IA301 - Logics and Symbolic AI
TP

Joaquim Minarelli Gaspar
Filipe Lacerda
Eduardo Guimarães
Yaël

Palaiseau, France

Novembre 2024

Table des matières

1 OWL with Protege

1.1 Simple Ontology (Preliminary Case)

We explore the tool using the given example.

Let's answer the few questions asked at the end of this part.

— What do we observe when we check the inferred classes ?

Answer : The inferred classes are the same as the asserted classes. This means that there are no additional logical consequences or inferences made by the reasoner beyond what has already been explicitly asserted in the ontology. This indicates that the ontology is consistent, and the asserted hierarchy or relationships between classes are correctly modeled according to the ontology's logical structure.

— Some DL queries examples and results.

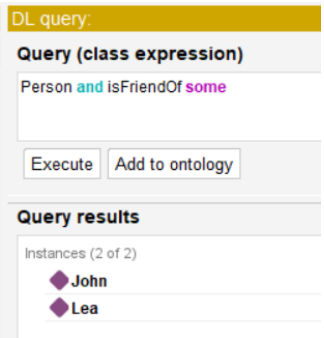

FIGURE 1 – DL query : Persons who have friends

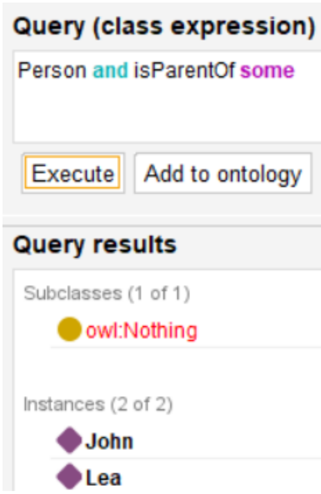

FIGURE 2 – Persons who are Parents

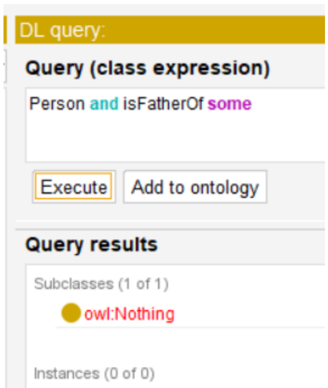

FIGURE 3 – Persons who are Fathers : Issue there

FIGURE 4 – Adding two axiom to the Father Class

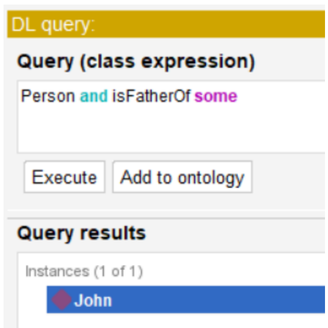

FIGURE 5 – This enabled the reasoner to infer that John is a Father

FIGURE 6 – All Images Displayed in Two Rows with Harmonized Widths

1.2 Ontology of Ecosystem Interactions and Resource Consumption

To explore the logical relationships related to the consumption of natural resources and the environment, the theme chosen for the ontology was the model of an ecosystem, with a focus on the relationships and interactions among organisms (such as animals, plants, and microorganisms), their environments, and the resources they require.

Overall, our model presents three main classes that interact with each other : the classes of **Organisms**, **Resources**, and **Habitats**. Organisms are further divided into three main subclasses : Animals, Plants, and Microorganisms. These organisms inhabit specific Environments and consume a set of natural resources provided by those environments.

Additionally, important relationships can be inferred from this model, with the main ones being :

— The relationship of coexistence, where two organisms cohabit the same environment without one being a predator of the other, or, when applied to microorganism, if they do not consume the same resource ;

— Relationships involving the consumption of certain natural resources based on the characteristics of the organisms and the environment in which they live ;

— Relationships concerning the food chain within the ecosystem, based on the type of food/energy source required by the organism along with characteristics of the environment.

The following image illustrates the relationship diagram that was implemented in the Protege software.

