`

CSE488: Ontologies and the semantic web

Major Task – Movies Ontology

Team 3

Team Members	ID
Alaa Mohamed Mohamed Hamdy	**19P6621**
Engy Ahmed Hassan	**19P4390**
Seifeldin Sameh Mostafa Elkholy	**19P3954**
Osama Saad Farouk	**20P3943**
Yara Mostafa Ibrahim	**19P1120**

Github Link: https://github.com/DedRec/Ontology_Moviefinder_Project

Table of Contents

Table of Figures

1. Problem Statement

The goal of this project is to create an ontology using the Protégé editor that accurately models the domain of movies, including various elements such as personnel (directors, writers, actors), thematic categories (genres), and production details (year, country, language). This project seeks to establish a structured ontology to represent these relationships and attributes in a semantic web framework, utilize restrictions and conditions to define specific rules or characteristics within the ontology, define property types such as transitive, symmetric, or inverseOf where necessary to accurately model relationships between entities, establish domains and ranges for properties to specify their applicability and scope within the ontology and finally introduce additional concepts or properties as needed to enrich the ontology's descriptive power and utility.

2. Modeling the ontology

The ontology consists of several entities, properties and restrictions which are going to be described in details in the following subsections.

Number of entities

1. **Classes:**
    - **Person Class:** Represents individuals involved in the movie industry, such as actors, directors, and writers.
    - **Actor, Director, Writer Classes:** Specialized subclasses of Person, representing individuals with specific roles.
    - **Genre Class:** Represents different categories or genres that movies can belong to.
    - **Movie Class:** Represents a film, which can have various properties such as title, director, actors, genre, etc.
2. **Individuals:**

- Various individuals are instances of the classes defined above. For example, Edgar Wright, Quentin Tarantino, Pulp Fiction, etc.
- Actors, directors, and writers are all individual instances of the Person class.
- Movies like Pulp Fiction, Baby Driver, La La Land, etc., are individual instances of the Movie class.
- Genres like Thriller, Drama, Comedy, etc., are individual instances of the Genre class.

Number of Relations

In this section we provide explanation of the relationships (properties) defined between entities in the ontology.

There are several relationships defined in the ontology:

1. **Object Properties:**
    - **hasActor, hasDirector, hasWriter:** These properties establish relationships between movies and the individuals involved in their creation.
    - **isActorOf, isDirectorOf, isWriterOf:** These properties are inverses of the above properties, establishing relationships from individuals to the movies they are associated with.
2. **Data Properties:**
    - Properties like **hasAge, hasGender, hasNationality** establish relationships between individuals and their attributes.
    - Properties like **hasCountry, hasLanguage, hasTitle, hasYear** establish relationships between movies and their attributes.
3. **Restrictions:**
    - The definition of the Movie class includes restrictions stating that each movie must have at least one director, one writer, and one actor.

- It also have relations and their inverses such as isActorOf and hasActor, isDirectorOf and hasDirector, isWriterOf and hasWriter.
- These restrictions ensure that movies are properly defined with essential personnel involved.

Overall, the ontology provides a structured representation of individuals involved in the movie industry, their roles, and the movies they are associated with, along with attributes such as age, gender, nationality, etc.

3. Logic

In this section , we provide a description of the logic used to define the ontology, including axioms, restrictions, and inference rules.
The ontology is defined using Description Logic (DL), which is a formal knowledge representation language used to define classes, properties, and relationships between entities. Here's an overview of the logic used to define the ontology:

1. **Axioms:**
    - **Class Axioms:** These axioms define the classes in the ontology and their relationships. For example, defining classes like Person, Actor, Director, Writer, Genre, and Movie.
    - **Property Axioms:** These axioms define the object and data properties used in the ontology, such as hasActor, hasDirector, hasWriter, hasAge, hasGender, etc.
    - **Individual Axioms:** These axioms define individual instances of classes, such as specific actors, directors, writers, movies, and genres.
2. **Restrictions:**
    - **Minimum Cardinality Restrictions:** The ontology includes restrictions stating that each movie must have at least one director, one writer, and one actor. This ensures that movies are properly defined with essential personnel involved.

- **Class Equivalence Restrictions:** The Movie class is defined using class equivalence restrictions that specify necessary conditions for a class to be considered a Movie. These conditions include having at least one director, one writer, and one actor.

3. **Inference Rules:**

    - **Inverse Property:** Inverse properties are defined to establish bidirectional relationships between entities. For example, the isActorOf property is defined as the inverse of the hasActor property. This allows inference engines to deduce relationships in both directions.
    - **Class Subsumption:** Subclass relationships between classes are defined to represent hierarchical relationships. For example, Actor, Director, and Writer are subclasses of Person.
    - **Property Domain and Range:** Domain and range axioms are defined for properties to specify the classes of individuals that can participate in these relationships. For example, the domain of the hasActor property is Movie, and its range is Actor.
    - **Transitive Property:** Although not explicitly defined in the provided ontology, transitive properties can be used to infer indirect relationships. For example, if actor A has acted in movie M1, and movie M1 has the same director as movie M2, then actor A can be inferred to have a relationship with the director of movie M2.

Overall, the logic used in the ontology leverages DL constructs such as axioms, restrictions, and inference rules to formalize the representation of entities, properties, and relationships in the domain of movies and their associated personnel.

4. Building & Modelling the Ontology

4.1. RDF Turtle Serialization

```
#prefixes
        @prefix : <http://www.semanticweb.org/owl/owlapi/turtle#> .
        @prefix owl: <http://www.w3.org/2002/07/owl#> .
        @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
        @prefix xml: <http://www.w3.org/XML/1998/namespace> .
        @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
        @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
        @base <http://www.semanticweb.org/owl/owlapi/turtle#> .
        [ rdf:type owl:Ontology
         ] .

        #    Object Properties

        :hasActor rdf:type owl:ObjectProperty ;
                owl:inverseOf :isActorOf ;
                rdfs:domain :Movie ;
                rdfs:range :Actor .

        :hasDirector rdf:type owl:ObjectProperty ;
                  owl:inverseOf :isDirectorOf ;
                  rdfs:domain :Movie ;
                  rdfs:range :Director .

        :hasGenre rdf:type owl:ObjectProperty ;
                rdfs:domain :Movie ;
                rdfs:range :Genre .

        :hasWriter rdf:type owl:ObjectProperty ;
                 owl:inverseOf :isWriterOf ;
                 rdfs:domain :Movie ;
                 rdfs:range :Writer .

        :isActorOf rdf:type owl:ObjectProperty .

        :isDirectorOf rdf:type owl:ObjectProperty .

        :isWriterOf rdf:type owl:ObjectProperty .

        #    Data properties

        :hasAge rdf:type owl:DatatypeProperty ;
               rdfs:domain :Person ;
```

```
        rdfs:range xsd:integer .

:hasCountry rdf:type owl:DatatypeProperty ;
            rdfs:domain :Movie ;
            rdfs:range xsd:string .

:hasGender rdf:type owl:DatatypeProperty ;
           rdfs:domain :Person ;
           rdfs:range xsd:string .

:hasLanguage rdf:type owl:DatatypeProperty ;
             rdfs:domain :Movie ;
             rdfs:range xsd:string .

:hasName rdf:type owl:DatatypeProperty ;
         rdfs:domain :Person ;
         rdfs:range xsd:string .

:hasNationality rdf:type owl:DatatypeProperty ;
                rdfs:domain :Person ;
                rdfs:range xsd:string .

:hasTitle rdf:type owl:DatatypeProperty ;
          rdfs:domain :Movie ;
          rdfs:range xsd:string .

:hasYear rdf:type owl:DatatypeProperty ;
         rdfs:domain :Movie ;
         rdfs:range xsd:integer .

#    Classes

:Actor rdf:type owl:Class ;
       rdfs:subClassOf :Person .

:Director rdf:type owl:Class ;
          rdfs:subClassOf :Person .


:Genre rdf:type owl:Class .
:Movie rdf:type owl:Class ;
       owl:equivalentClass [ rdf:type owl:Restriction ;
                             owl:onProperty :hasActor ;
                             owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
                             owl:onClass :Movie
                           ] ,
```

```
                    [ rdf:type owl:Restriction ;
                      owl:onProperty :hasDirector ;
                      owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
                      owl:onClass :Movie
                    ] ,
                    [ rdf:type owl:Restriction ;
                      owl:onProperty :hasWriter ;
                      owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
                      owl:onClass :Movie
                    ] .

:Person rdf:type owl:Class .

:Writer rdf:type owl:Class ;
       rdfs:subClassOf :Person .

#    Individuals

<#Action> rdf:type owl:NamedIndividual ,
                            :Genre .

<#Baby_Driver> rdf:type owl:NamedIndividual ,
           :Movie ;
                        :hasActor < #Edgar_Wright> ;
                        :hasDirector <#Edgar_Wright> ;
                        :hasWriter < #Edgar_Wright> ;
                        :rdfs:label "Baby Driver" .

<#Boogie_Nights> rdf:type owl:NamedIndividual ,
                                  :Movie ;
                            :hasActor < #Paul_Thomas_Anderson> ;
                            :hasDirector < #Paul_Thomas_Anderson> ;
                            :hasWriter < #Paul_Thomas_Anderson> ;
                            :rdfs:label "Boogie Nights" .

 <#Brittany_Kaiser> rdf:type owl:NamedIndividual ,
                                    :Actor ,
                                    :Person ;
                          :isActorOf <#Great_Hack> ;
                        rdfs:label "Brittany Kaiser" .

<#Carole_Cadwalladr> rdf:type owl:NamedIndividual ,
                                       :Actor ,
                                       :Person ;
                                 :isActorOf <#The_Great_Hack> ;
                            rdfs:label "Carole Cadwalladr" .
```

```
<#Comedy> rdf:type owl:NamedIndividual ,
                    :Genre .


<#Crime> rdf:type owl:NamedIndividual ,
                    :Genre .
<#Damien_Chazelle> rdf:type owl:NamedIndividual ,
                    :Director ,
                    :Person ,
                    :Writer ;
                    :isDirectorOf <#La_La_Land> ;
                    :isWriterOf <#La_La_Land> ;
                    rdfs:label "Damien Chazelle" .


<#David_Carroll> rdf:type owl:NamedIndividual ,
                    :Actor ,
                    :Person ;
                    :isActorOf <#The_Great_Hack> ;
                    rdfs:label "David Carroll" .


<#Drama> rdf:type owl:NamedIndividual ,
                    :Genre .

<#Edgar_Wright> rdf:type owl:NamedIndividual ,
                    :Actor ,
                    :Director ,
                    :Writer ;
                    :isActorOf <#Kill_Bill_vol1> ,
                    <#Pulp_Fiction> ;
                    :hasGender "Male" ;
                    :rdfs:label "Edgar Wright" .


<#Erin_Barnett> rdf:type owl:NamedIndividual ,
                    :Person ,
                    :Writer ;
                    :isWriterOf <#The_Great_Hack> ;
                    rdfs:label "Erin Barnett" .


<#Jehane_Noujaim> rdf:type owl:NamedIndividual ,
                    :Director ,
                    :Person ;
                    :isDirectorOf <#The_Great_Hack> ;
```

```
                    rdfs:label "Jehane Noujaim" .

< #John_Travolta> rdf:type owl:NamedIndividual ,
                                :Actor ;
                        :isActorOf <#Pulp_Fiction> ;
                        :hasGender "Male" ;
                         rdfs:label "John Travolta" .

<#Jojo_Rabbit> rdf:type owl:NamedIndividual ,
                              :Movie ;
                    :hasActor <#Taika_Waititi> ;
                    :hasDirector <#Taika_Waititi> ;
                     rdfs:label "Jojo Rabbit" .

<#Karim_Amer> rdf:type owl:NamedIndividual ,
                              :Director ,
                              :Person ,
                              :Writer ;
                  :isDirectorOf <#The_Great_Hack> ;
                  :isWriterOf <#The_Great_Hack> ;
            rdfs:label "Karim Amer" .


< #Kill_Bill_vol1> rdf:type owl:NamedIndividual ,
                               :Movie ;
                     :hasActor <#Quentin_Tarantino> ;
                     :hasDirector <#Quentin_Tarantino> ;
                     :hasWriter <#Quentin_Tarantino> ,
                                 <#Uma_Thurman> ;
                rdfs:label "Kill Bill vol1" .

<#La_La_Land> rdf:type owl:NamedIndividual ,
                              :Movie ;
                    :hasDirector <#Damien_Chazelle> ;
                    :hasWriter <#Damien_Chazelle> ;
              rdfs:label "La La Land" .

<#Paul_Thomas_Anderson> rdf:type owl:NamedIndividual ,
                                        :Actor ,
                                        :Director ,
                                        :Writer ;
                              :isActorOf <#Boogie_Nights> ,
                                          <#There_Will_Be_Blood> ;
                              :isDirectorOf <#Boogie_Nights> ,
                                             <#There_Will_Be_Blood> ;
                              :isWriterOf <#Boogie_Nights> ,
```

```
                                                <#There_Will_Be_Blood> ;
                                        :hasAge 51 ;
                                        :hasGender "Male" ;
                                        :hasNationality "American" ;
                                      rdfs:label "Paul Thomas Anderson" .

<#Pedro_Kos> rdf:type owl:NamedIndividual ,
                                :Person ,
                                :Writer ;
                        :isWriterOf <http://www.co-
ode.org/ontologies/ont.owl#The_Great_Hack> ;
                    rdfs:label "Pedro Kos" .

<#Pulp_Fiction> rdf:type owl:NamedIndividual ,
                                  :Movie ;
                          :hasActor <#John_Travolta> ,
                                    <#Quentin_Tarantino> ,
                                    <#Uma_Thurman> ;
                          :hasDirector <#Quentin_Tarantino> ;
                          :hasGenre <#Crime> ,
                                    <#Thriller> ;
                             :hasWriter <#Quentin_Tarantino> ;
                               :hasCountry "USA" ;
                               :hasLanguage "English" ;
                               :hasYear 1994 ;
                           rdfs:label "Pulp Fiction" .

<Quentin_Tarantino> rdf:type owl:NamedIndividual ,
                                        :Actor ,
                                        :Director ,
                                        :Writer ;
                                 :isActorOf <#Kill_Bill_vol1> ,
                                            <#Pulp_Fiction> ;
                                 :isDirectorOf <#Kill_Bill_vol1> ,
                                               <#Pulp_Fiction> ;
                                 :isWriterOf <#Kill_Bill_vol1> ,
                                             <#Pulp_Fiction> ;
                                 :hasGender "Male" ;
                              rdfs:label "Quentin Tarantino" .


< #Shaun_of_the_Dead> rdf:type owl:NamedIndividual ,
                                          :Movie ;
                                  :hasActor <#Edgar_Wright> ;
                                  :hasDirector <#Edgar_Wright> ;
                                  :hasWriter <#Edgar_Wright> ;
```

```
                                    rdfs:label "Shaun of the Dead" .

<#Taika_Waititi> rdf:type owl:NamedIndividual ,
                                    :Actor ,
                                    :Director ;
                              :isActorOf <#Jojo_Rabbit> ,
                                          <#Thor:_Ragnarok> ;
                              :isDirectorOf <#Jojo_Rabbit> ,
                                            <#Thor:_Ragnarok> ;
                        rdfs:label "Taika Waititi" .

<#The_Great_Hack> rdf:type owl:NamedIndividual ,
                                            :Movie ;
                                 :hasActor <#Brittany_Kaiser> ,
                                           <#Carole_Cadwalladr> ,
                                           <#David_Carroll> ;
                                 :hasYear 2019 ;
                              rdfs:label "The Great Hack" .

<#There_Will_Be_Blood> rdf:type owl:NamedIndividual ,
                                                :Movie ;
                                       :hasActor <#Paul_Thomas_Anderson> ;
                                       :hasDirector <#Paul_Thomas_Anderson> ;
                                       :hasWriter <#Paul_Thomas_Anderson> ;
                                  rdfs:label "There will be blood" .

<#Thriller> rdf:type owl:NamedIndividual ,
                                 :Genre .
<#Uma_Thurman> rdf:type owl:NamedIndividual ,
                                          :Actor ,
                                          :Writer ;
                                 :isActorOf <#Pulp_Fiction> ;
                                 :isWriterOf <#Kill_Bill_vol1> ;
                                 :hasGender "Female" ;
                            rdfs:label "Uma Thurman" .
<#Thor:_Ragnarok> rdf:type owl:NamedIndividual ,
                                           :Movie ;
                                  :hasActor <#Taika_Waititi> ;
                                  :hasDirector <#Taika_Waititi> ;
                             rdfs:label "Thor".
```

