TP1 – Ontologie cinéma

Utilisation de Protégé pour construire des ontologies en OWL2

Use annotations to add labels

- Les termes « comédien » et « comédienne » désignent respectivement les mêmes types de personnes que « acteur » et « actrice ».
- Le terme « movie » en anglais est synonyme du terme « film ».
- Le terme « actor » en anglais (« actress » pour actrice) est synonyme du terme « acteur ».

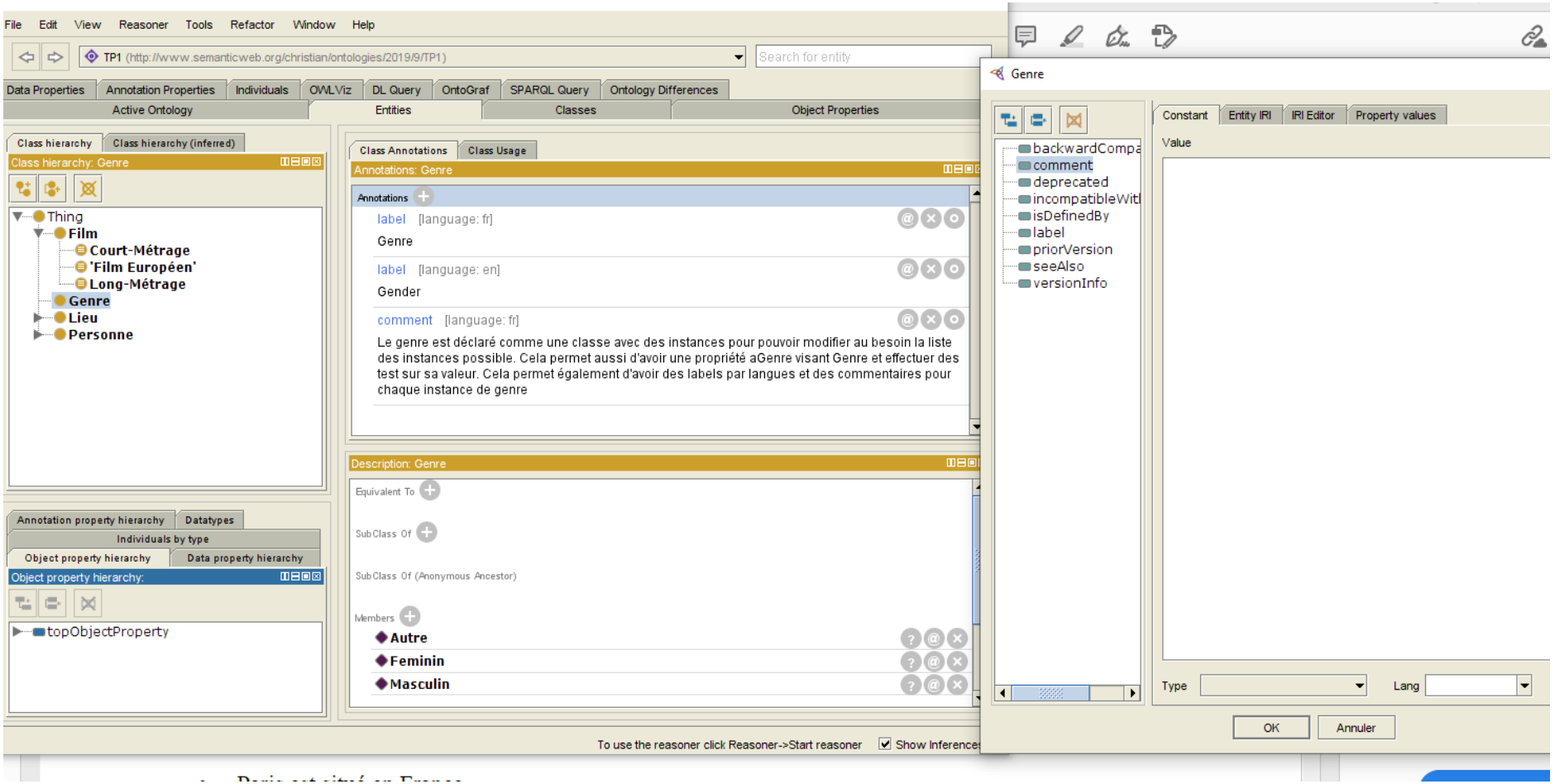
More precise property descriptions in OWL

- 3 types of properties
  - **owl:ObjectProperty** link resources :Flight :departsFrom :Airport
  - **owl:DatatypeProperty** links resources with (typed) litteral values :Flight :departsOn ^^xsd:date
  - **owl:AnnotationProperty** ignored by inference engines, just used as comments or extensions

- Constrains on properties
  - classes vs values
  - Property restriction :IntFlight :departsFrom :IntAirport
  - Cardinality :Flight :departsFrom exactly one :Airport

Annotation properties

- rdfs:annotationProperty rdfs:subClassOf rdfs:Property
- Comments in natural language
- Various types
  - rdfs:label
  - rdfs:comment
  - owl:seeAlso
  - owl:versionInfo
  - You can define your own


File Edit View Reasoner Tools Refactor Window Help
TP1 (http://www.semanticweb.org/christian/ontologies/2019/9/TP1)
Search for entity
Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
Active Ontology Entities Classes Object Properties
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Genre
Thing
Film
Court-Métrage
'Film Européen'
Long-Métrage
Genre
Lieu
Personne
Class Annotations Class Usage
Annotations: Genre
Annotations
label [language: fr]
Genre
label [language: en]
Gender
comment [language: fr]
Le genre est déclaré comme une classe avec des instances pour pouvoir modifier au besoin la liste des instances possible. Cela permet aussi d'avoir une propriété aGenre visant Genre et effectuer des test sur sa valeur. Cela permet également d'avoir des labels par langues et des commentaires pour chaque instance de genre
Description: Genre
Equivalent To
SubClass Of
SubClass Of (Anonymous Ancestor)
Members
Autre
Feminin
Masculin
Annotation property hierarchy Datatypes
Individuals by type
Object property hierarchy Data property hierarchy
Object property hierarchy:
topObjectProperty
To use the reasoner click Reasoner->Start reasoner
Show Inferences
Genre
backwardCompa
comment
deprecated
incompatibleWitl
isDefinedBy
label
priorVersion
seeAlso
versionInfo
Constant Entity IRI IRI Editor Property values
Value
Type
Lang
OK
Annuler

Properties

- Un film se déroule dans un lieu.
- Un lieu peut être situé dans un autre lieu.
- Un artiste joue dans un film.
- Un film a une note.
- Un film a une durée en minutes.
- Un film est réalisé par un réalisateur.
- Une personne a un genre.
- Un film a une nationalité.