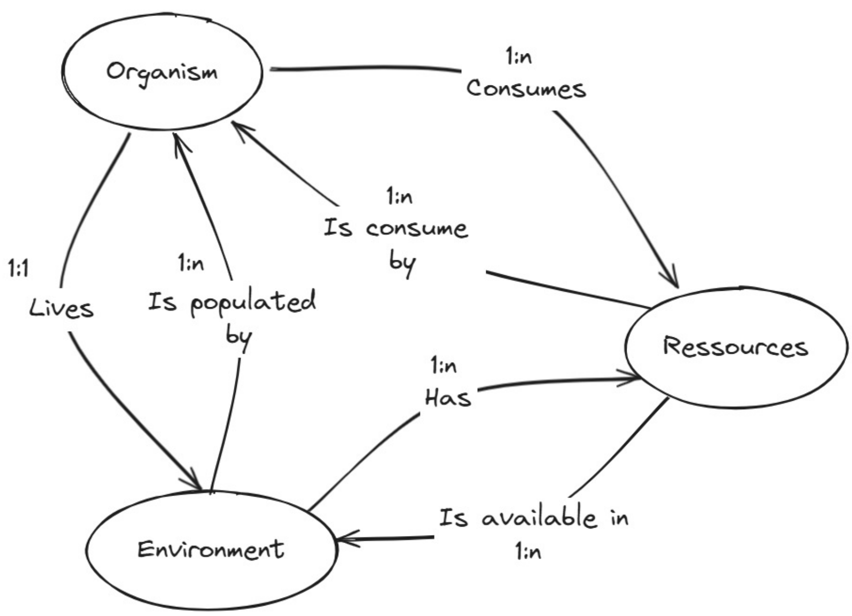

FIGURE 7 – Relationship diagram

In Figure 11, there is a cardinality relation above the arrows. These relate m individuals from the source category with n individuals from the target category. For example, the "Consumes" relation going from Organism to Resource indicates that 1 individual can consume n resources. Conversely, its symmetric relation, "Is consumed by," indicates that 1 resource can be consumed by n organisms.

1.3 Protege implementation

The following describes the classes and subclasses used to create the ontology in Protege.

- **Classes** (that are subclasses of owl :Thing) :
  - Organism ;
  - Habitat ;
  - Resource.

- **Subclasses** :
  - Animal, Plant and Microorganism as subclasses of Organim (all mutually disjoint from each other) ;
  - Herbivore and Carnivore as subclasses of Animals (they are both disjoint) ;
  - Bryophytes and Angiosperms as subclasses of Plants (all mutually disjoint from each other).

It is important to note that all classes and subclasses are mutually exclusive ; therefore, no individual from one class can belong to another, and no individual from one subclass can belong to another subclass. The hierarchy of classes can be better observed in the figure below.

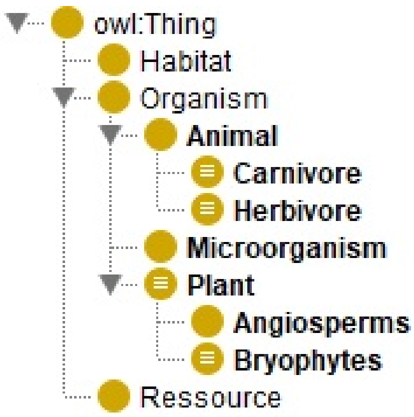

FIGURE 8 – Class hierarchy

Regarding the Class properties, we have the following **Data properties** :

- *weight* with domain Animal and range xsd :decimal ;
- *has_ Fruits* with domain Angiosperms and range xsd :boolean ;
- *chemical_formula* with domain Ressource and range xsd :string. We set this property to be unique, as it will be used as the identifier of the ressource ;
- *geographical_ coordinates* with domain Habitat and range xsd :string. It also has the unique property, and will be used as the identifier of the environment ;
- *scientific_ name* with domain Organism and range xsd :string. Again, it's unique and used as an indentifier for the organisms.

The following two figures illustrate both the declaration of data properties and the method used to ensure the uniqueness of the *scientific_ name*, for example.

