Introduction

At the beginning weeks of the 2020-2021 Spring semester, this group of students were tasked with the creation of an ontology using Protege-OWL that stores information about the films, series, documentaries, anime and other mediums of entertainment on Netflix. The main purpose of the system was to recommend the users of the system things to watch depending on their preferences (category, rating, new content, etc.) in the ontology. Additionally, a report with the content of the knowledge representation scheme, the acquired knowledge of the students during the creation and development of the project, and the necessary information about the system needed to be created. This report is that aforementioned report.

Class Creation & Modification

The project team initially thought of a class structure that consisted of two different main classes: Client and Netflix. The main thought behind this decision was that all the necessary information about the Netflix shows, movies, animations etc. could be kept in the Netflix class, while all the user information like names and nationality, with the possible preference object and data properties could be kept in the Client. The team worked for the majority of the project to handle the preferences of the client to be displayed in a DL Query within the Protégé system, to small degrees of success. After multiple attempts and variations, the team decided to contact the Project Coordinator Çiğdem Turhan in order to achieve some insight. During the meeting, current class structures and problems were shown, and the project coordinator acknowledged the limited capabilities of creating a preference system in the Protégé application without additional coding or outside help, and modified the project to become a query-based system that could optionally hold a singular user whose preferences can be changed manually. The team then set to creating the system with those goals in mind, and succeeded. The upcoming parts of the report goes to explain the current class hierarchy structure, object and data properties and their explanations.

Class Hierarchy explanation

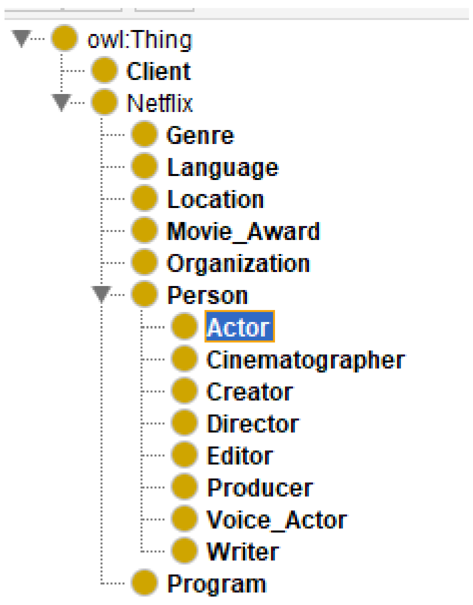


Figure 01: Class hierarchy

Figure 01 visualizes the class structure of the system. The class structure consists of 2 main classes, with the second class having a subclass. The client class includes basic information about the possible users, but the usability of this class is greatly reduced since the modification of the project, the Netflix class holds the information about the included shows in the system, with the person subclass holding the names of the important figures who work on those programs. The explanations of the classes are explained below.

Individual class explanation

Client: Main class where the user information is being kept.

Netflix: Main Class where the entertainment information is being kept.

Genre: What genre of entertainment the program falls under (Such as Drama, Thriller, Animation).

Language: The main language the show is created in.

Location: What the main locations the programs were shot in (Used in Live Action)

Movie_Award: The awards the program has won. (Such as Golden Globe or MTV Movie Awards)

Organization: The Main Company behind the organization of the program.(Such as Nickelodeon or Sony)

Person: Subclass where all the important figure names are located.

Actor: Main protagonist/actor of the program. (Such as Al Pacino for Scarface)

Cinematographer: Main accredited Cinematographer of Live-action programs.

Creator: The Creator of the original animation or anime (Such as Michael Dante DiMartino for Avatar)

Director: Director of the Program.

Editor: Main accredited Editor of the program.