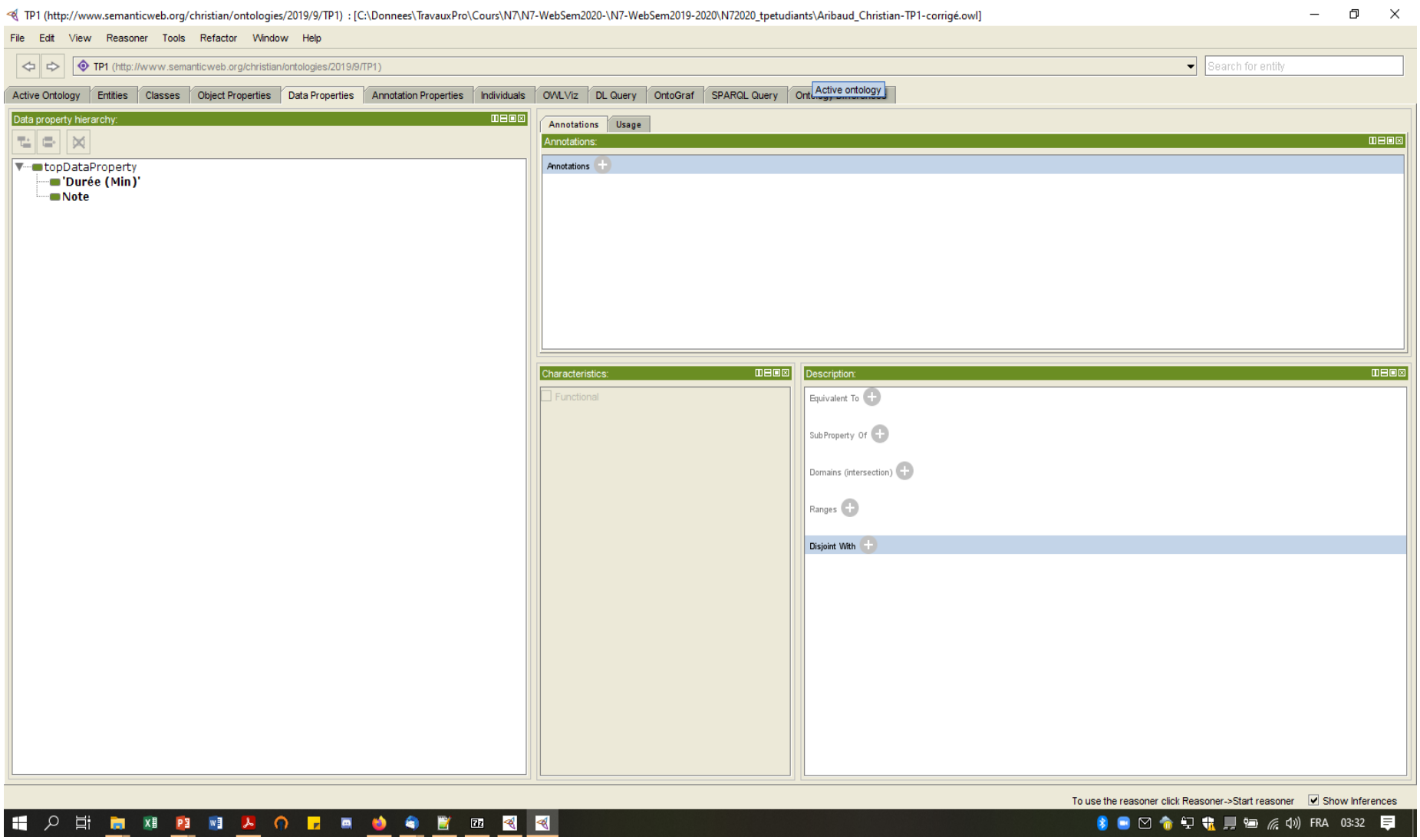
Properties: their ID

- Un film se déroule dans un lieu.
- Un lieu peut être situé dans un autre lieu.
- Un artiste joue dans un film.
- Un film a une note.
- Un film a une durée en minutes.
- Un film est réalisé par un réalisateur.
- Une personne a un genre.
- Un film a une nationalité.

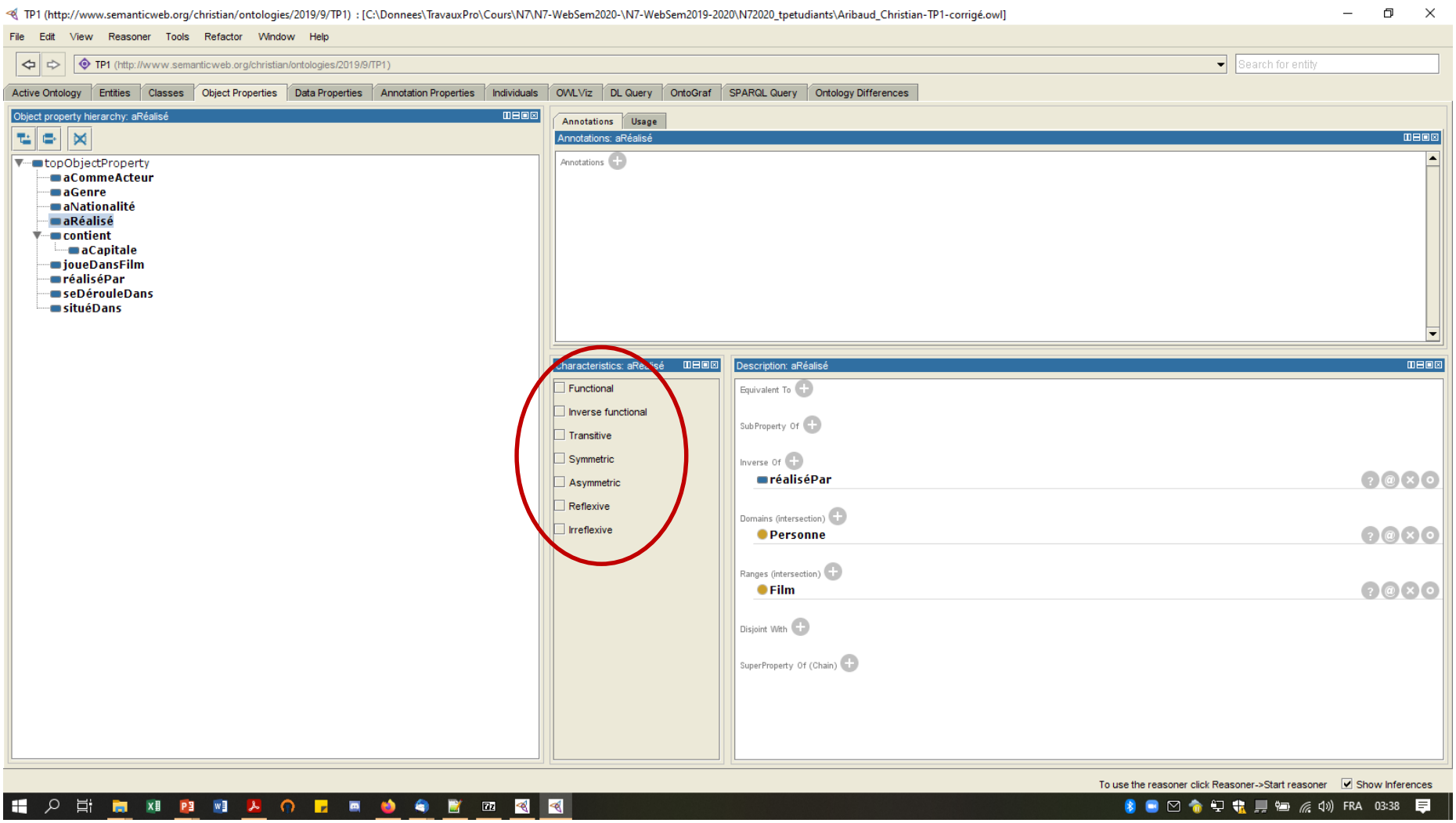
Properties in OWL: Data vs Object types

- DataProperties
  - Links a class with a literal (with an optional xsd type)
- ObjectTypeProperties
  - Link two classes
- Properties
  - can be organized in a hierarchy
  - Can have a domain and range (xsd type for data prop)

