvie

Property assertions: Micheal

Object property assertions

estFilsDe Thomas

estEnfantDe Thomas

estEnRelationDeFraterieAvec Micheal

estEnRelationDeFraterieAvec Tom

estEnfantDe Alex

Data property assertions

name "Micheal"

age 5

Property assertions: Thomas

Object property assertions

estParentDe Micheal

estEnfantDe Peter

estFilsDe Peter

estParentDe Tom

estEnRelationDeFraterieAvec Chloé

estEnRelationDeFraterieAvec Paul

estEnRelationDeFraterieAvec Thomas

estEnRelationDeFraterieAvec Sylvie

estEnfantDe Marie

se_marier_avec Alex

Property assertions: Claude

Object property assertions

estEnfantDe Sylvie

estFilleDe Sylvie

estEnRelationDeFraterieAvec Claude

estEnRelationDeFraterieAvec Jean

estEnfantDe John

Property assertions: Paul

Object property assertions

estEnfantDe Peter

estFilsDe Peter

estEnRelationDeFraterieAvec Chloé

estEnRelationDeFraterieAvec Paul

estEnRelationDeFraterieAvec Thomas

estEnRelationDeFraterieAvec Sylvie

estEnfantDe Marie

Data property assertions

name "estFilsDe"

age 38

Property assertions: Sylvie

Object property assertions

estEnfantDe Marie

estFilleDe Peter

estParentDe Claude

estEnfantDe Peter

estFilleDe Marie

se_marier_avec John

estEnRelationDeFraterieAvec Chloé

estEnRelationDeFraterieAvec Paul

estEnRelationDeFraterieAvec Thomas

estEnRelationDeFraterieAvec Sylvie

estParentDe Jean

Property assertions: Tom

Object property assertions

estEnfantDe Thomas

estFilsDe Thomas

estEnRelationDeFraterieAvec Micheal

estEnRelationDeFraterieAvec Tom

estEnfantDe Alex

Un enfant prend les nationalités de ses parents

$\text{estParentDe}(?p, ?x) \wedge \text{nationalite}(?p, ?n) \rightarrow \text{nationalite}(?x, ?n)$

Les personnes âgées de moins de 12 ans sont des enfants

$\text{Personne}(?x) \wedge \text{age}(?x, ?a) \wedge \text{swrlb:lessThan}(?a, 12) \rightarrow \text{Enfant}(?x)$

Description: Jean

Types +

- homme
- Personne
- Enfant

Description: Claude

Types +

- Femme
- Personne
- Enfant

Description: Micheal

Types +

- homme
- Personne
- Enfant

Description: Tom

Types +

- homme
- Personne
- Enfant

Description: Enfant

Equivalent To +

SubClass Of +

- Personne

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

- Claude
- Jean
- Micheal
- Tom