This ontology defines a conceptual model for describing movies, genres, and individuals involved in the film industry. It establishes relationships between movies and their actors, directors, and writers through object properties like **hasActor**, **hasDirector**, and **hasWriter**. Additionally, it captures attributes of individuals such as their age, gender, and nationality using data properties like **hasAge** and **hasNationality**. Genres are also represented as a class, and movies are categorized into these genres. Each movie is characterized by properties like title, year of release, country, and language.

4.2. Pellet Consistency Check

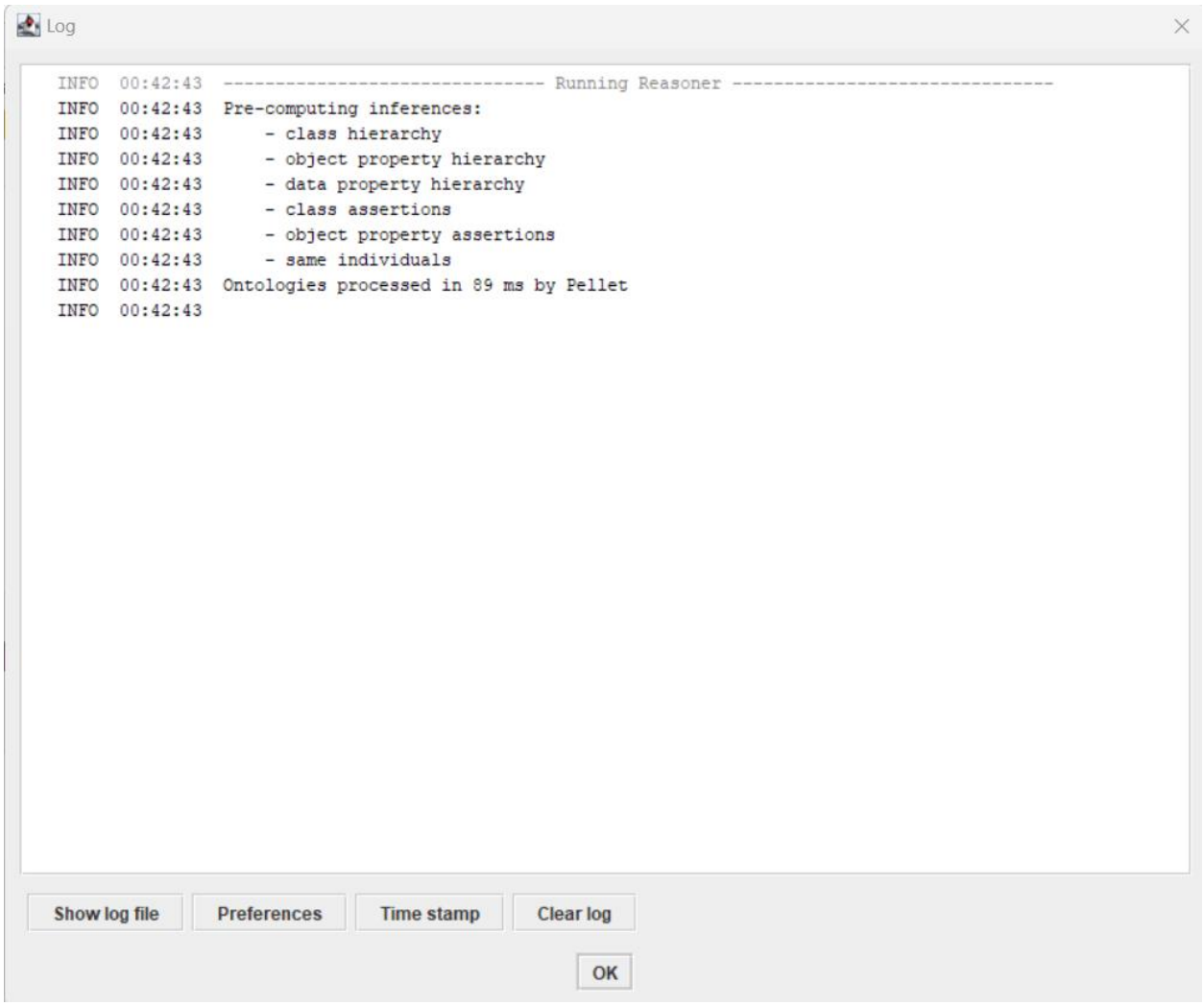


Figure 1 PELLET consistency check

As shown in the screenshot above, the ontology passed the pellet reasoner consistency check with no errors.

4.3. Ontology Visualization

4.3.1 Asserted Graph

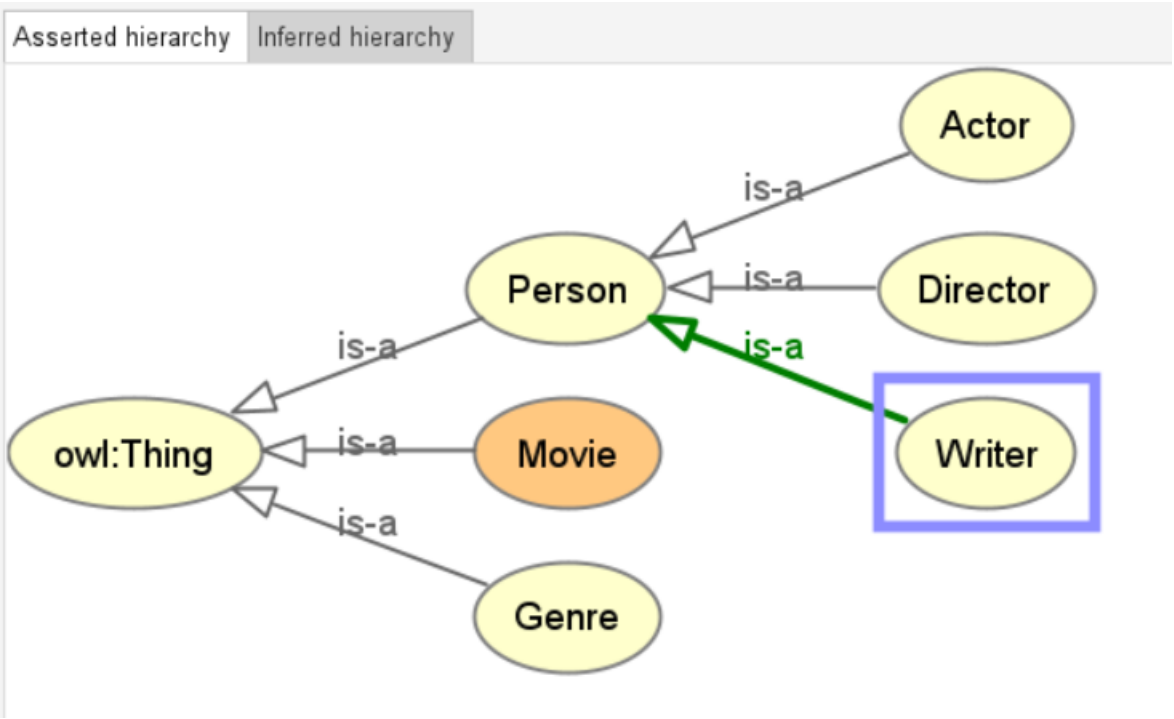


Figure 2 Ontology Asserted Graph

4.3.2 Inferred Graph

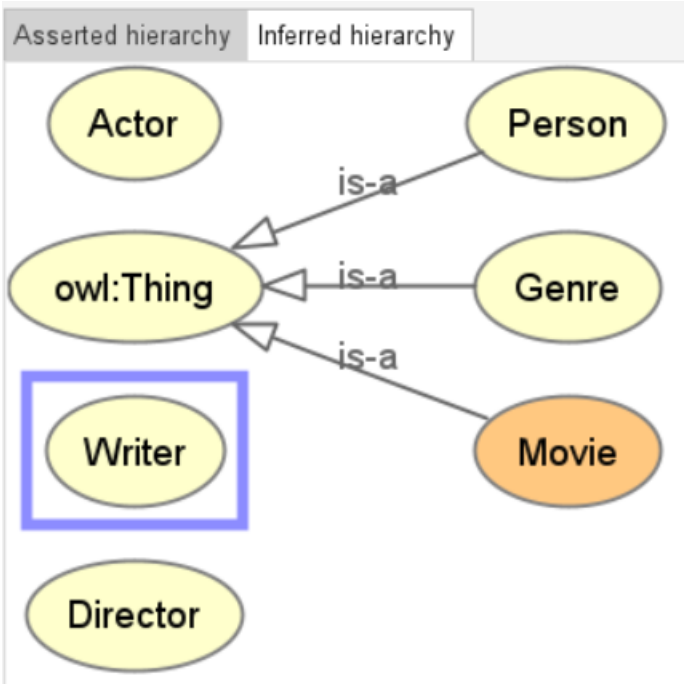


Figure 3 Ontology Inferred Graph

4.3.3 Data Flow Diagram

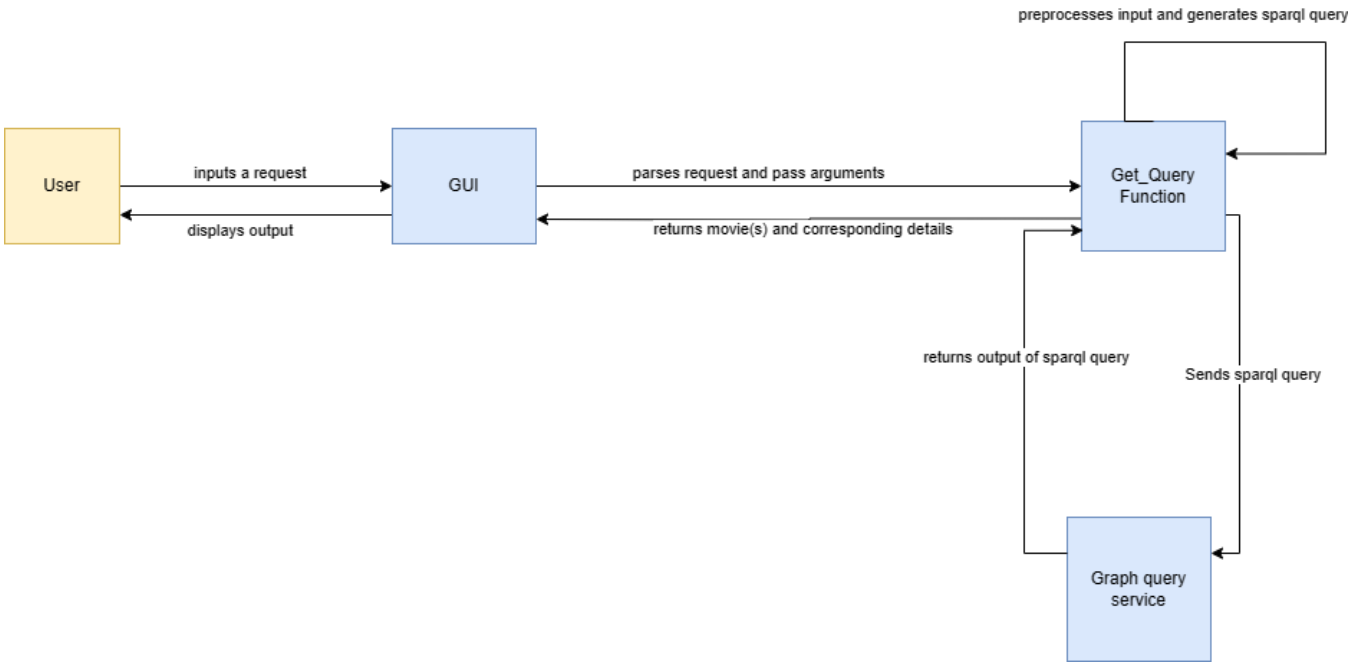


Figure 4 Data Flow Diagram

5. Populating the ontology

We populated the ontology using the examples given in the project description as well as the below individuals:

Figure 5 Individuals of Person Class

Figure 6 Individuals of Movie Class

Figure 7 Individuals of Genre Class

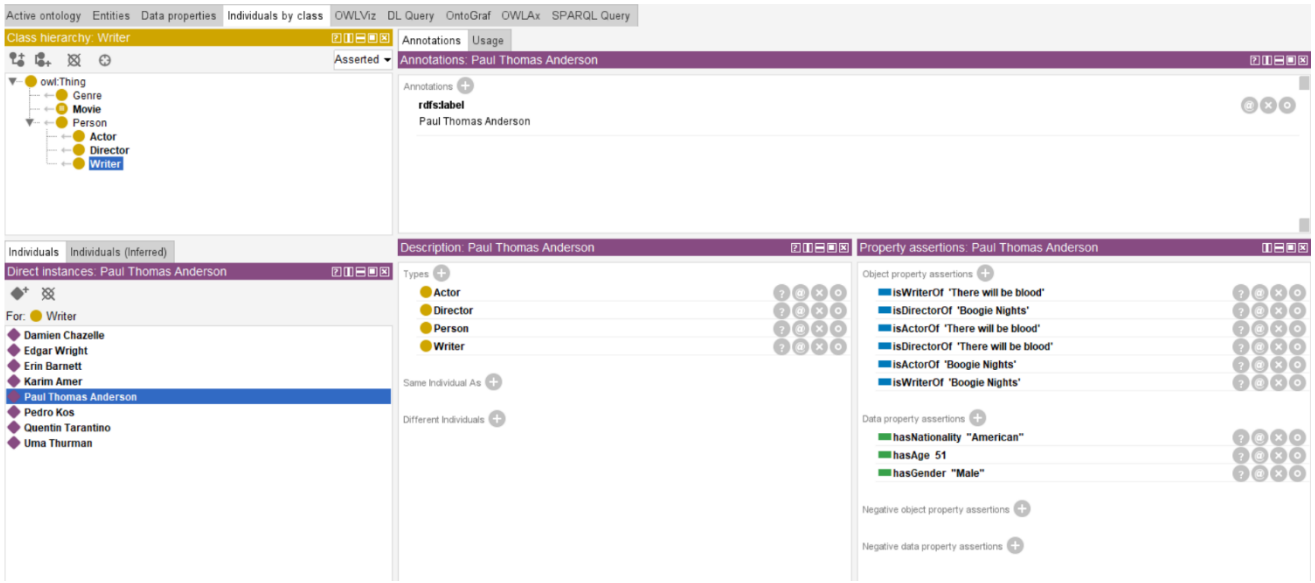
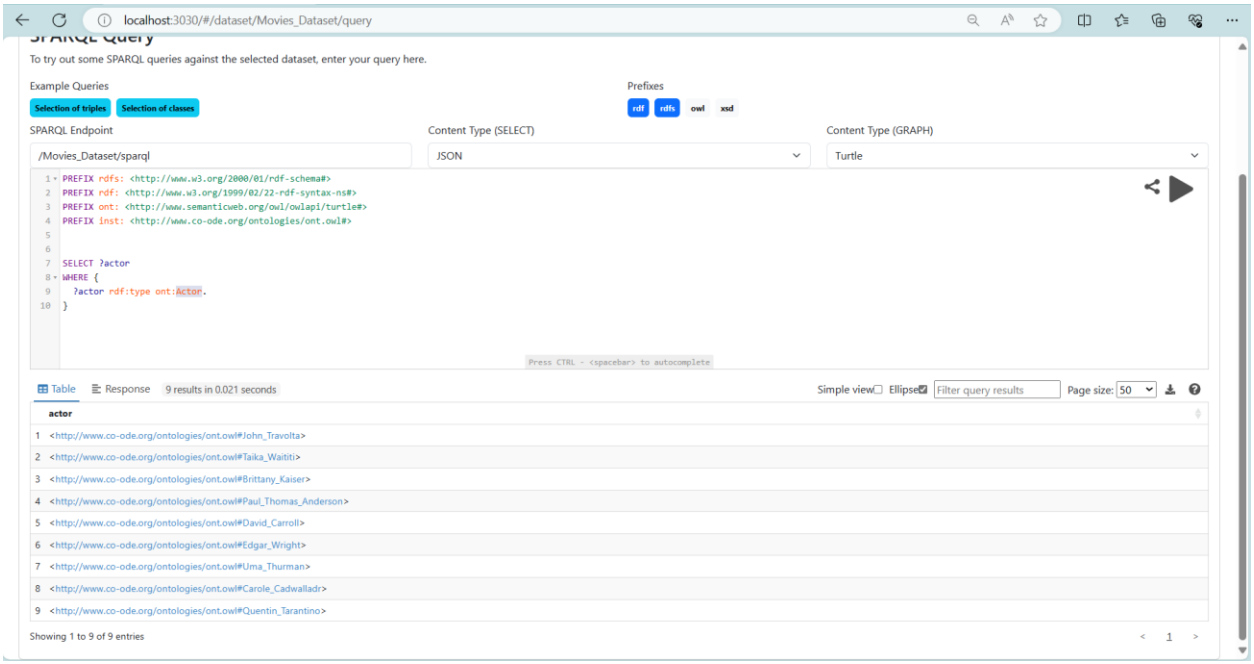


Figure 8 Screenshots of properties in Protégé

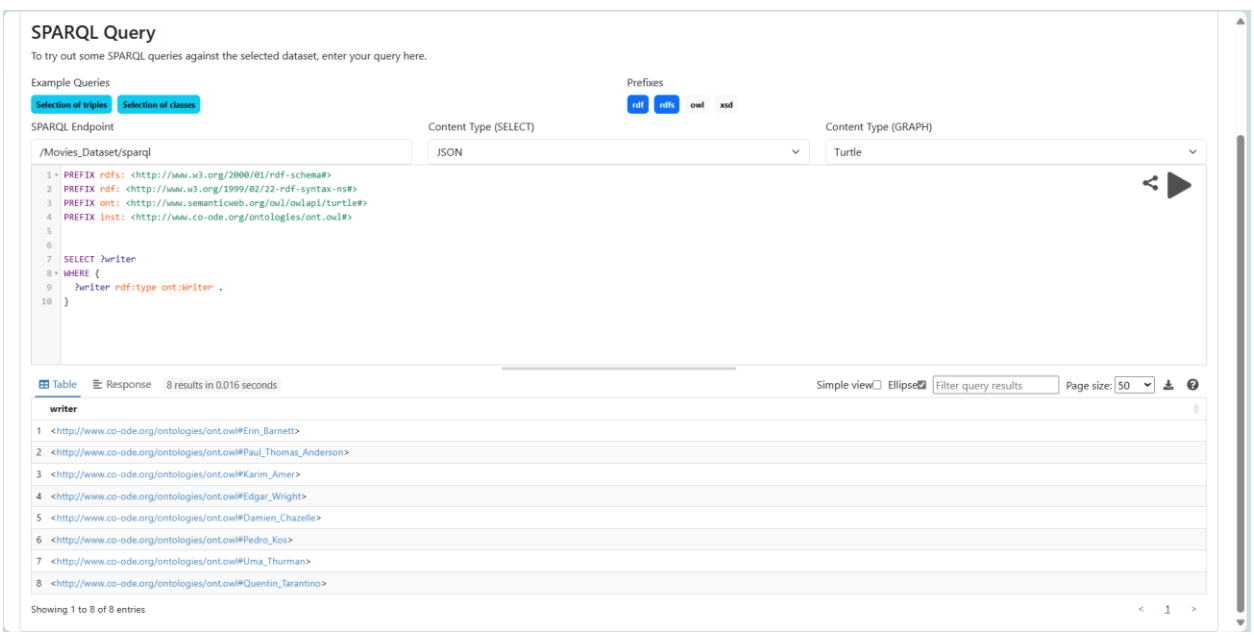
6. Querying the ontology using SPARQL

For this part we used the apache Jena endpoint for querying the queries below:

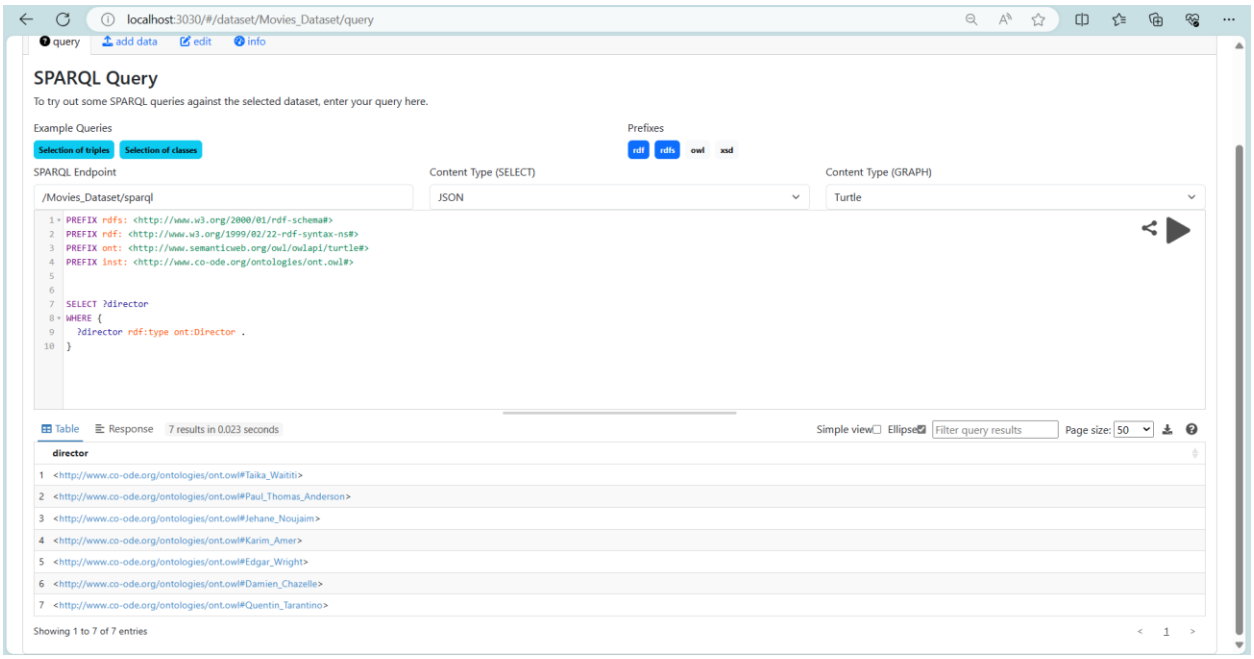
1. *List the instances of the class Actor*



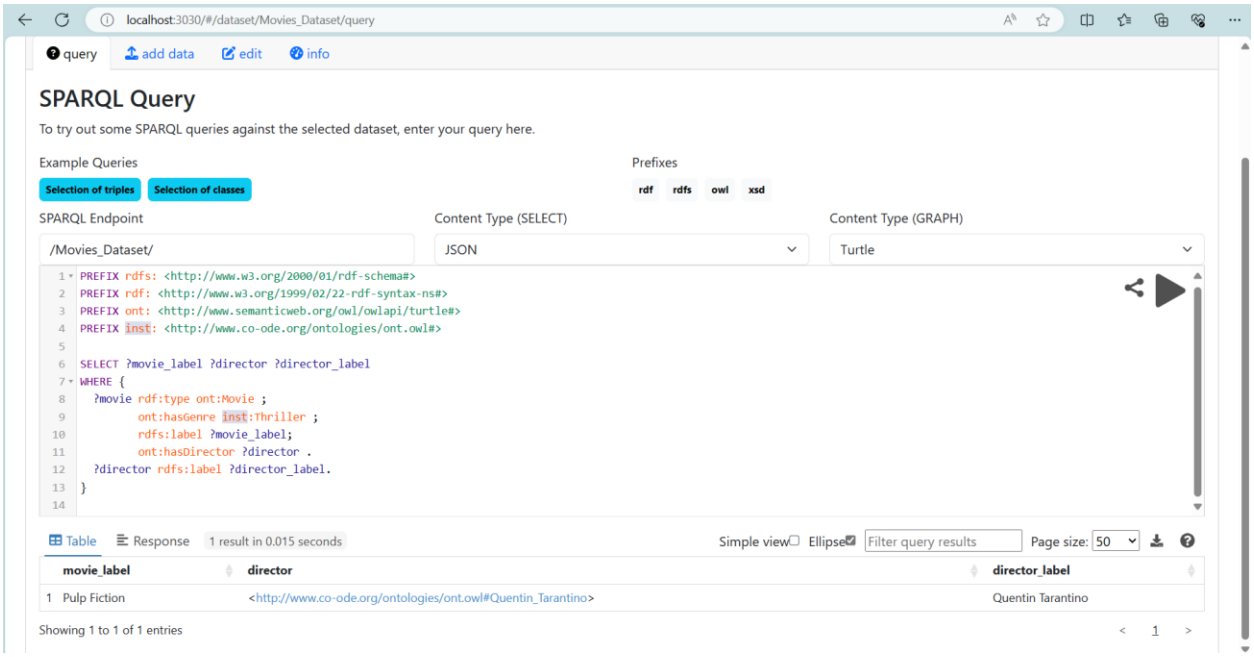
2. *List the instances of the class writer*