Data properties



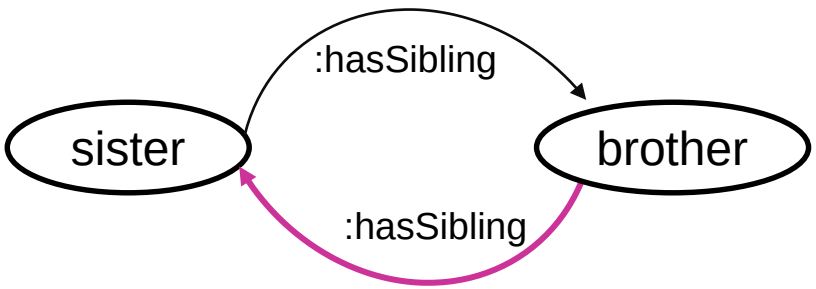
Object properties

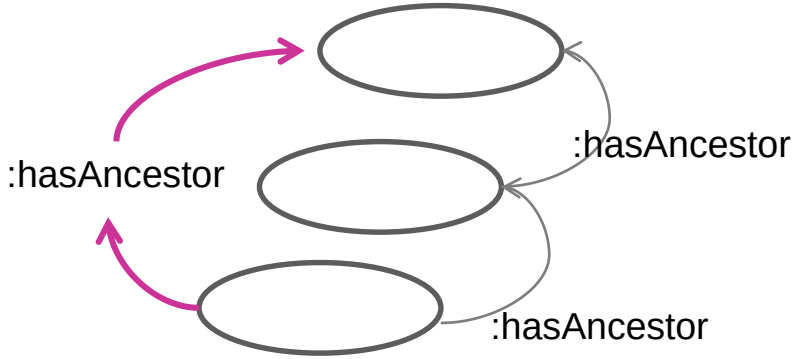


More precise property definitions

- Functional properties = mandatory property
  - Ex: if :departsFrom is functional it means
  - If f :Flight(f) then ∃ a, :Airport(a) and f :departsFrom a
  - $\forall f\ (\mathbf{Flight}(f) \rightarrow \exists\ a,\ (\mathbf{Airport}(a) \wedge \mathbf{departsFrom}\ (f,a))$

- Symmetry :hasSibling
- Transitivity :hasAncestor





- Inverse :hasAncestor and :hasDescendant

Property definition : Property restriction

- Value constrains
  - Restricts all the values of a property

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:allValuesFrom rdf:resource="#Human"/>
</owl:Restriction>
```

  - Restricts at least one value of a property

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:someValuesFrom rdf:resource="#Physician"/>
</owl:Restriction>
```

  - Assigns a value to a property

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#hasParent"/>
    <owl:hasValue rdf:resource="#Marie"/>
</owl:Restriction>
```

Property definition : Property restriction

- Cardinality constraints

  nb of times that a resource can play the same role for a property with distinct values

```
<owl:Restriction>
        <owl:onProperty rdf:resource="#hasParent"/>
        <owl:maxCardinality>2</owl:maxCardinality>
</owl:Restriction>
```

owl:minCardinality

owl:cardinality

Class definition

```
<owl:Class rdf:about="#MarieChild">
        <rdfs:subClassOf>
                <owl:Restriction>
                <owl:onProperty rdf:resource="#hasParent"/>
                <owl:hasValue rdf:resource="#Marie"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Ex: what does this class define?

```
@prefix ex: <http://example.org/>
ex:PersonList rdfs:subClassOf
        [
        a owl:Restriction ;
        owl:onProperty rdf:first ;
        owl:allValuesFrom ex:Person
        ] ,
        [
        a owl:Restriction ;
        owl:onProperty rdf:rest ;
        owl:allValuesFrom ex:PersonList
        ] .
```

Ex: what does this class define?

```
@prefix ex: <http://example.org/>


ex:Human rdfs:subClassOf [
      owl:intersectionOf (
            [
            a owl:Restriction ;
            owl:onProperty ex:hasFather ;
            owl:maxCardinality 1
            ] ,
            [
            a owl:Restriction ;
            owl:onProperty ex:hasMother ;
            owl:maxCardinality 1
            ] .
 )]
```

- Any ex:Human has these 2 properties
  - at most one Father
  - at most one Mother
- Not all things that have One Father and One mother are Humans
- Else, use owl:equivalentClass

Axiomes de propriétés

prop. algébriques 

- Propriétés algébriques de propriétés

  <owl:SymmetricProperty rdf:ID="hasSpouse/">
  <owl:TransitiveProperty rdf:ID="hasAncestor/">
  <owl:ReflexiveProperty rdf:about="#hasRelative"/>

- Relations entre propriétés
  - Relations inverse

    <owl:ObjectProperty rdf:ID="hasChild">
    < owl:inverseOf rdf:resource="#hasParent"/>
    </owl:ObjectProperty>
  - Relations équivalentes (en termes d'extension)

    owl:equivalentProperty
- Contraintes de cardinalité

  <owl:FunctionalProperty rdf:ID="#hasMother"/>
  <owl:InverseFunctionalProperty rdf:ID="#isMotherOf"/>

Ontology population with instances and relations

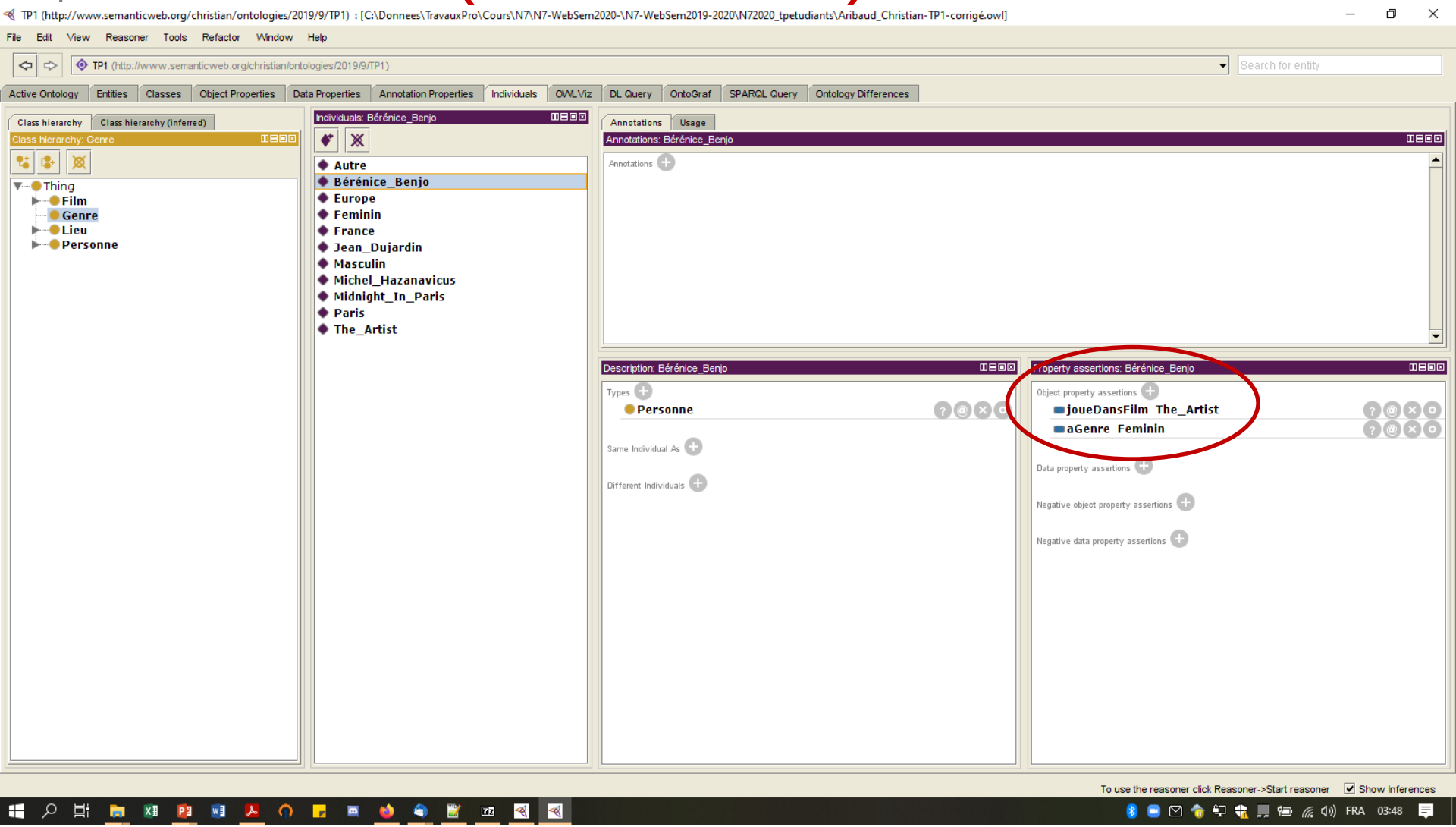
- Paris est situé en France.
- France est situé en Europe.
- Midnight in Paris s'est déroulé à Paris.
- Midnight in Paris a pour réalisateur Woody Allen.
- Jean Dujardin a joué dans The Artist.
- Bérénice Bejo a joué dans The Artist.
- Bérénice Bejo est du genre féminin.
- Jean Dujardin est du genre masculin.

