Artificial Intelligence

Laboratory 2: Family Relatives Ontology

Nurulla Zholdoshov

1 Introduction

In this report I will describe the process followed to create an ontology, which will help to find someone's ancestors or descendants. It is important to know yourself, your family relatives and your family member relationships. For instance, if you need blood in a rare group, you can find easily in your relatives.

2 Vocabulary

The vocabulary of this ontology is very ordinary and very easy to understand.

- Human: one of the main elements, represents the main objects in this ontology.
- Male: this one identifies which describe biological sex—in humans, non-human animals, and other organisms.
- Female: this one identifies which describe biological sex—in humans, non-human animals, and other organisms.
- Parent: it is the caretaker of a child. A person's father or mother.
- Father: a man in relation to his child or children.
- Mother: a woman in relation to his child or children.
- Grandparent: a parent of one's father or mother; a grandmother or grandfather.
- Grandfather: the father of one's father or mother.
- Grandmother: the mother of one's father or mother.
- Child: it is a person younger than the age of majority. A person's son or daughter.
- Son: a boy or man in relation to either or both of his parents.
- Daughter: a girl or man in relation to either or both of his parents.
- Grandchild: a child of one's son or daughter.
- Grandson: the son of one's son or daughter.
- Granddaughter: the daughter of one's son or daughter.
- Married: people are legally committed to each other — they're joined in marriage, or wedded.
- Husband: a married man considered in relation to his spouse.
- Wife: a married woman considered in relation to his spouse.

- Sibling: is a gender neutral word for a relative that shares at least one parent with the subject. A person's brother or sister.
- Brother: a man or boy in relation to other sons and daughters of his parents.
- Sister: a woman or girl in relation to other sons and daughters of his parents.

3 Developing the ontology

After the retrieving all the data, we can begin the process to develop the ontology. All of the elements above can be defined as classes.