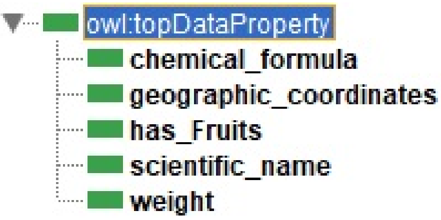

FIGURE 9 – Data properties

FIGURE 10 – Unique property

Finally, we have the following **Object properties** :

— *eat* with domain Animal and range Animal ∪ Plant ;
— *livesIn* with domain Organism and range Habitat ;
— *consumeTheResource* with domain Organism and range Resource ;
— *isEatenBy* with domain Organism and range Animal (is the inverse property *eat*) ;
— *isOccupiedBy* with domain Habitat and range Organism (is the inverse of *livesAt*) ;
— *hasTheResource* with domain Habitat and range Resource ;
— *isInTheEnvironment* with domain Resource and range Habitat (inverse of the property *hasTheResource*)
— *coExistWith* with domain Organism and range Organism. This property is symmetric. In addition, this property is different if both organisms are animals/plants or if they are both microorganisms :

1. A and B are animals/plants : A coexists with B means that A and B lives at the same environment (livesIn(A) == livesIn(B)) **and** they do not share neither the relation A isEatenBy B nor B isEatenBy A ;
2. A and B are both microorganisms : A coexists with B means that A and B lives at the same environment (livesIn(A) == livesIn(B)) **and** they do not consume the same resource through the relation consumeTheResource ;

The list of object properties in Protege can be seen in the figure below.

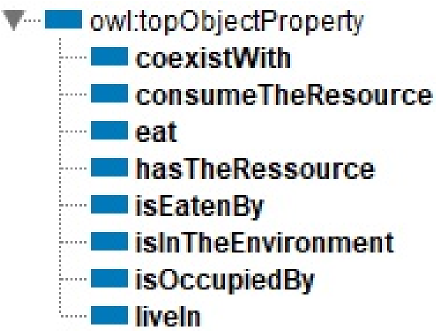

FIGURE 11 – Object properties

The inference rules were described using *SWRL* (Semantic Web Rule Language), which allows for writing it in a practical and concise manner. Below, we provide a detailed description of all the inference rules related to object and data properties.

We start by the following inference rule : if Organism a eat Organism b, and Organism a live in the Habit *env*, then b should also live in *env*. This is described by :

```
eat(?a, ?b) ^ liveIn(?a, ?env) -> liveIn(?b, ?env)
```

The same way, if Organism *a* coexists with Organism *b*, and *a* live in the Habitat *env*, *b* should also live in *env*.

```
coexistWith(?a, ?b) ^ liveIn(?a, ?env) -> liveIn(?b, ?env)
```

Now, regarding the plants, we can infer that a plant is an angiosperm if it has a fruit, which is a binary variable. Therefore, we have the following rule :

```
has_Fruits(?a, true) -> Angiosperms(?a)
```

From now on, we will focus on discussing the coexistence relationship, which is the most complex relation of the ontology. As previously described, according to our rules, a carnivorous animal will always coexist with another carnivorous animal as long as they inhabit the same environment. Additionally, we add the rule that animal *a* must be different from *b*, as we do not want the relation to be reflexive. To ensure that they are both different, we require that their scientific names be different, using the syntax *differentFrom*.

```
Carnivore(?a) ^ Carnivore(?b) ^ liveIn(?a, ?env) ^ liveIn(?b, ?env) ^ scientific_name(?a,
```

We use the same logic to describe the coexistence for the herbivores, as follows :

```
Herbivore(?a) ^ Herbivore(?b) ^ liveIn(?a, ?env) ^ liveIn(?b, ?env) ^ scientific_name(?a,
```

Also, carnivorous animals will always coexist with the plants that inhabit the same environment. In this case, it's enough to add the proposition *differentFrom(?a, ?b)* because if *a* is a Carnivore (by the previous proposition) and *b* is a Plant (also stated earlier), they should be different because both classes are disjoint.

```
Carnivore(?a) ^ Plant(?b) ^ liveIn(?a, ?env) ^ liveIn(?b, ?env) ^ differentFrom(?a, ?b) ->
```

Next, we have the case of coexistence between plants, which is similar to the herbivore's case :

```
Plant(?a) ^ Plant(?b) ^ liveIn(?a, ?env) ^ liveIn(?b, ?env) ^ scientific_name(?a, ?name_a)
```

Finally, the coexistence between two microorganism happens when they both live in the same environment **and** do not consume the same resource. To guarantee that both resources are different, we use again their unique identifier, which is the *chemical_formula* :

```
Microorganism(?a) ^ Microorganism(?b) ^ consumeTheResource(?a, ?res_a) ^ consumeTheResourc
```