Ontology population with instances and relations

- Explicit requirements
  - Paris est situé en France.
  - France est situé en Europe.
  - Midnight in Paris s'est déroulé à Paris.
  - Midnight in Paris a pour réalisateur Woody Allen.
  - Jean Dujardin a joué dans The Artist.
  - Bérénice Bejo a joué dans The Artist.
  - Bérénice Bejo est du genre féminin.
  - Jean Dujardin est du genre masculin.

- Implicit (common sense) knowledge
  - Paris is a Town, France is a Country, Europe is a Continent
  - Midnight in Paris is the title of a Movie
  - Woody Allen is a person, and a film director
  - Jean Dujardin is an actor and a man …
  - Bérénice Bejo is an actress and a woman
  - The Artist is the title of a movie

Individuals (= instances)



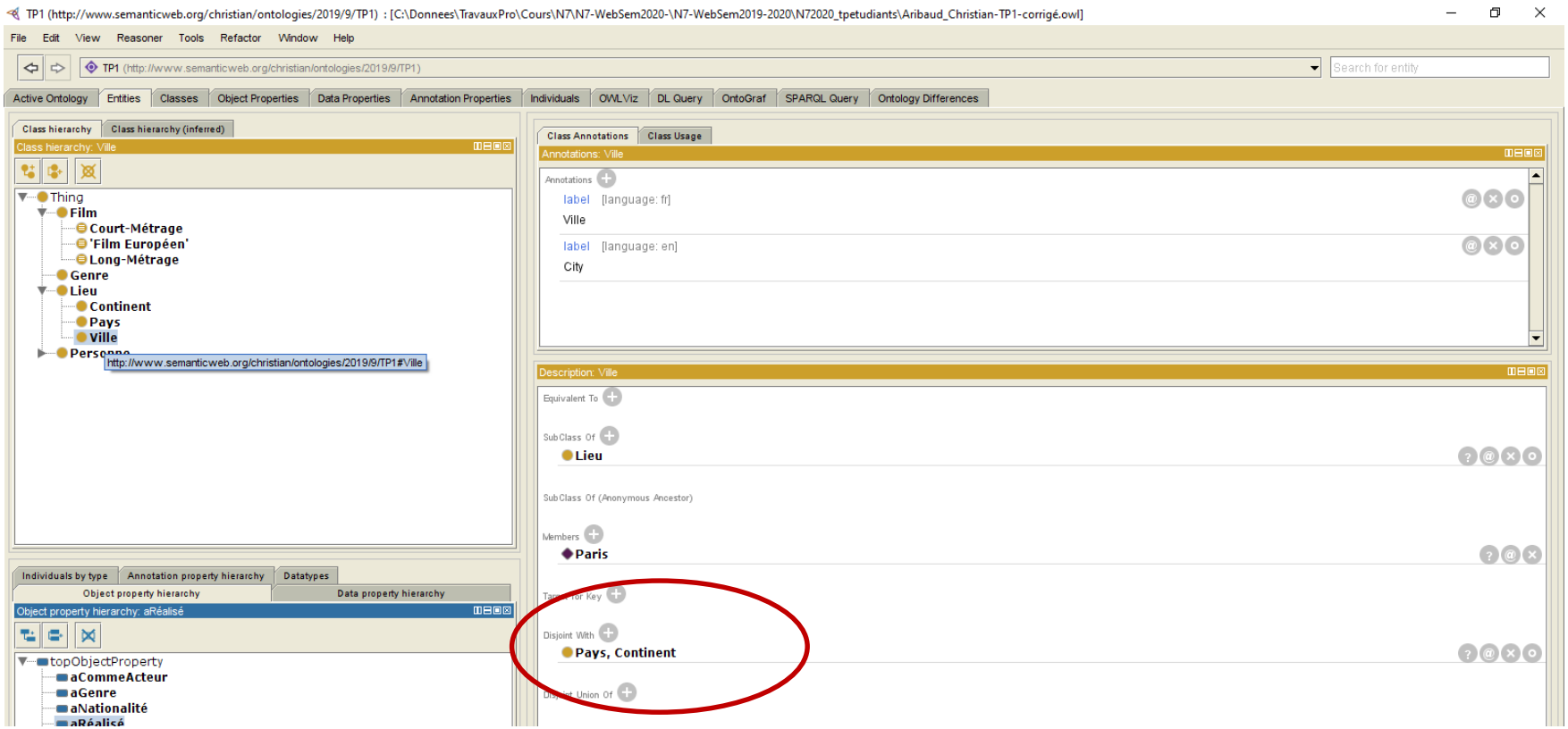
- Synomynie : Movie = film
  - 1 concept, 2 labels
- Polysémie : Monaco
  - 2 concepts : Monaco-ville, Monaco-pays avec le même label (Monaco)
- Redondance :
  - éviter de dire la même chose de plusieurs fois (ou alors c'est pour le vérifier)
  - Si on peut /veut trouver une connaissance en raisonnant , ne pas le dire AUSSI avec des classes

Heavy-weight ontology

- Toute instance de Ville ne peut pas être un Pays.
- Toute instance de Pays ne peut pas être un Continent.
- Un court-métrage est un film ayant une durée en minutes inférieure ou égale à 59 minutes.
- Un long-métrage est un film ayant une durée en minutes supérieure à 59 minutes.
- Un homme est une personne qui a pour genre masculin.

Disjoint classes

- Toute instance de Ville ne peut pas être un Pays.
- Toute instance de Pays ne peut pas être un Continent.



Class description (1)

- The class of all OWL classes owl:Class
  owl:Class rdfs:subClassOf rdfs:Class

- Named class
  <owl:Class rdf:ID="Human"/>

- Enumeration of individuals: extended definition

```
<owl:Class rdf:ID="Continent" > <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Africa"/>
        <owl:Thing rdf:about="#America"/>
        <owl:Thing rdf:about="#Asia"/>
        <owl:Thing rdf:about="#Australia"/>
        <owl:Thing rdf:about="#Antarctica"/>
        <owl:Thing rdf:about="#Europe"/>
    </owl:oneOf>
</owl:Class>
```

$$\forall x \; \text{Continent}(x) \rightarrow (x = \text{Africa}) \vee (x = \text{America}) \vee (x = \text{Asia}) \vee (x = \text{Australia}) \vee (x = \text{Antartica}) \vee (x = \text{Europe})$$

```
<Continent> a owl:Class ;
        owl:oneOf
        ( <Europe> <Africa> <America> <Asia>
        <Australia> <Antartica>) .
```

1. Turtle writing
2. Formal semantics

Class description (2)

- rdfs:subclassOf

```
<owl:Class rdf:ID="Opera">
        <rdfs:subClassOf rdf:resource="#MusicalWork"/>
</owl:Class>
```

- Disjunction of classes

:Country a **owl:Class .**
:Country owl:disjointWith :Continent .

```
<owl:Class rdf:ID="Country">
   <owl:disjointWith rdf:resource="#Continent"/>
</owl:Class>
```

Class description (3)

- Class definition as union of classes
  :MyFavoritePet = owl:unionOf (:cat :rabbit)
  $\forall x$ (MyFavoritePet(x) $\leftrightarrow$ (Cat(x) v Rabbit(x)))


```
<owl:Class rdf:ID=« MyFavoritePet">
    <owl:equivalentClass>
        <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#Cat"/>
            <owl:Class rdf:about="#Rabbit"/>
        </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>

:MyFavoritePet a owl:Class ;
    owl:equivalentClass [
        a owl:Class ;
        owl:unionOf ( :Cat :Rabbit )
    ] .
```

Class description (4)

- Class definition as intersection of classes
  :Man = owl:intersectionOf (:cat :rabbit)
  $\forall x$ (Man(x) $\leftrightarrow$ (Person(x) $\wedge$ Male(x)))

```
<owl:Class rdf:ID= "Man">
      <owl:equivalentClass>
            <owl:Class>
            <owl:intersectonOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Person"/>
                        <owl:Class rdf:about="#Male"/>
            </owl:intersectonOf >
            </owl:Class>
      </owl:equivalentClass>
</owl:Class>

:Man a owl:Class ;
      owl:equivalentClass [
            a owl:Class ;
            owl:intersectionOf ( :Person :Male ) .
      ] .
```

Relations and individuals

- Un homme est une personne qui a pour genre masculin.
- Une femme est une personne qui a pour genre féminin.