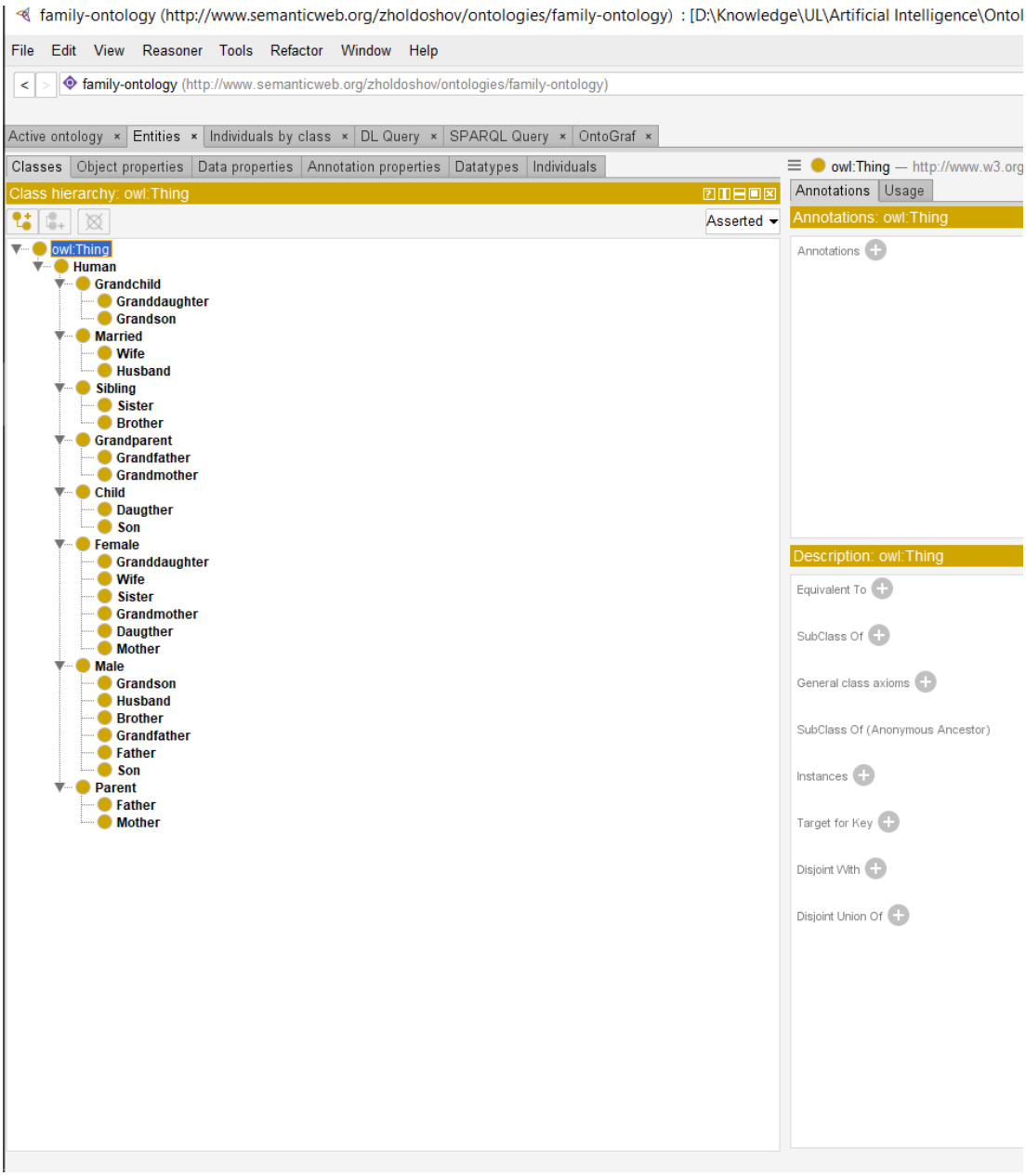

Figure 1: Class hierarchy

Figure 1 show the classes defined to represent each element. Here is only two types of elements, male or female. This is because to find person father or mother, son or daughter etc.