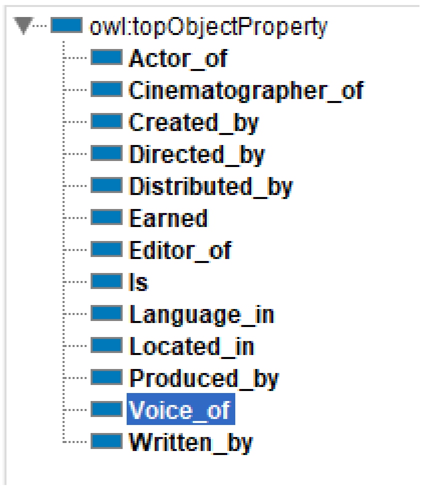
Producer: Main producer of the Program.

Voice_Actor: Main voice actors for animation and anime programs (Such as Tom Hanks in Toy Story).

Writer: If the show is a visual adaptation of another work of art, the writer of the original work of art is **stored** here (Such as J.R.R Tolkien for The Lord Of The Rings)

Program: The names of all the programs included in the system.

Object properties



owl:topObjectProperty
- Actor_of
- Cinematographer_of
- Created_by
- Directed_by
- Distributed_by
- Earned
- Editor_of
- Is
- Language_in
- Located_in
- Produced_by
- Voice_of
- Written_by

Figure 02: Object Property hierarchy

The object properties help the system make connections between individuals when necessary. In our system, the object properties mainly make the connections between the programs and their individual parts. The The domains and ranges of the object properties can be seen on Table 01. The actor, cinematographer and editor properties have differentiating range and domain values compared to their counterparts. The reason for this is that those objects were among the first objects that were designed on this project, and their values were deemed correct at the time. After modifying the system, the original values of those values stayed in the system, but currently they have no negative effects on the system despite being on the opposite side of the table they should have been.

Object Property Name	**Domain**	**Range**
Actor_of	Actor	Program
Cinematographer_of	Cinematographer	Program
Created_by	Program	Creator
Directed_by	Program	Director
Distributed_by	Program	Organization
Earned	Program	Movie_Award
Editor_of	Editor	Program
Is	Program	Genre
Language_in	Program	Language
Located_in	Program	Location
Produced_by	Program	Producer
Voice_of	Program	Voice_Actor
Written_by	Program	Writer

Table 01: Object Property Table

Data properties

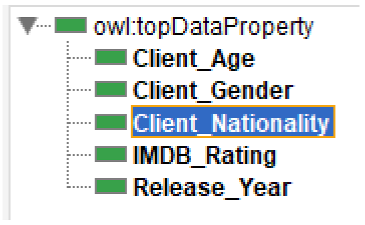
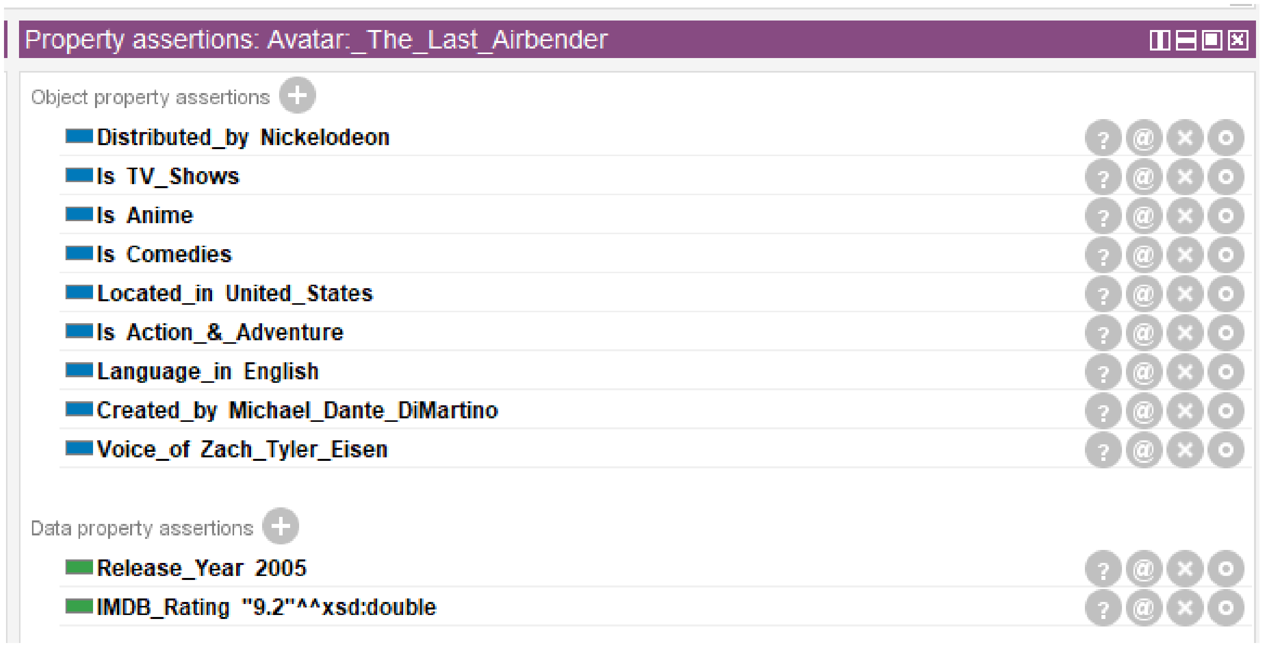