- Inheritence
  - Man rdfs:subClassOf person with male gender
- Equivalence
  - All persons with male gender are men
  - All men have male gender

Run the reasonner and test the impact of each option

- Range existence and constraints
  - some // only // exactly i // max i // min i
  - Person hasGender some // only // exactly 1 // max 1 // min 1 Gender
- How did you define hasGender ?
  - rdfs:domain : avec :Person? Thing?
  - rdfs:range Gender? Thing ?

Run the reasonner and test the impact of each option

Retour au TP

- Toute instance de Ville ne peut pas être un Pays.
- Toute instance de Pays ne peut pas être un Continent.

- Un court-métrage est un film ayant une durée en minutes inférieure ou égale à 59 minutes.
- Un long-métrage est un film ayant une durée en minutes supérieure à 59 minutes.

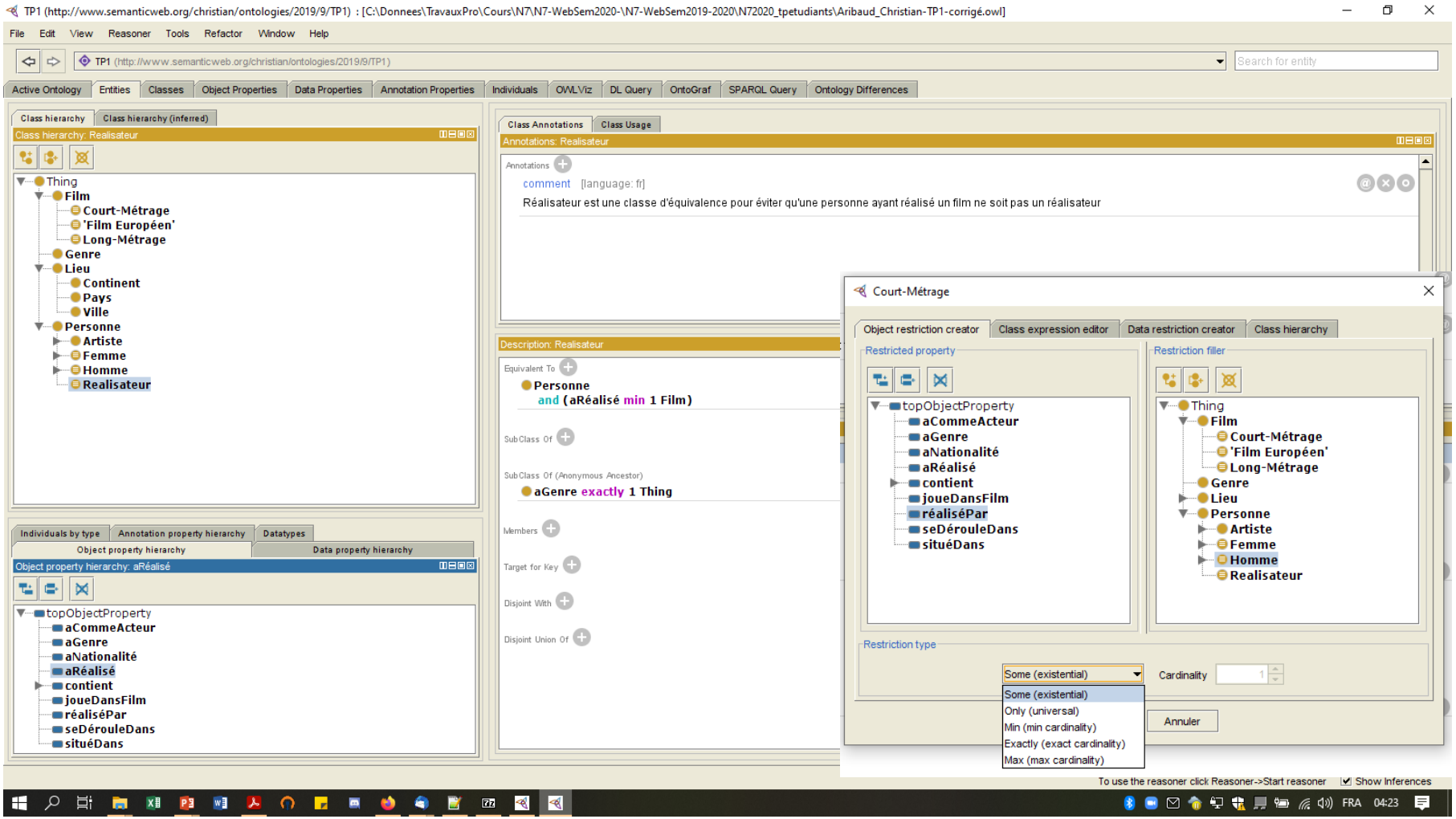
- Un homme est une personne qui a pour genre masculin.
- Une femme est une personne qui a pour genre féminin.
- Un acteur masculin est une personne qui joue dans un film et qui est un homme.
- Un acteur féminin est une personne qui joue dans un film et qui est une femme.

- Toute instance d'un long-métrage ne peut pas être un court-métrage.

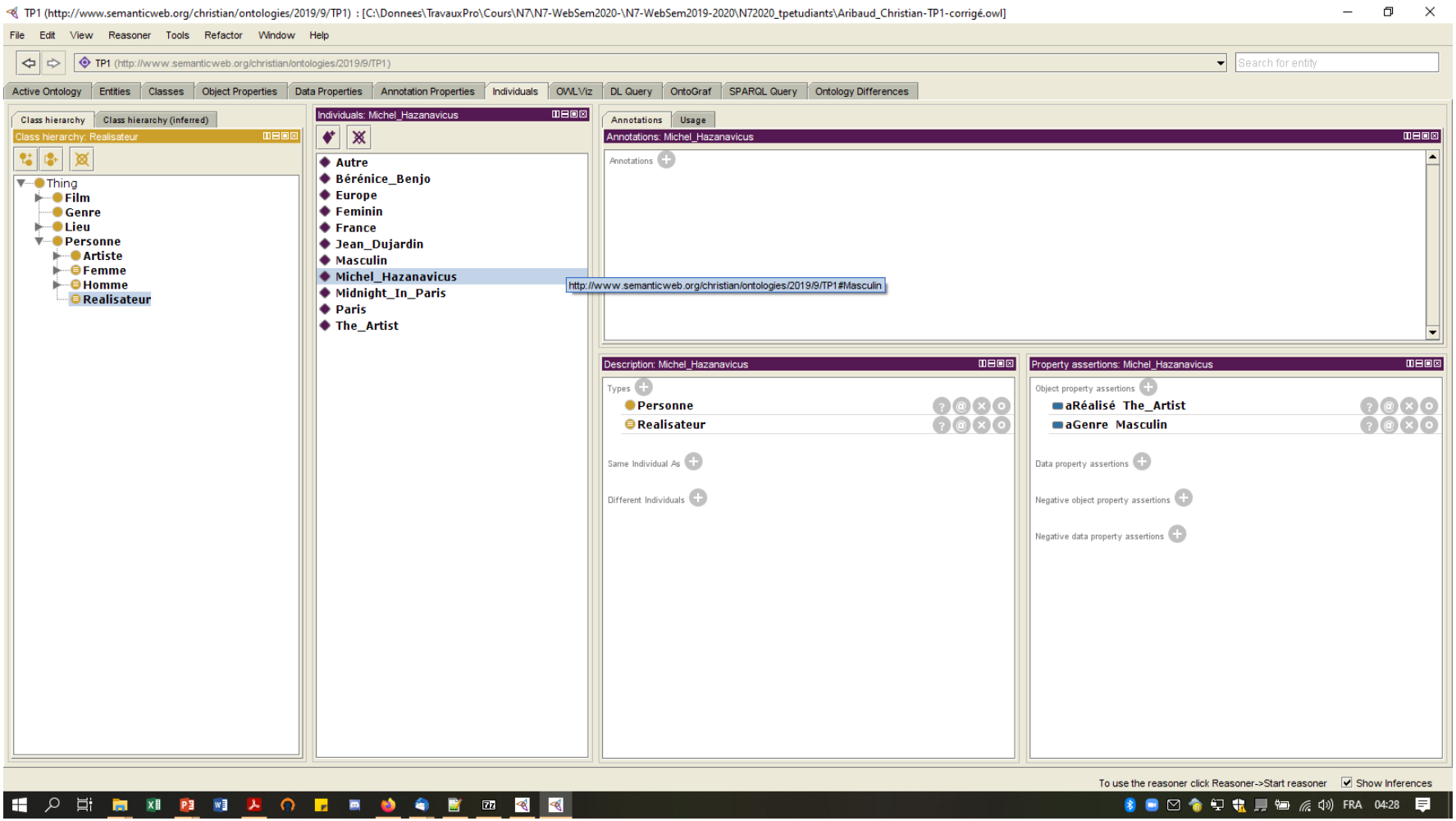
- La propriété « a comme acteur » exprime la relation inverse de « joue dans film ».
- La propriété « contient » exprime la relation inverse de « se situe dans ».