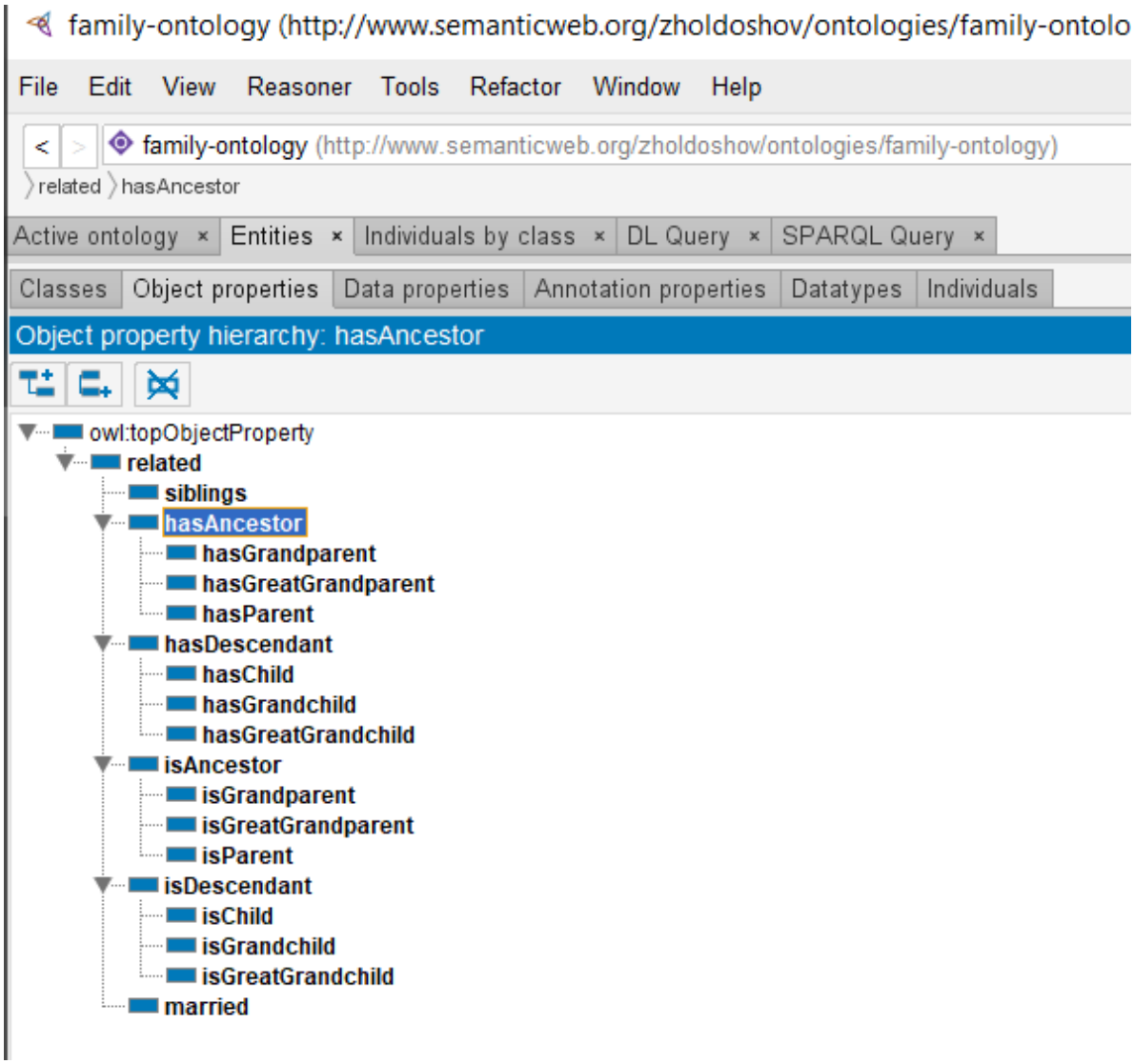

Figure 2: Properties

In figure 2 the properties defined for the elements can be seen. All the relationships between the classes specified in the vocabulary are established with them. One example is the *hasGrandparent* described above, that person has a parent of him father or mother.