Figure 03: Data Property hierarchy

The Data properties hold the necessary information when it only needs to be gathered, not to be connected. In our system, the object properties mainly hold the information about clients (Which are not currently used), the release years and IMDB Ratings of Programs. The domains and ranges of the data properties can be seen on Table 02.

Data Property Name	**Domain**	**Range(Variable type)**
Client_Age	Client	integer
Client_Gender	Client	String
Client_Nationality	Client	string
IMDB_Rating	Program	Double
Release_Year	Program	integer

Table 02: Data Property Table



Property assertions: Avatar:_The_Last_Airbender

Object property assertions

Distributed_by Nickelodeon

Is TV_Shows

Is Anime

Is Comedies

Located_in United_States

Is Action_&_Adventure

Language_in English

Created_by Michael_Dante_DiMartino

Voice_of Zach_Tyler_Eisen

Data property assertions

Release_Year 2005

IMDB_Rating "9.2"^^xsd:double

Figure 04: Sample Property assertion of Avatar: The Last Airbender.

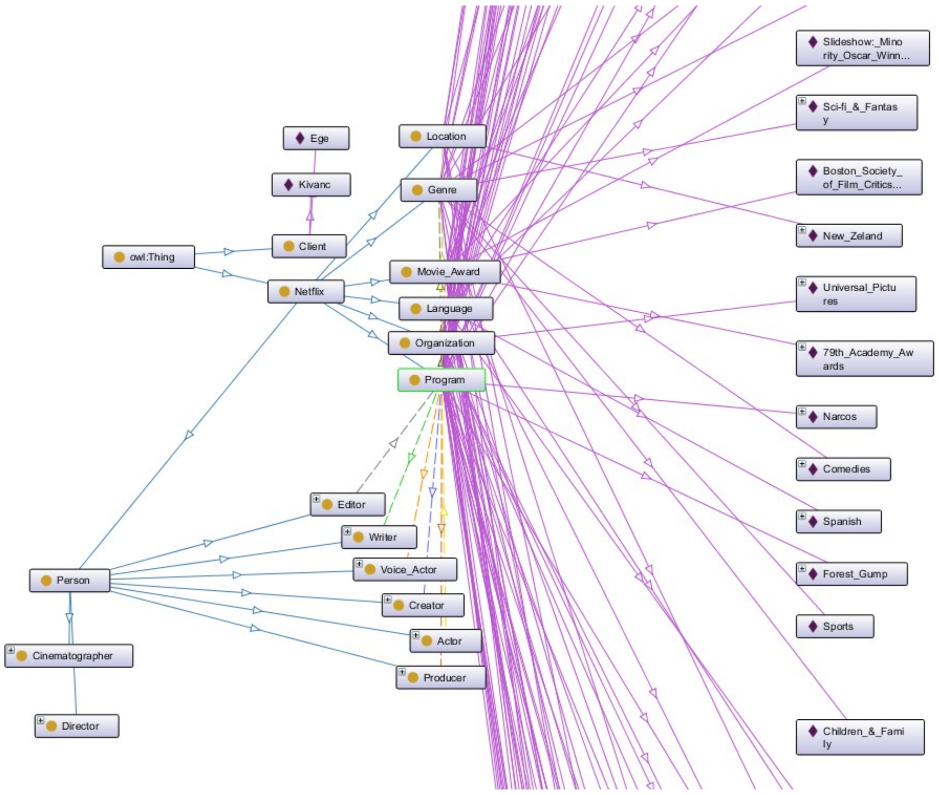
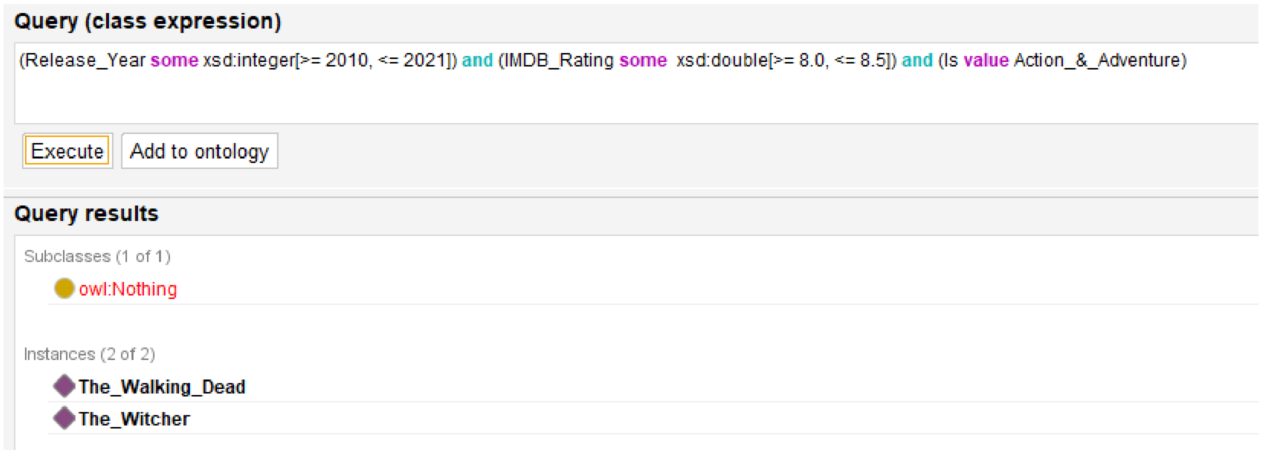


Figure 05: OntoGraf screenshot



```
(Release_Year some xsd:integer[>= 2010, <= 2021]) and (IMDB_Rating some  xsd:double[>= 8.0, <= 8.5]) and (Is value Action_&_Adventure)
```

```
Subclasses (1 of 1)
    owl:Nothing

Instances (2 of 2)
    The_Walking_Dead
    The_Witcher
```

Figure 06: Simple Sample Query

Current System

needs to be The current ontology system works by connecting the object properties to their corresponding class instances and making the DL query commands to help the user find a certain program according to their criteria. A sample OntoGraf of the system can be seen on Figure 05. This figure only shows a very small number of samples since the system currently holds 30 programs. If the whole OntoGraf structure needed to be shown, this report would need a dozen more pages because of the exceedingly high number of instances. A sample of the necessary connections between instances and objects can be seen in a property assertion table on Figure 05. Finally, a sample DL Query while the system is in action can be seen on Figure 07.

5. Conclusion

In conclusion, the team has learned to create ontologies, create a system that can reason and use knowledge with the user it is interacting with, improve their pre-existing knowledge of using DL Queries, create specific interconnected class structures that trade information with each other in Protégé during this project. The members are currently capable of designing a similar system thanks to their current knowledge of Protégé, ontologies, data structures, data and object hierarchy knowledge; and this knowledge is sure to help each team member while designing similar projects in the future.