- Si un lieu A est situé dans un lieu B et que ce lieu B est situé dans un lieu C, alors le lieu A est situé dans le lieu C (utilisez les caractéristiques de la relation).
- À tout pays correspond une et une seule capitale (créez la relation et définissez ses caractéristiques).

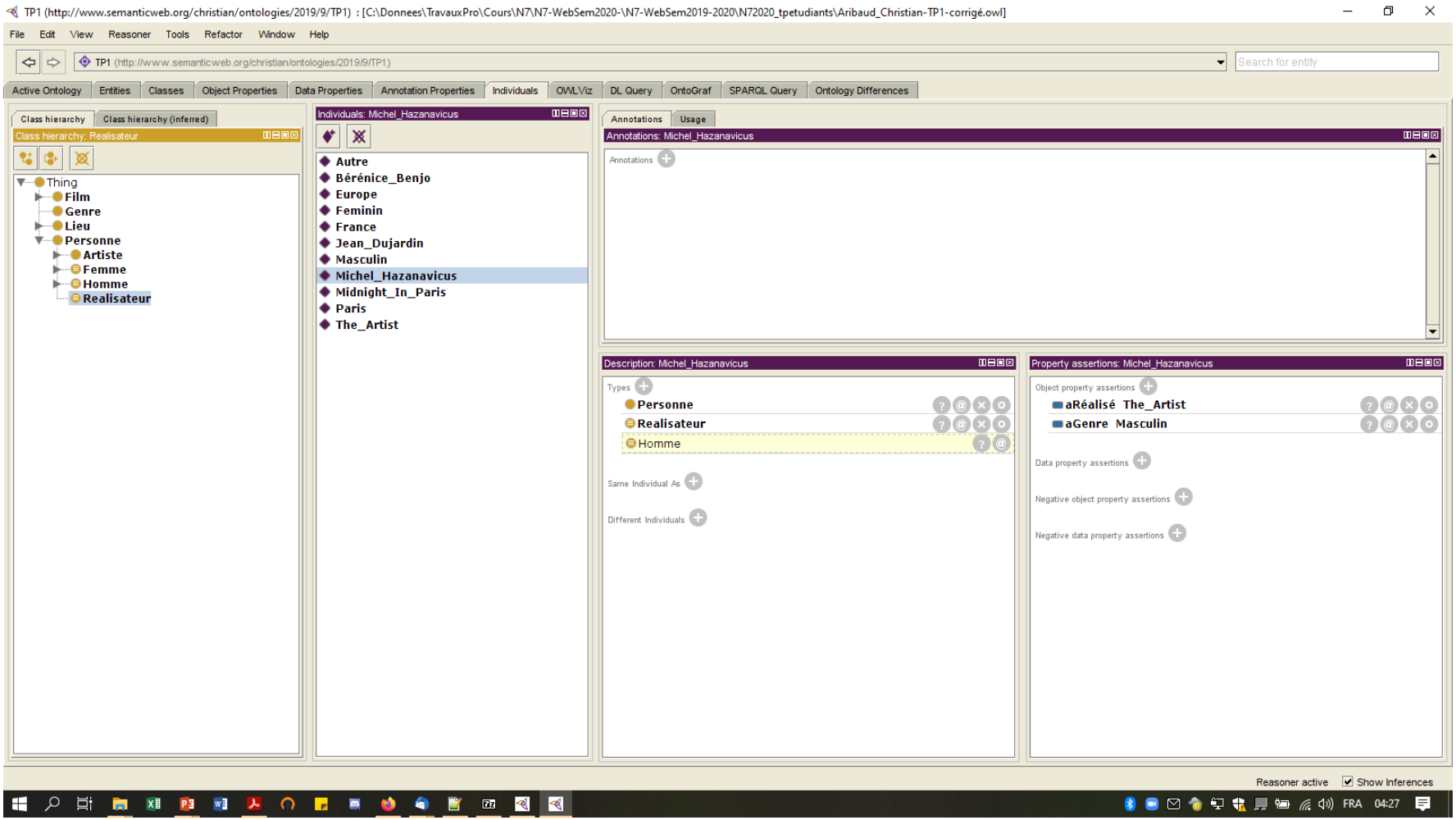
Class properties



Individual properties



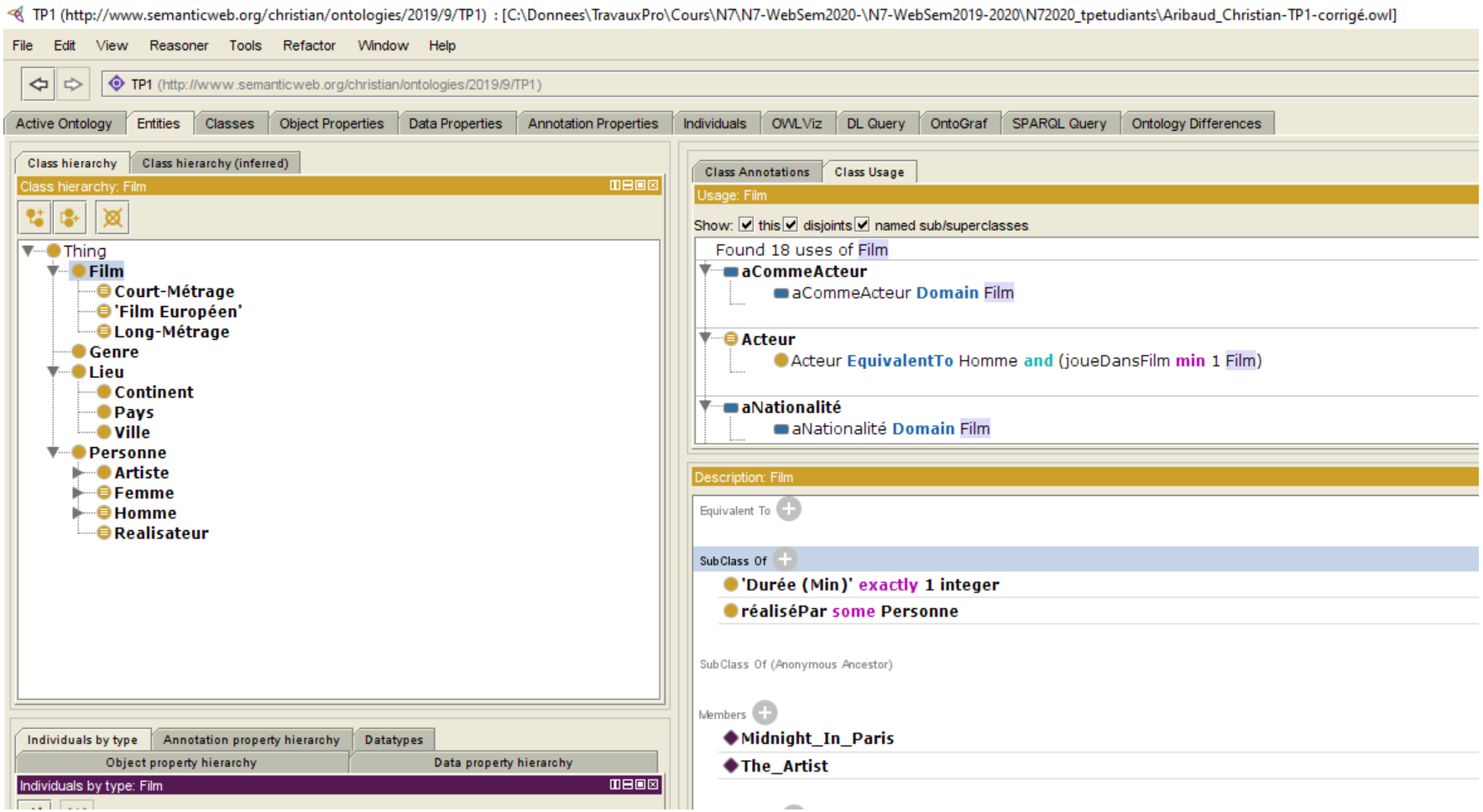
Individual properties after reasoning



Defined classes vs primitive classes