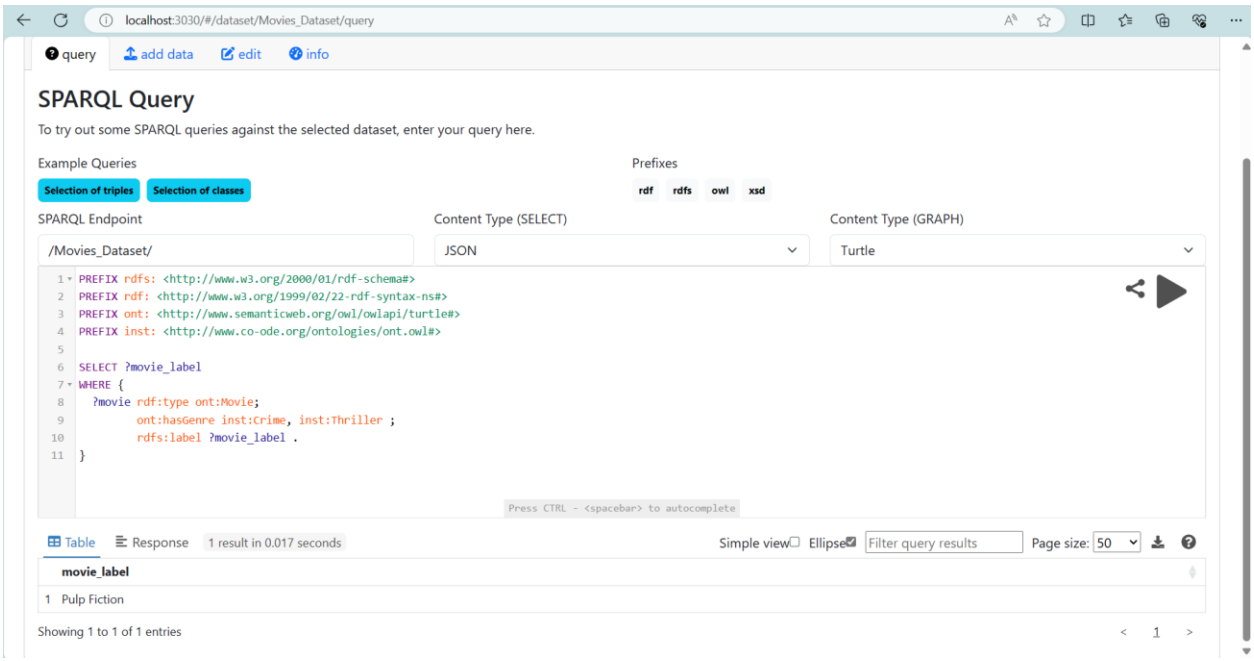
3. *List the instances of the class director*



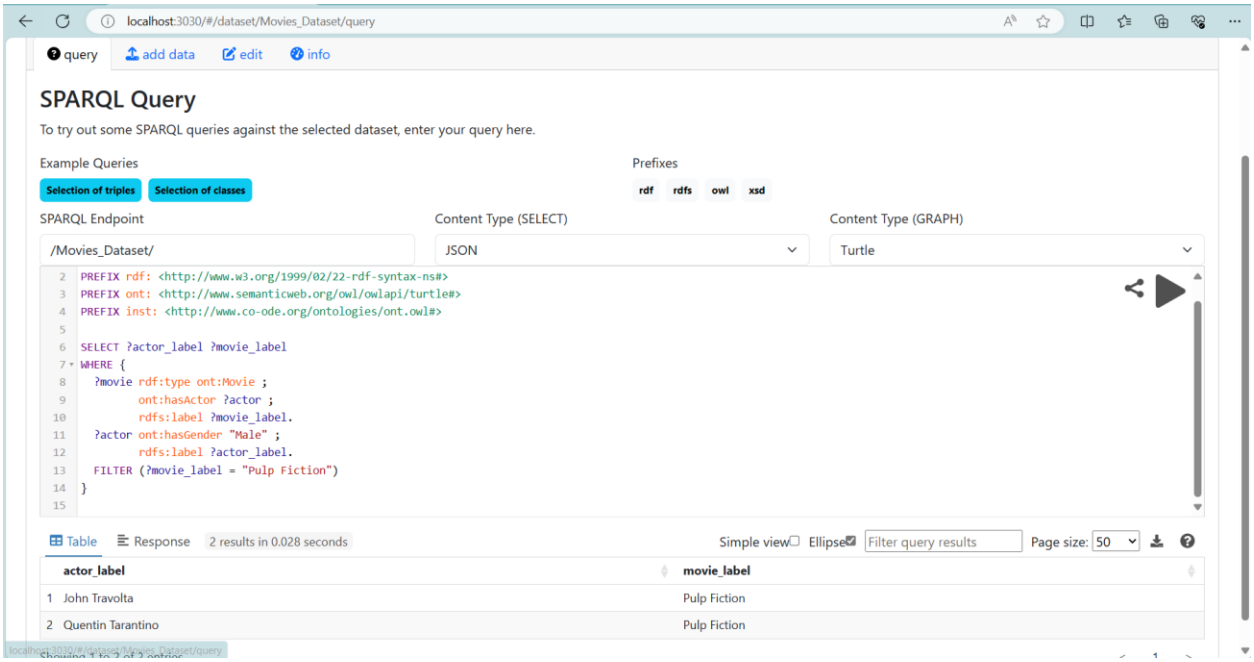
4. *List the name of all Thriller movies. For each one, display its director.*



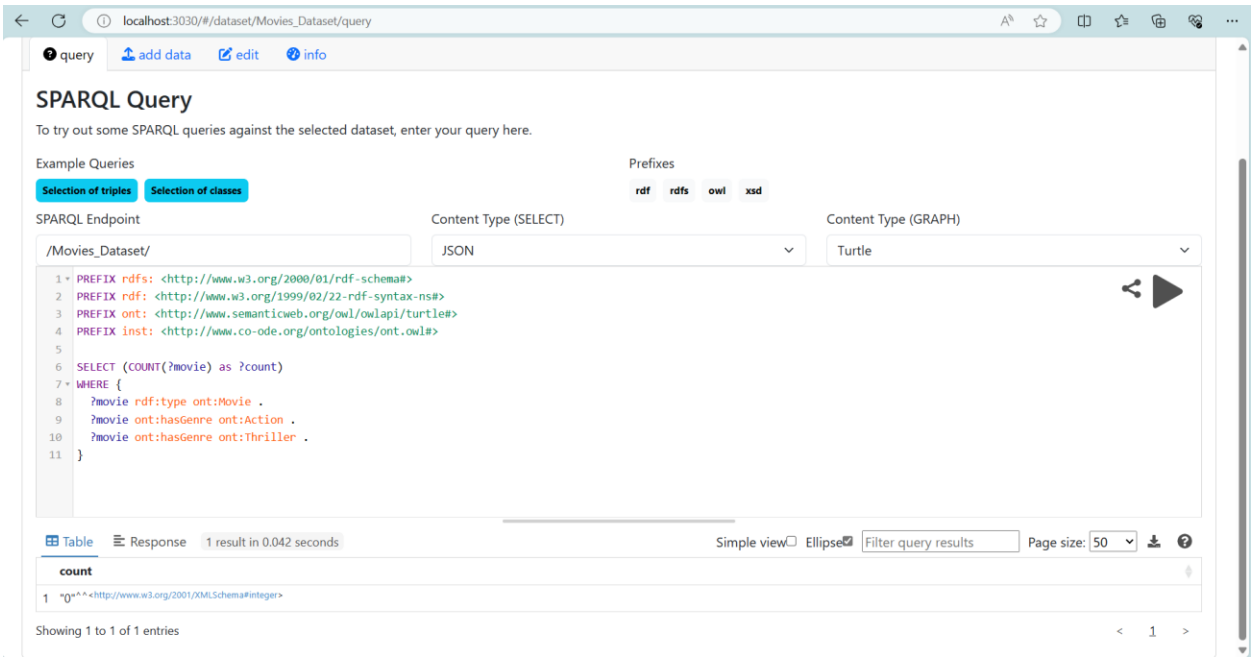
5. *List the name of all Crime Thriller movies.*



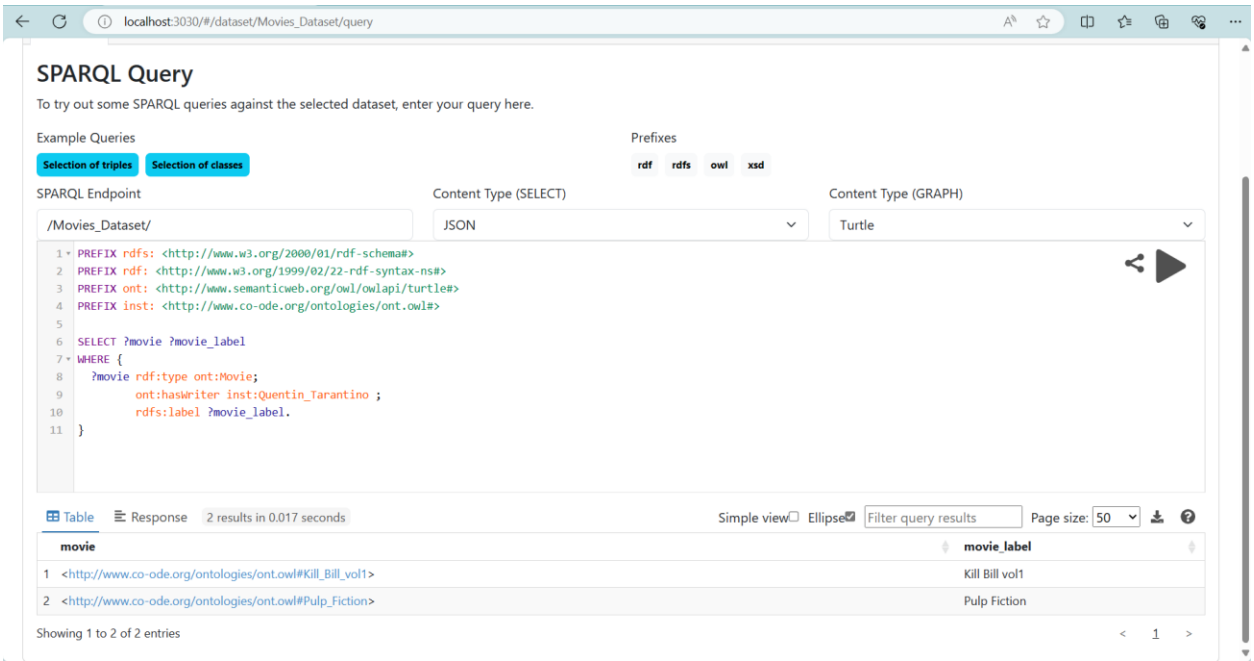
6. *list the male actors in the movie in specific film*



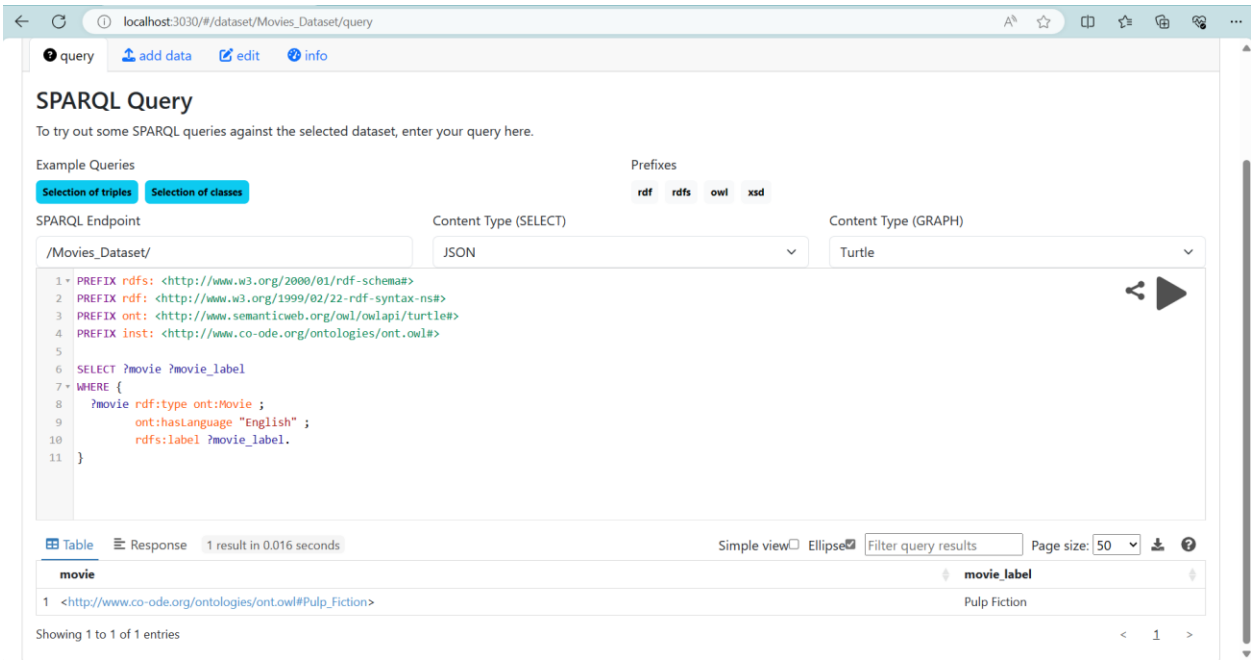
7. *How many movies have both "Action" and "Thriller" as genres?*