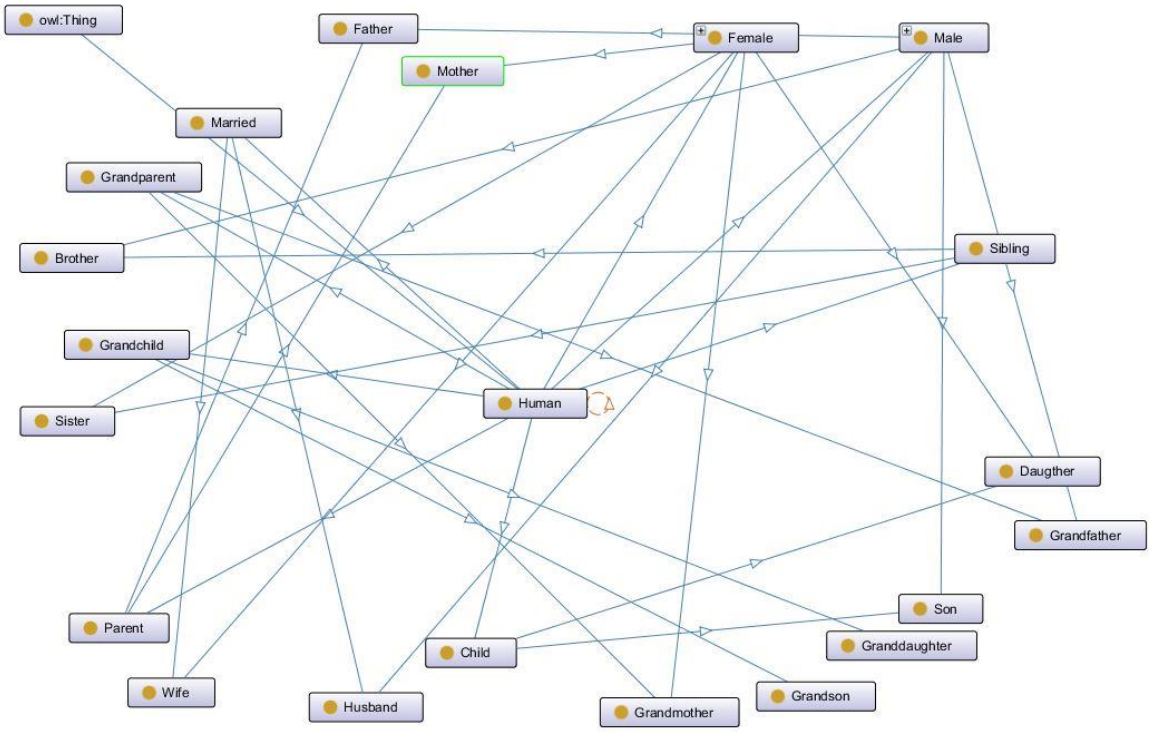

Figure 3: Ontograph

The ontograph, which is displayed in figure 3, contains all the relationships seen in figure 2, as well as the class hierarchy represented graphically.

4 Ontology checking

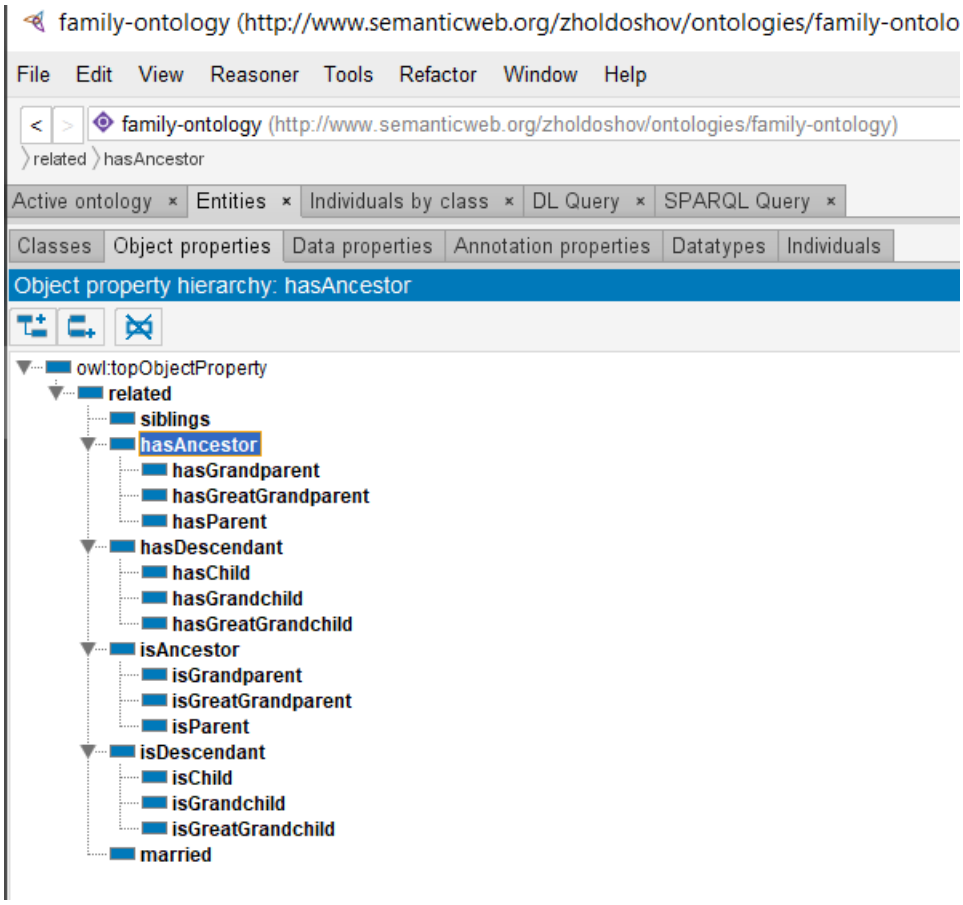

Figure 4: No errors after running HermiT

To test the ontology, one reasoner was run. It was HermiT, which threw no errors, seen in figure 4.