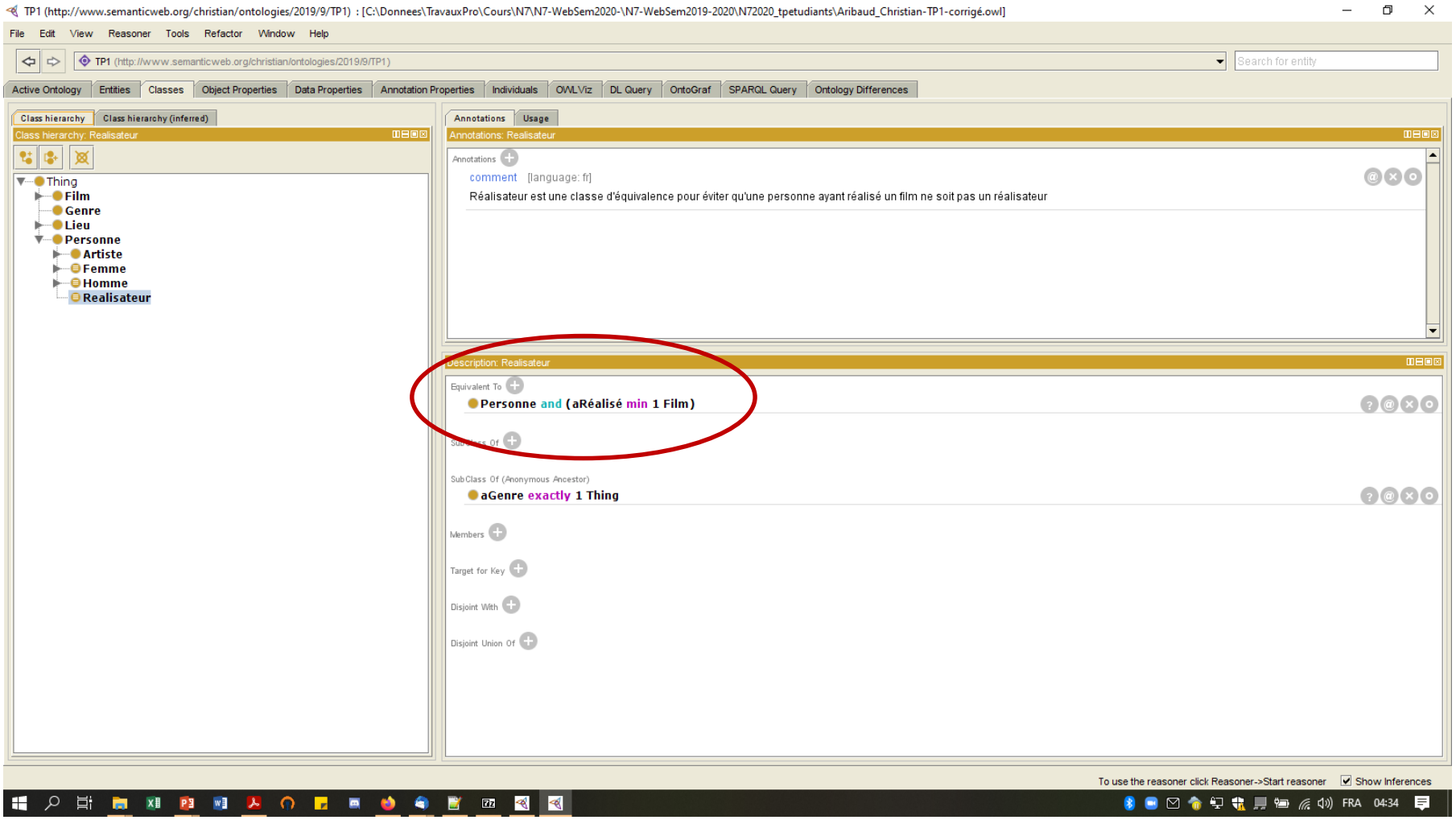
- Un court-métrage est un film ayant une durée en minutes inférieure ou égale à 59 minutes.
- Un long-métrage est un film ayant une durée en minutes supérieure à 59 minutes.
- Un réalisateur est un personne qui réalise au moins un film

- Primitives classes
  - defined as rdfs:subClassOf (in the class hierarchy)
- Defined classes
  - Defined as equivalent to a logic formulas
  - May inherit of a logic formula
  - Not necessarily in the hierarchy