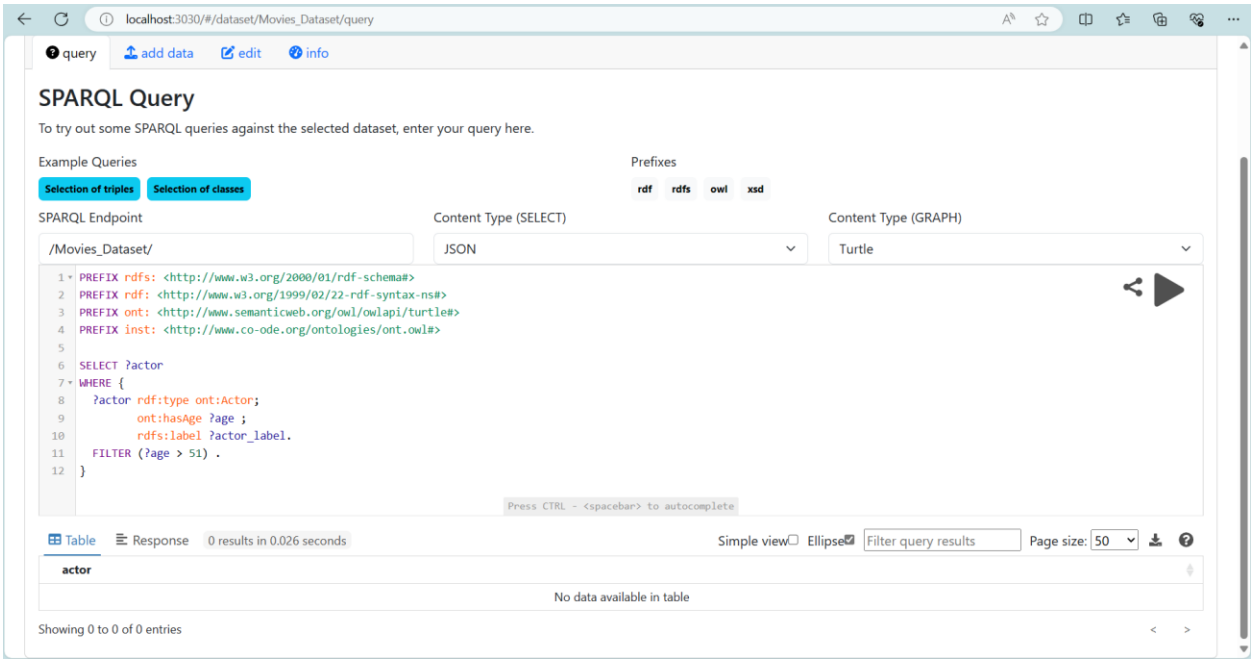
8. *List all the movies written by a specific writer.*



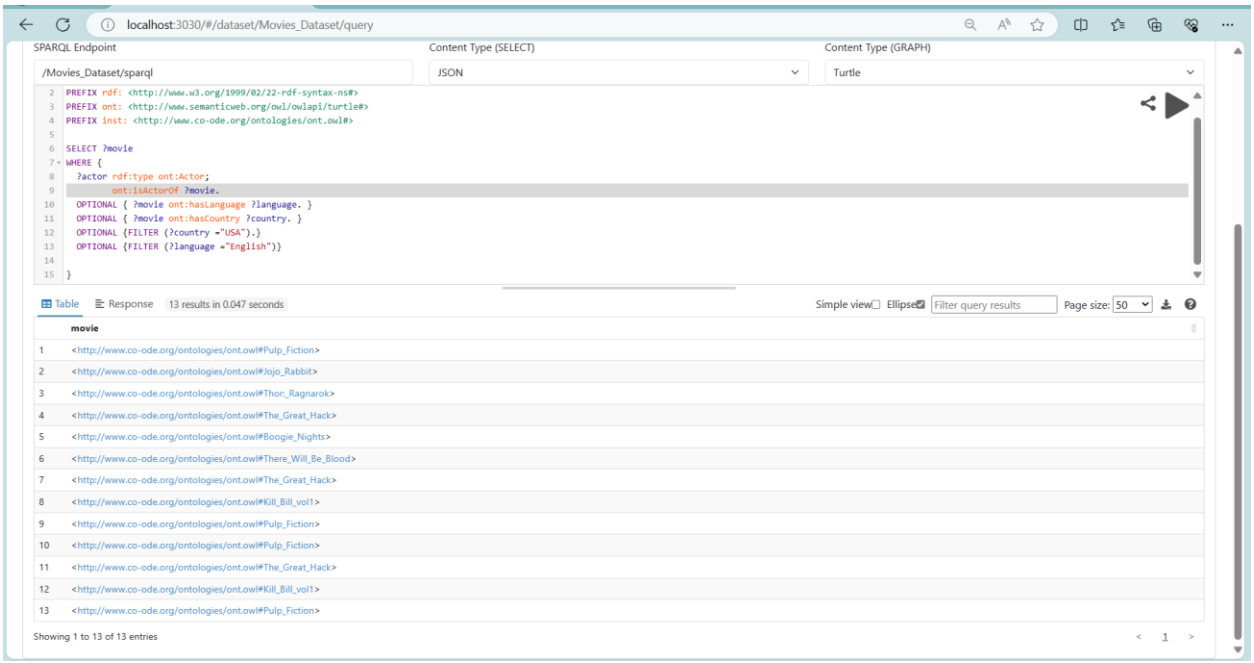
9. *Find movies with a certain language.*



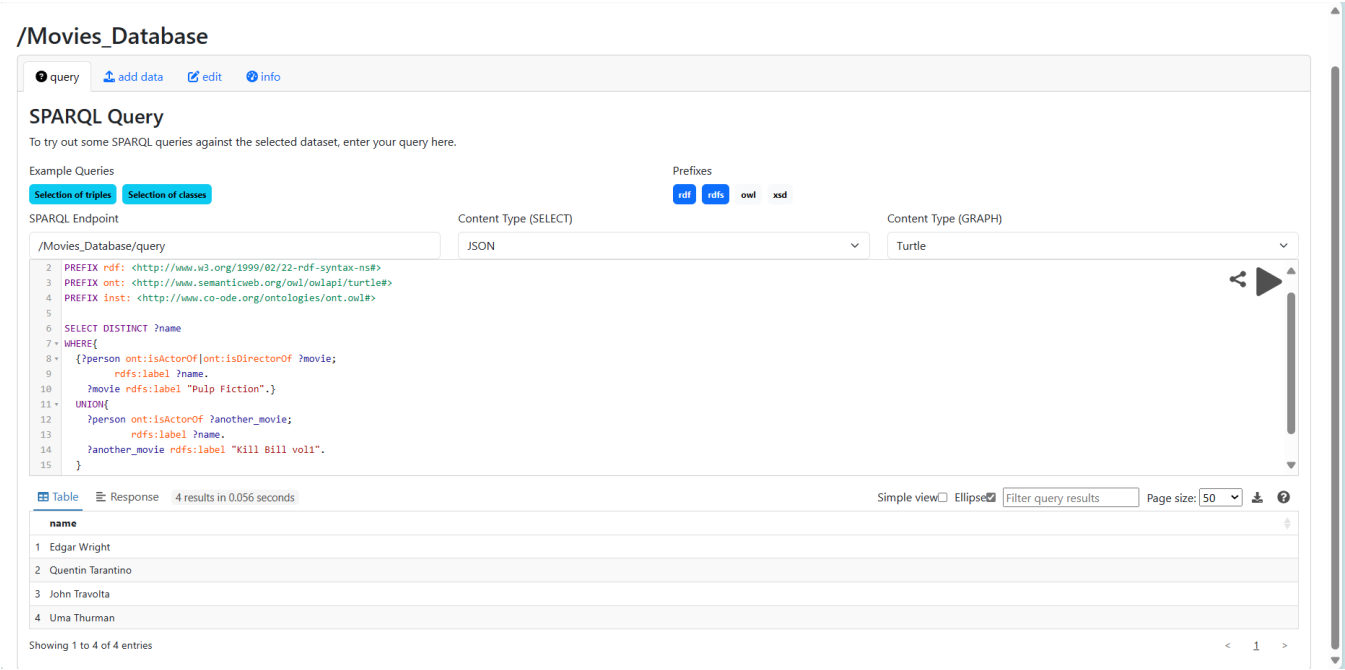
10. *List the name of Actors older than 51 years.*



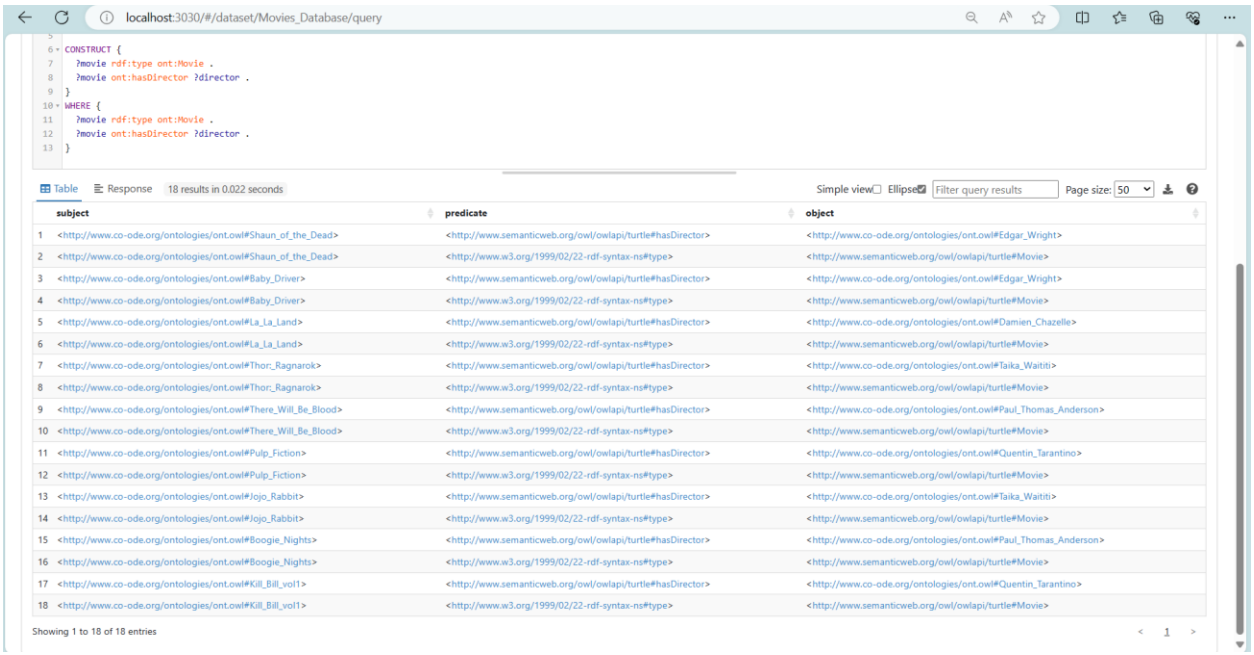
11. A query that contains at least 2 Optional Graph Patterns



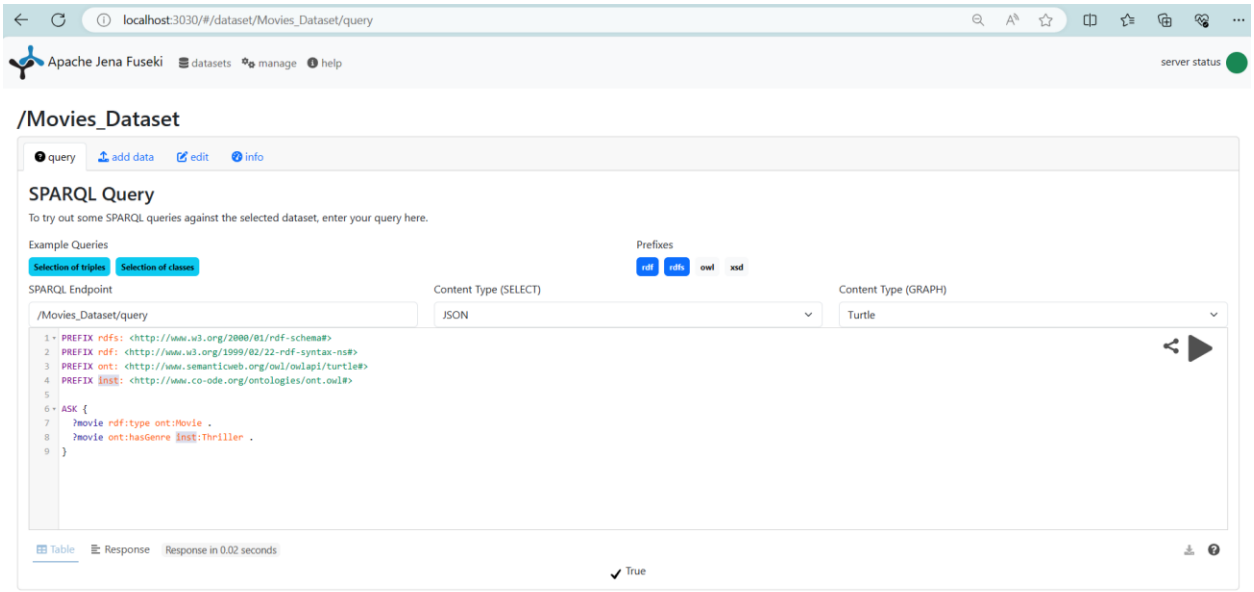
12. A query that contains at least 2 alternatives and conjunctions