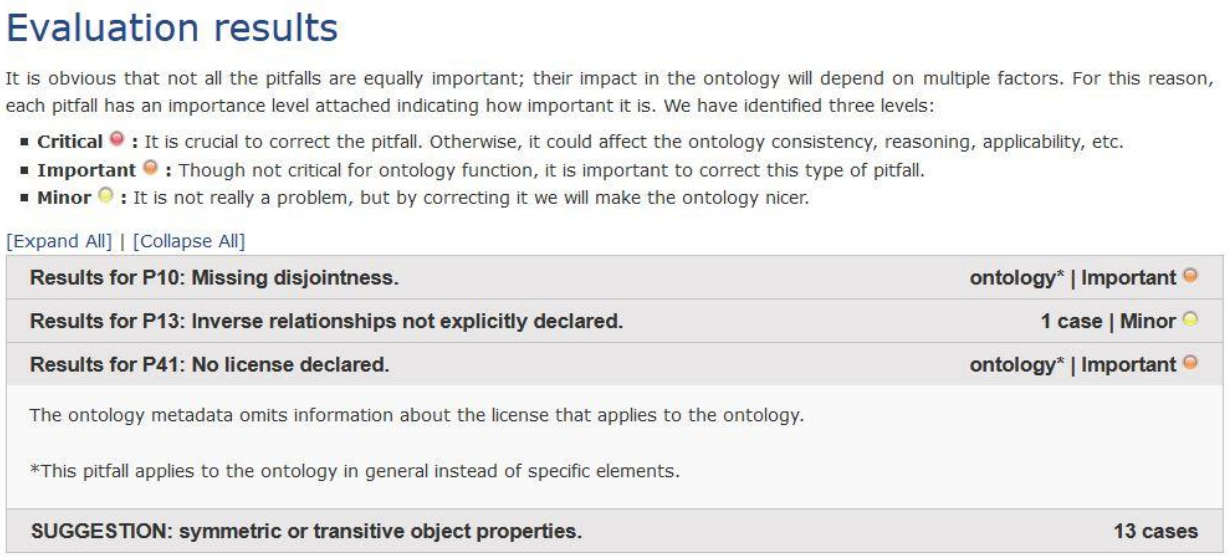

Figure 5: OOPS tool results

Using the OOPS! Web tool, the ontology was analyzed, which results are shown in figure 5. The main errors were not fixed due to fact that both of them I can't find solution inside the ontology, but it doesn't affect to the ontology. I think sometimes the analyzer shows wrong suggestions, for example: 'SUGGESTIONS: symmetric or transitive object properties. 13 cases'.

4 SPARQL

I take one random family relatives or family tree to test them in my ontology. I add all them to the individuals.

```
SELECT ?subject ?object
    WHERE { ?subject fam:hasParent ?object }
```

This query would return all the parents that have the object.

```
SELECT ?subject ?object
    WHERE { ?subject fam:hasChild ?object }
```

This query would return all the child that have the object.

```
SELECT ?subject ?object
    WHERE { ?subject fam:married ?object }
```

This query would return all the married objects.

```
SELECT ?subject ?object
```

```
        WHERE { ?subject fam:siblings ?object }
```

This query would return all the siblings that have the object.

```
SELECT ?subject ?object
        WHERE { ?subject fam:hasGreatGrandchild ?object }
```

This query would return all the Great Grandchild that have the object.