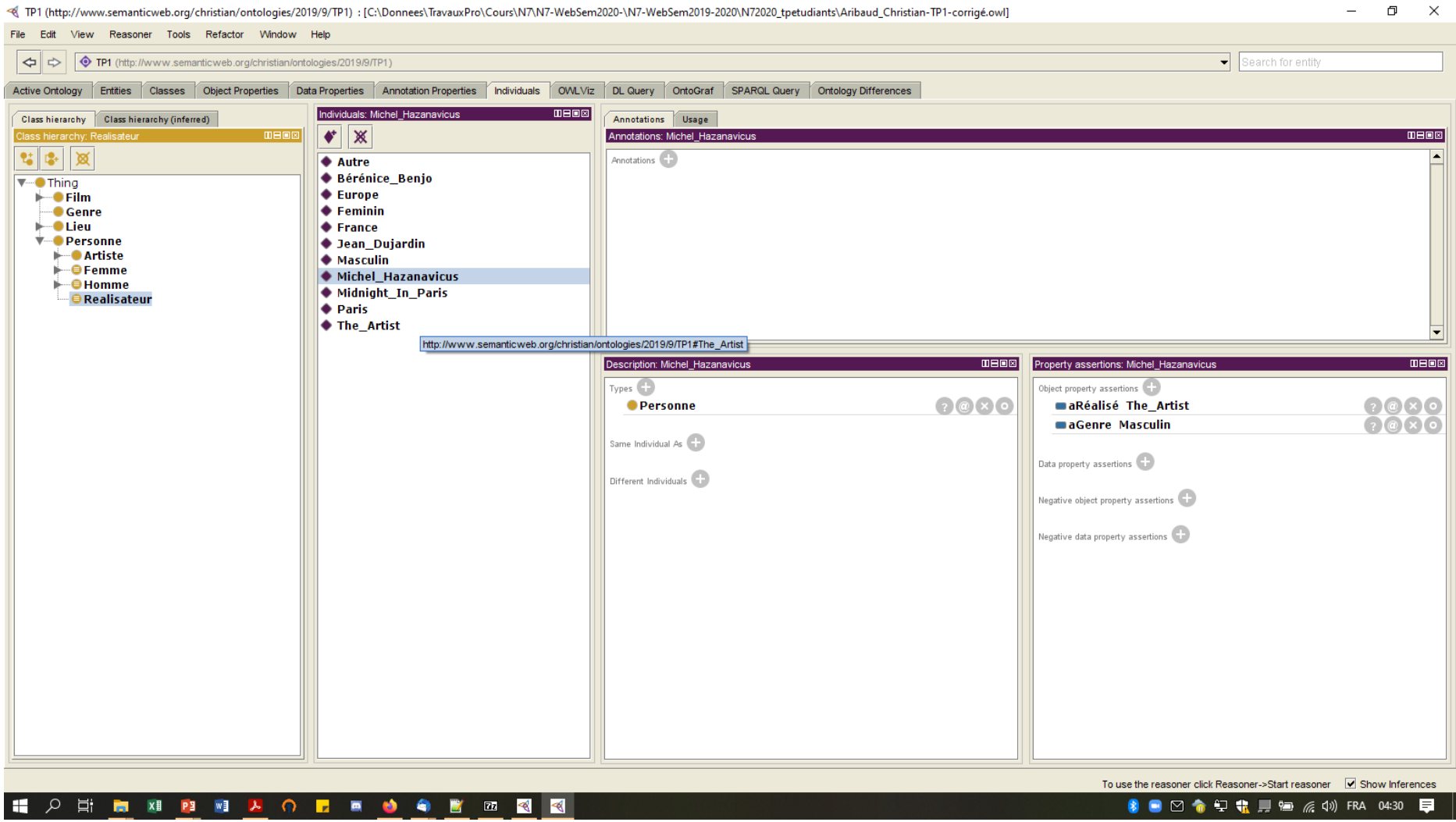

TP1 (http://www.semanticweb.org/christian/ontologies/2019/9/TP1) : [C:\Donnees\TravauxPro\Cours\N7\N7-WebSem2020-\N7-WebSem2019-2020\N72020_tpetudiants\Aribaud_Christian-TP1-corrigé.owl]
File Edit View Reasoner Tools Refactor Window Help
TP1 (http://www.semanticweb.org/christian/ontologies/2019/9/TP1)
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Film
Thing
Film
Court-Métrage
'Film Européen'
Long-Métrage
Genre
Lieu
Continent
Pays
Ville
Personne
Artiste
Femme
Homme
Realisateur
Individuals by type Annotation property hierarchy Datatypes
Object property hierarchy Data property hierarchy
Individuals by type: Film
Class Annotations Class Usage
Usage: Film
Show: this disjoints named sub/superclasses
Found 18 uses of Film
aCommeActeur
aCommeActeur Domain Film
Acteur
Acteur EquivalentTo Homme and (joueDansFilm min 1 Film)
aNationalité
aNationalité Domain Film
Description: Film
Equivalent To
SubClass Of
'Durée (Min)' exactly 1 integer
réaliséPar some Personne
SubClass Of (Anonymous Ancestor)
Members
Midnight_In_Paris
The_Artist

Equivalent class

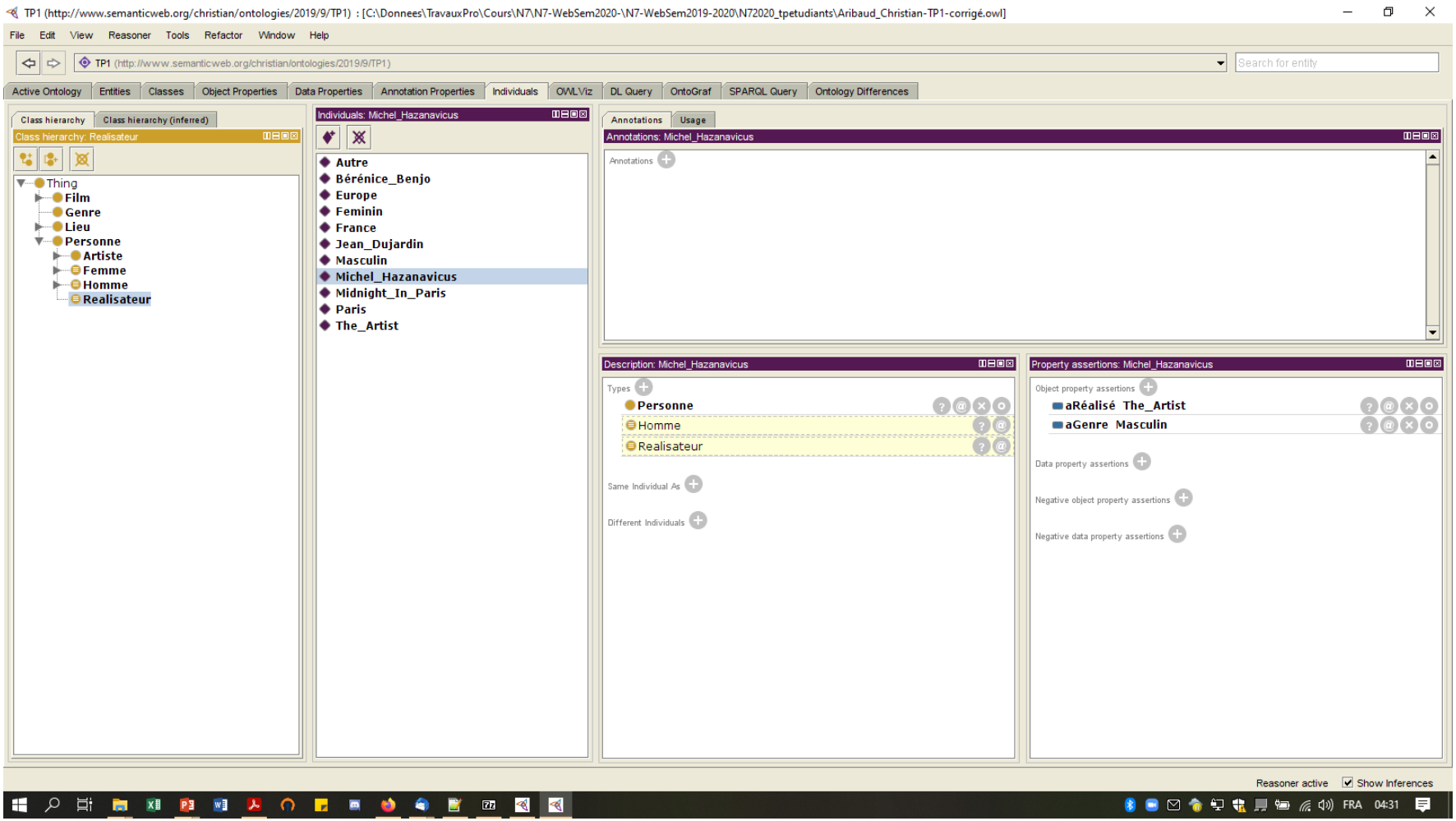
- Un réalisateur est une personne qui réalise au moins un film



Individual types before /after reasoning



Individual types before /after reasoning



Formules logiques pour exprimer des axiomes

- Si un pays a pour capitale une ville, alors ce pays contient cette ville (utilisez la notion de sous-propriété).
- Si un film se déroule dans un lieu et que ce lieu est situé dans un autre lieu, alors le film s'est aussi déroulé dans cet autre lieu (utilisez les chaînes de propriété).
- Un film européen est un film qui a une nationalité associée à un pays d’Europe

Property chain and sub-properties