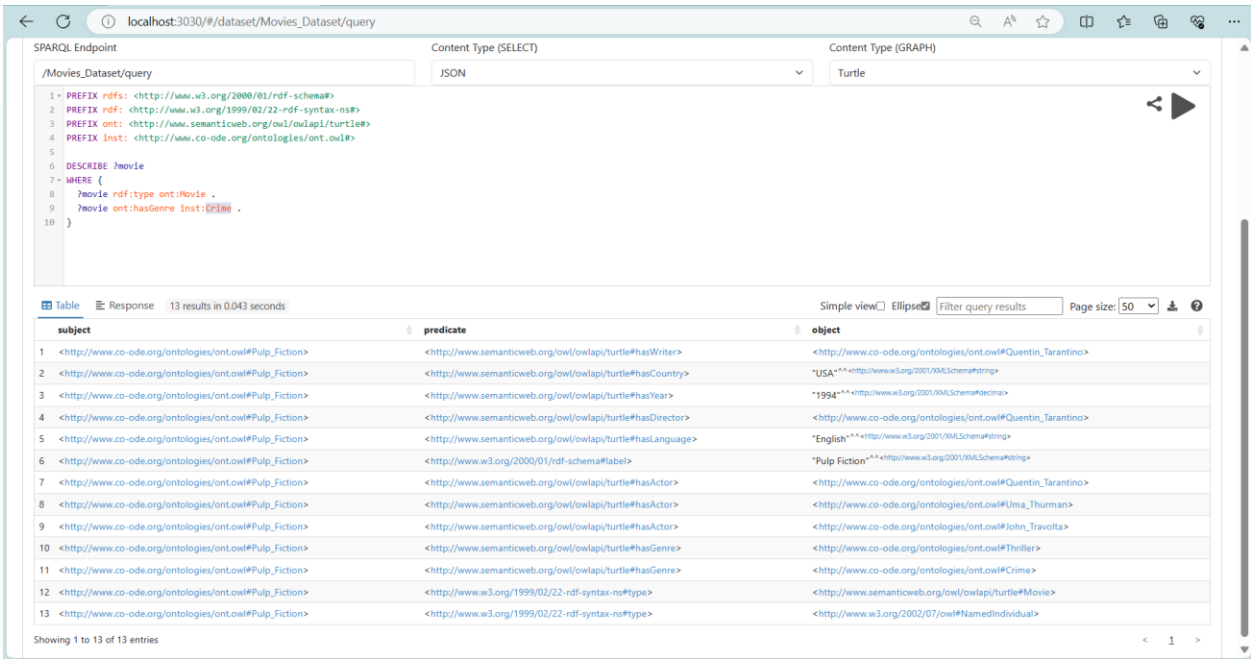
13. A query that contains a CONSTRUCT query form



14. A query that contains an ASK query form



15. A query that contains a DESCRIBE query form



7. Snapshots of Interface

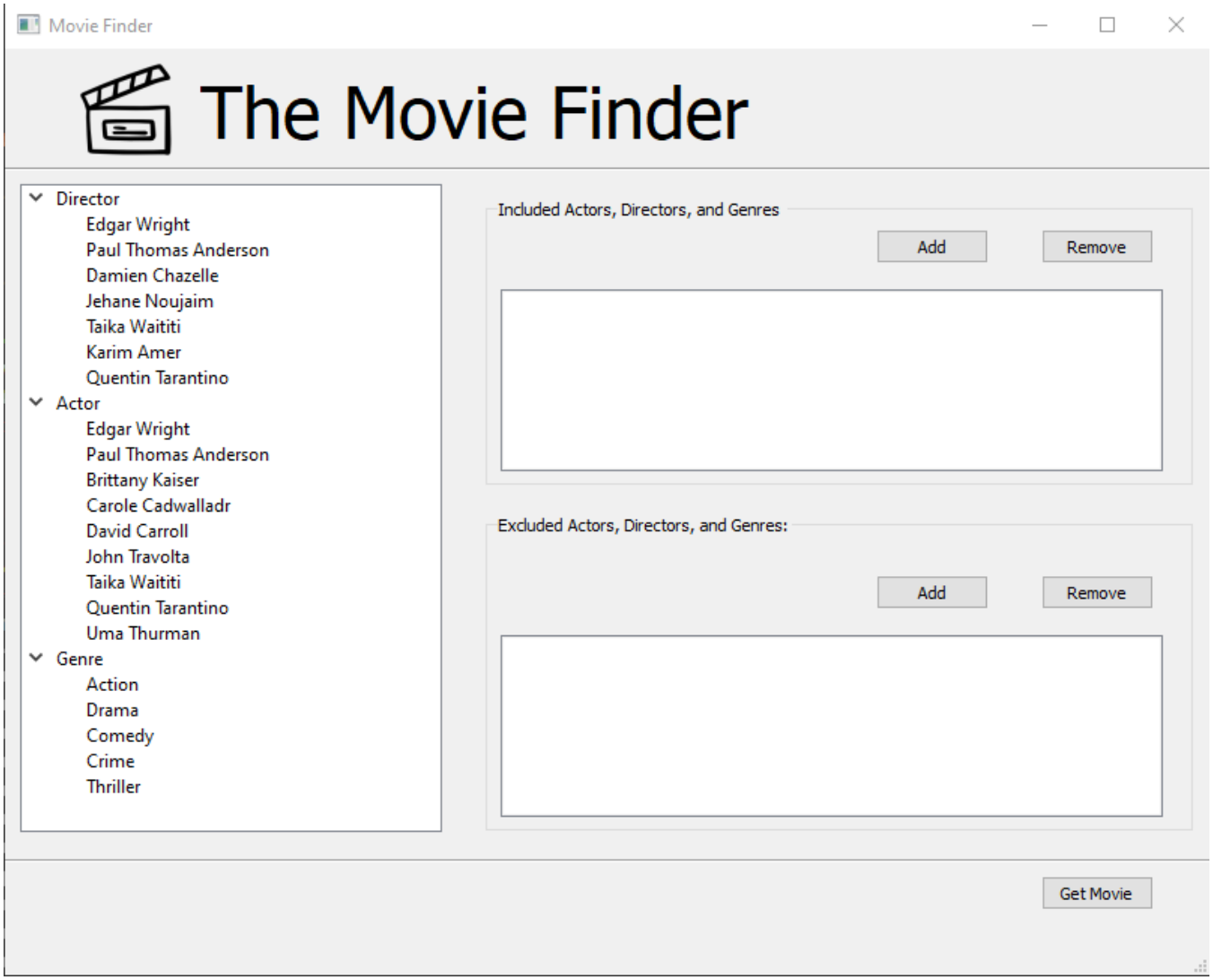


Figure 9 GUI Homepage

This is the homepage of our system , when we click on the button “Get Movie” it directs us to the next page , where it shows all movies in the ontology.

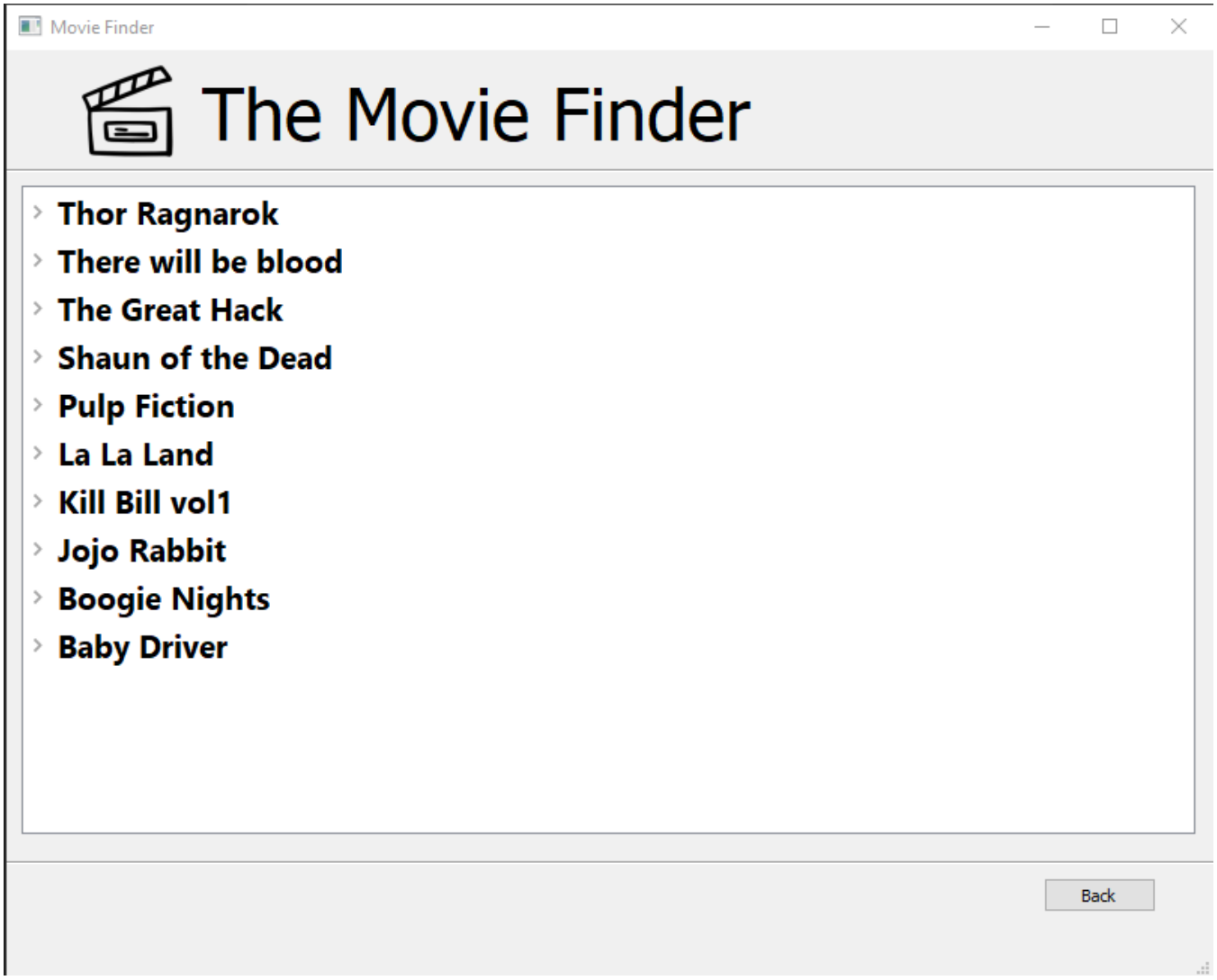


Figure 10 Get movies output

This screens shows all movies in the ontology

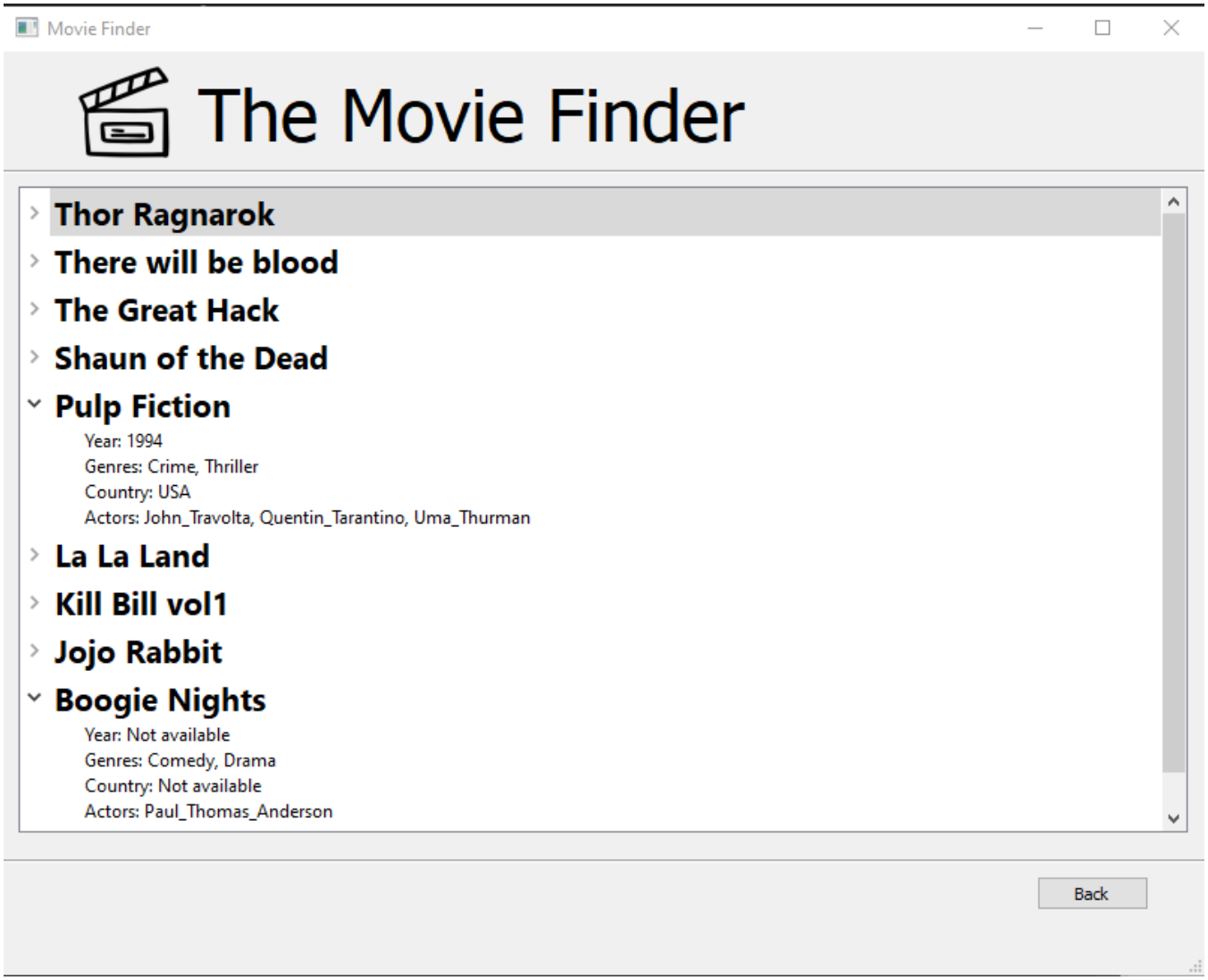


Figure 11 Movies Information

When extending the movie we get the detail of the movies

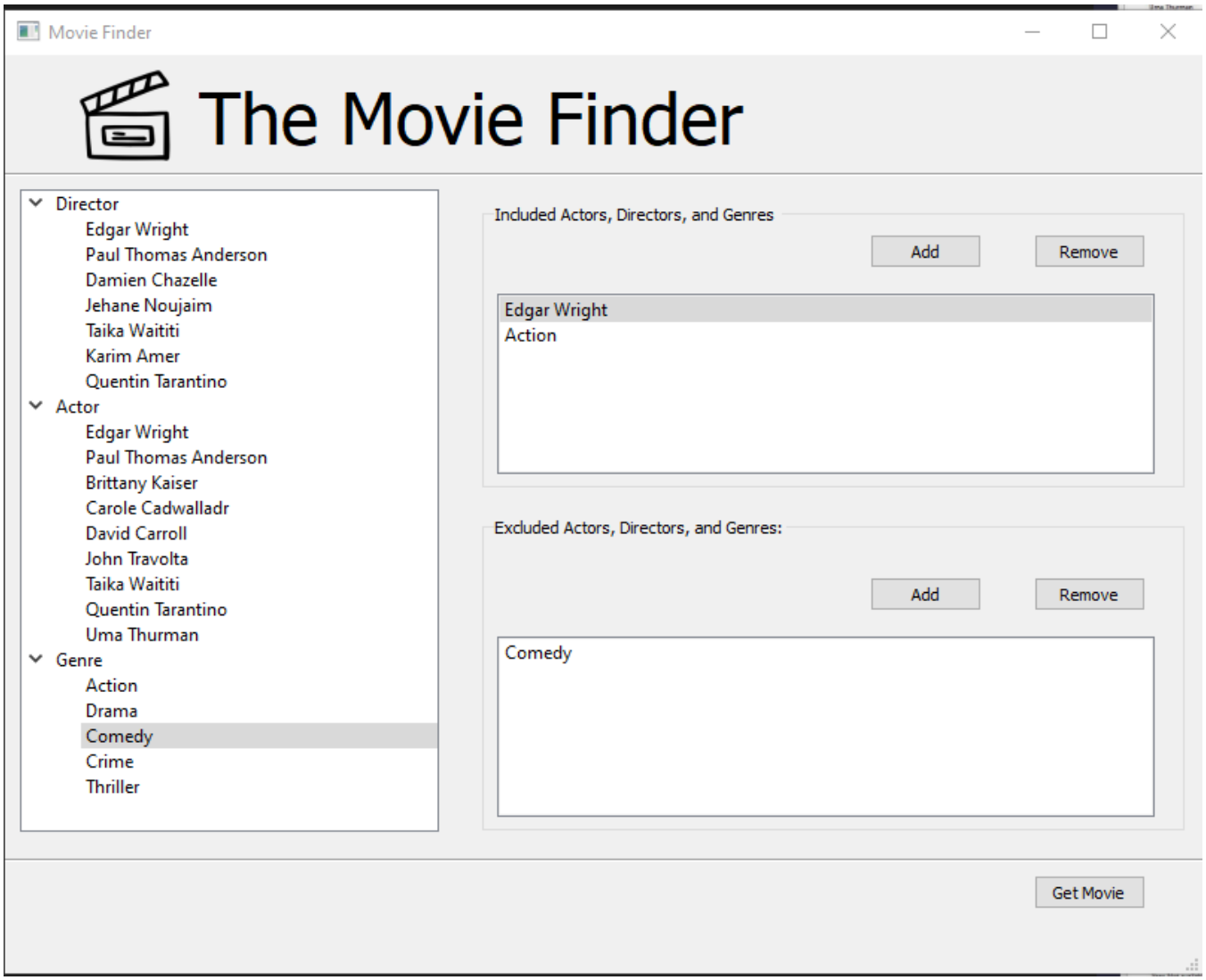


Figure 12 Test Application 1

We added restrtictions to the query , where we only want movies that includes the individual “Edgar Wright” and excluding movies where the Genre is “Comedy”

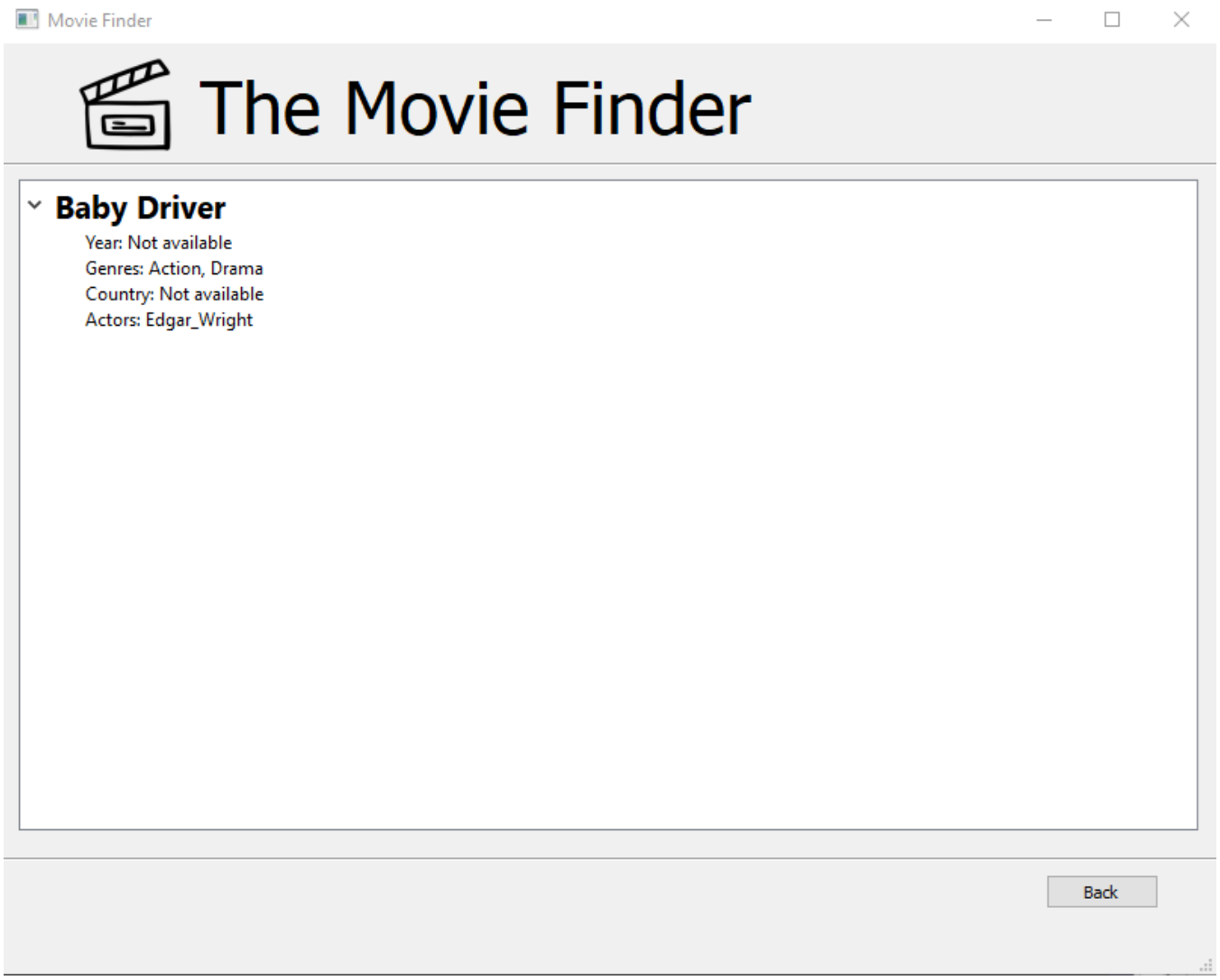


Figure 13 output of test1

The output clearly shows that the restrictions were applied successfully.

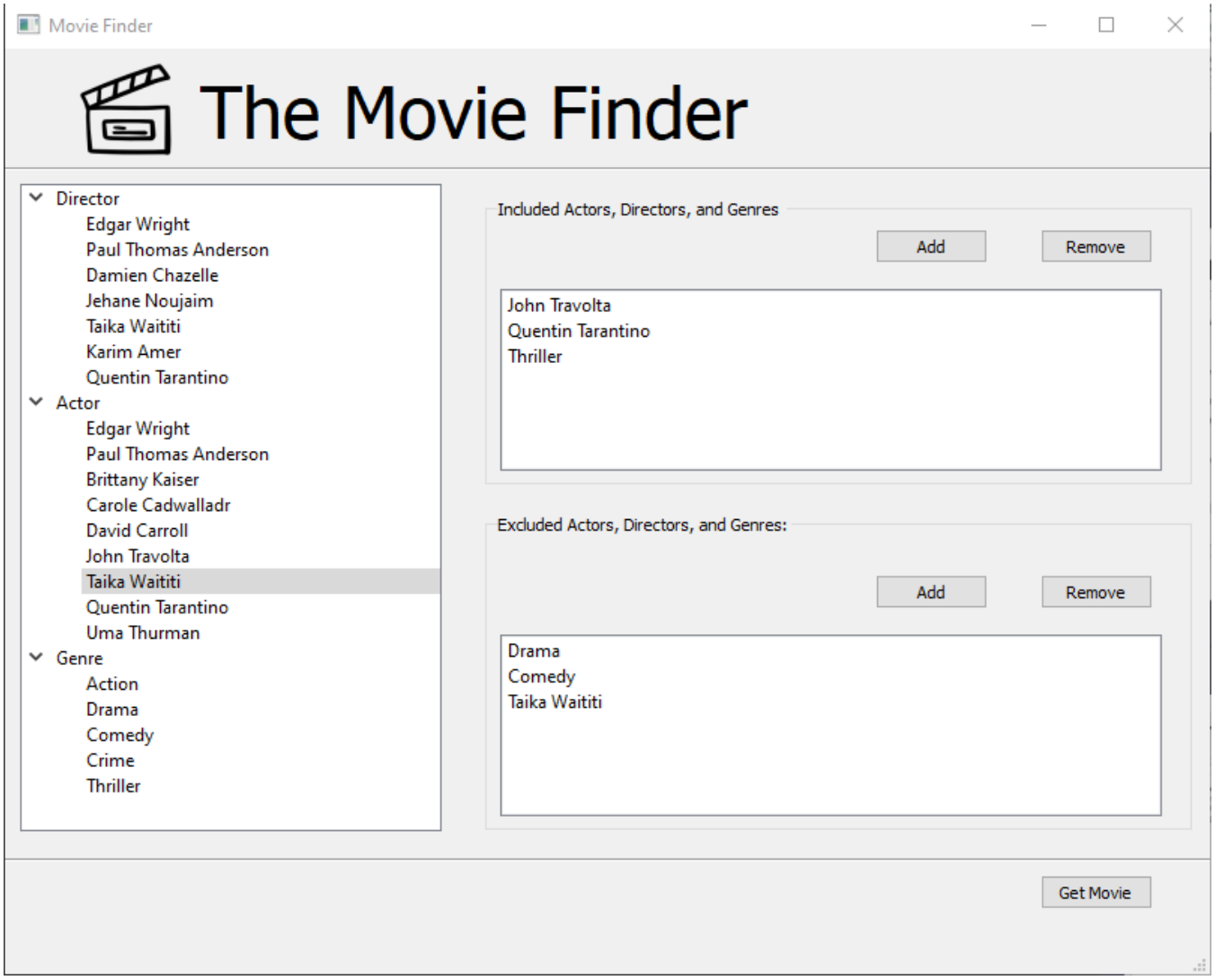


Figure 14 Test Application 2

Here we added the restrictions where we want to include the above actors,directors,and genres while excluding the below above actors,directors,and genres

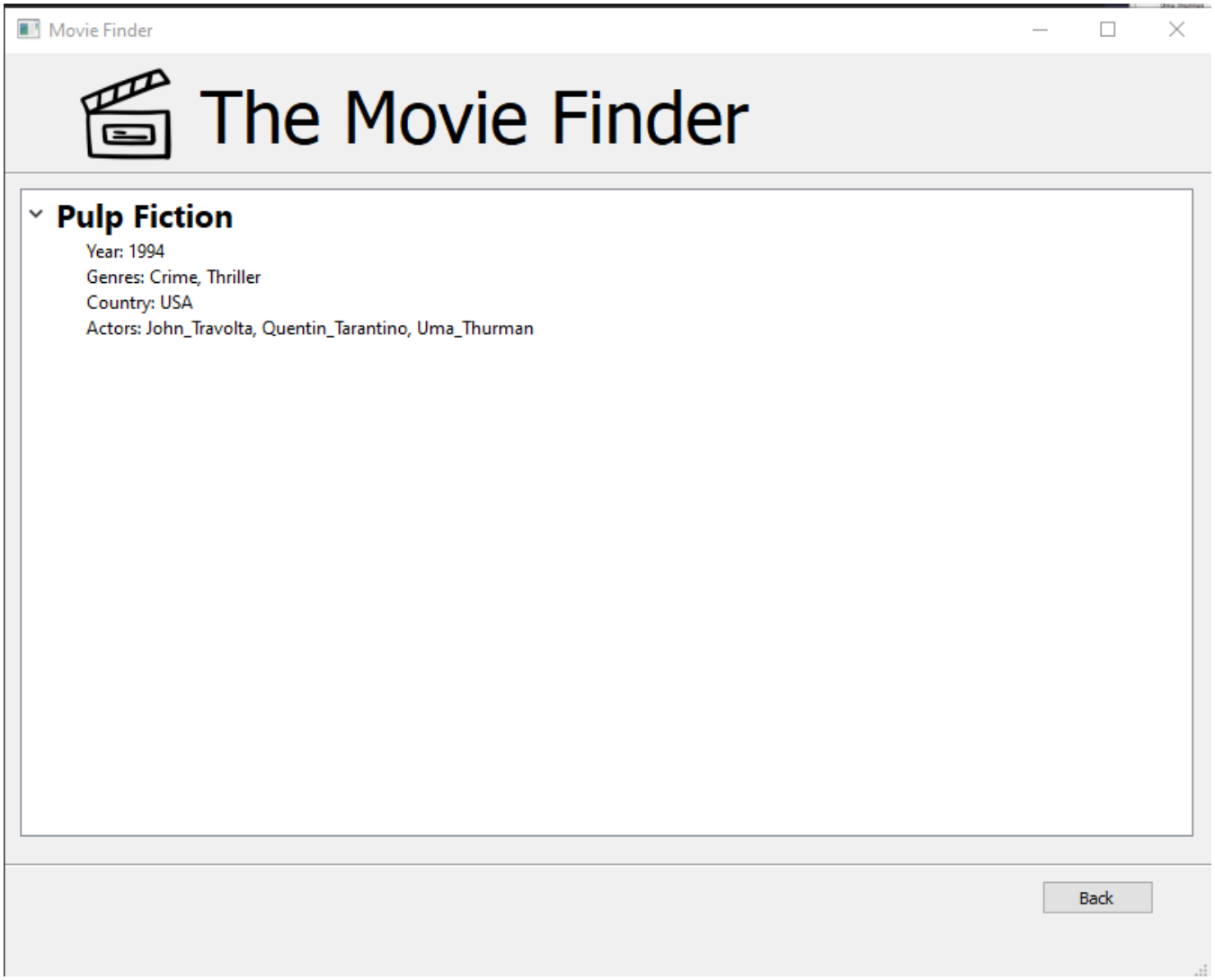


Figure 15 output of test 2

The output clearly shows that the restrictions were applied successfully.

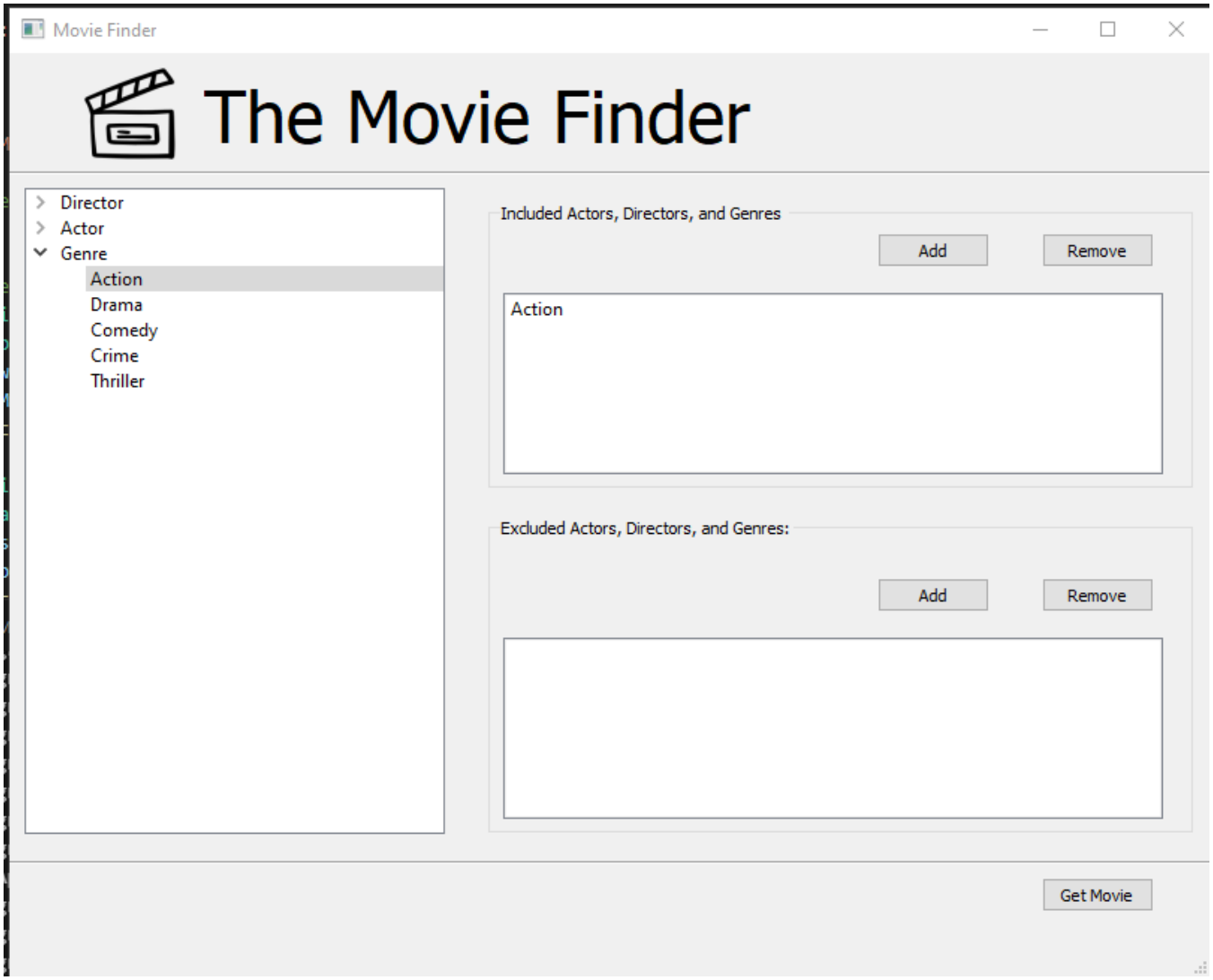


Figure 16 Test Application 3

This test case tests the scenario where we try to add the same restriction in the include , the next screenshot will try to add the same restriction in the exclude, this will result in removing the restriction from the include as it is not logical to include and exclude the same restriction

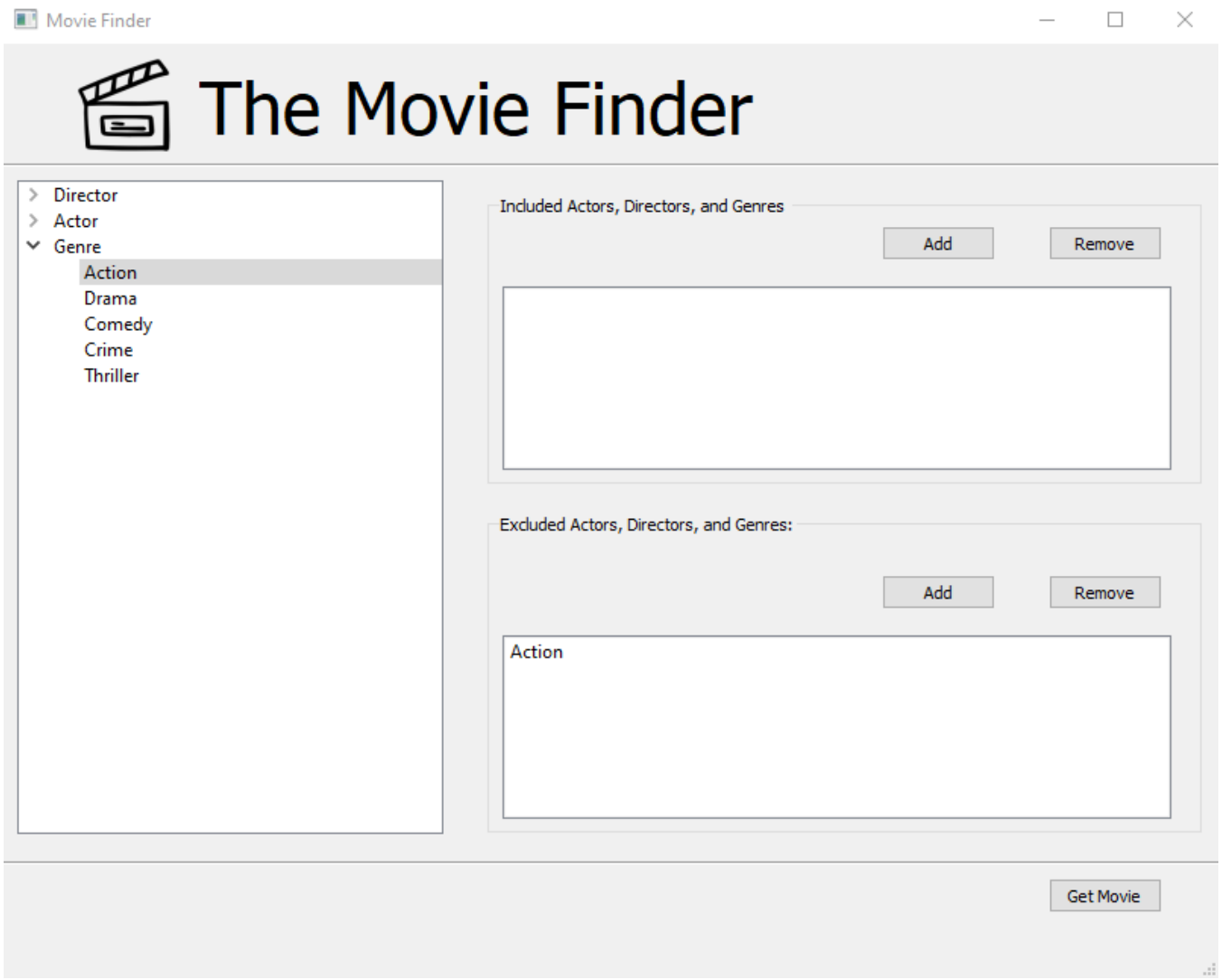


Figure 17 Test application 3

Restriction here is removed from the include and only left in the exclude.

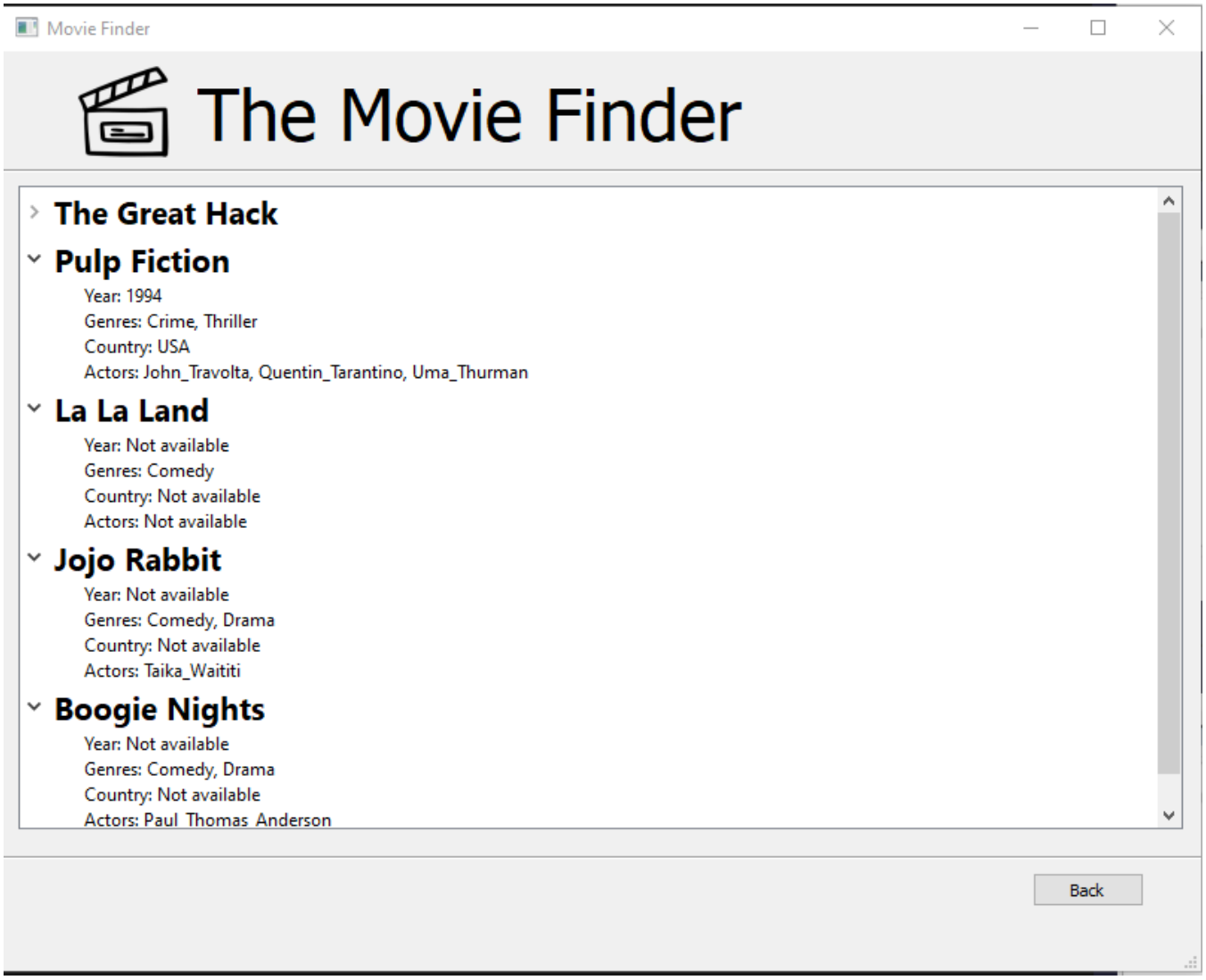


Figure 18 Output of Test Application 3

The output shows that the exclude restriction have been satisfied successfully

The above the test cases covers almost all functionalities of the application including the corner cases. Hence the application proved ontology consistency, inference correctness, and query performance.