OWL

Read

- Foundations of Semantic Web Technologies: chapter 4, 5.

Supplementary reading:

- OWL Pizzas: Practical Experience of Teaching OWL-DL:Common Errors & Common Patterns

Now we will take our family ontology one step further by adding more semantics using OWL. First, for a soft start and to get into Protégé, ontology editing and OWL, we will start by looking at an existing tutorial ontology, the pizza ontology. Parts of this exercise will be a revisit of first week's exercise.

1 The Pizza ontology

The pizza ontology is a well-known ontology in the semantic web community. It is developed for educational purposes by the University of Manchester, which is a leading university in the development of semantic technologies.

The pizza ontology and a tutorial that uses it is found at

- `http://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes`
- `http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/`

The tutorial is primarily for learning how to use Protégé 4. Use it to get help on how to use Protégé in the coming exercises.

1.1 Exercise

Open the pizza ontology in Protégé. Take some time to browse the class hierarchy, the property hierarchies and the individuals and note how the ontology describes the domain of pizzas.

1.2 Exercise

Find Margherita and see how it is defined as a pizza with only cheese and tomato topping. Look at the definition of VegetarianPizza. Is a Margherita pizza a vegetarian pizza? Why / why not?

1.3 Exercise

Find `hasIngredient`. What is the domain and range of this property? What are the subproperties of `hasIngredient`? What is the inverse property of `hasIngredient`? What property characteristics does `hasIngredient` have?

1.4 Exercise

Classify the ontology by choosing a reasoner and then "classify" in the reasoner menu. In the "Inferred class hierarchy" two classes show up as subclasses of `owl:Nothing`. Answer the following questions:

- In general, what is the difference between the asserted class hierarchy and the inferred class hierarchy?
- What does it mean for a class to be a subclass of `owl:Nothing`?
- Explain why these two classes appear as subclasses of `owl:Nothing`.
- Find Margherita in the inferred class hierarchy and see which classes are inferred as superclasses of Margherita.

1.5 Exercise

Add a new class Grandiosa as a subclass of NamedPizza. Define "Grandiosa" as something which

- `hasTopping some HamTopping`,
- `hasTopping some TomatoTopping` and
- `hasTopping some CheeseTopping`.

Classify the ontology. What superclasses are inferred as superclass of Grandiosa? Explain why.

1.6 Exercise

State in the ontology that a Grandiosa pizza comes from Norway, and that Norway is different from the other countries already present in the pizza ontology. Apply reasoning and explain the results.

2 Family relations in OWL

So far we have only been allowed to use RDFS vocabulary to describe family relations. Now we will extend our description using OWL constructs. OWL is more expressive than RDFS and allows us to express many more restrictions on properties and class membership than RDFS does.

In this exercise we will only use OWL (1) DL vocabulary (and not OWL 2, which will be next week's exercises). This language is explained in W3C's OWL Web Ontology Language Reference, which may be a valuable resource for these exercises. OWL Web Ontology Language Overview contains a list of the constructs available in RDFS and the different dialects of OWL 1: OWL lite, OWL DL and OWL Full. See also W3C's "portal" on OWL.

You may use Protégé as your editor, but you are also welcome to use a plain text editor to the exercises. Note that there are different OWL languages and that different editors have different tastes. If you are using Protégé as editor, consult the Protégé pizza tutorial. If your using a plain text editor, use the OWL validator and try also regularly to open your file in Protégé. If you have problems using Protégé, consult the Protégé OWL Tutorial.

The OWL vocabulary we will use is listed below. The list is a slightly compacted version of the one found on OWL Web Ontology Language Overview. Almost all items in the list will be put to use in these exercises.

- RDFS Features: `Class`, `rdfs:subClassOf`, `rdf:Property`, `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`, `Individual`
- Header Information: `Ontology`, `imports`
- Annotation Properties, `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`, `rdfs:isDefinedBy`, `AnnotationProperty`, `OntologyProperty`
- Class Axioms: `oneOf`, `dataRange`, `disjointWith`, `unionOf`, `complementOf`, `intersectionOf`
- (In)Equality: `equivalentClass`, `equivalentProperty`, `sameAs`, `differentFrom`, `AllDifferent`, `distinctMembers`
- Property Characteristics: `ObjectProperty`, `DatatypeProperty`, `inverseOf`, `TransitiveProperty`, `SymmetricProperty`, `FunctionalProperty`, `InverseFunctionalProperty`
- Property Restrictions: `Restriction`, `onProperty`, `allValuesFrom`, `someValuesFrom`,
  `minCardinality`, `maxCardinality`, `cardinality`, `hasValue`
- Datatypes: XSD datatypes

For each of the modelling exercises below express the exercise text as a set of description logic (DL) axioms.

2.1 Exercise

Make a new ontology file. Give it the namespace

`http://www.ifi.uio.no/INF3580/v18/family.owl#`

Import the family RDFS file you wrote in last week's exercise.

2.1.1 Tip

Note, as mentioned in an exercise in last week's exercises, not all ontology editors and reasoners interprets manages to handle RDFS as OWL, so you might want to convert your family RDFS file to OWL. Changing all instances of `rdfs:Class` to `owl:Class` and instances of `rdf:Propery` to either `owl:ObjectProperty` or `owl:DatatypeProperty` should take care of most convertion problems.

2.2 Exercise

State that a person has at least one father and one mother.

2.2.1 Tip 1

The exercises are formulated in normal language on purpose. It is up to you to decide how this is best expressed in OWL.

2.2.2 Tip 2

My solution (yours may be different) as a DL axiom:

$$Person \sqsubseteq \exists hasFather.Person \sqcap \exists hasMother.Person$$

2.3 Exercise

State that a person can only have one mother and only one father.

2.4 Exercise

State that a woman can only have female as gender, and a man can only have male as gender.

2.5 Exercise

State that nothing can be both male and female.

2.6 Exercise

Define the gender so that there can only be the genders man and woman.

2.7 Exercise

Explain what disjointness is. For all pair of classes in the family ontology, add the correct disjoint axioms.

2.8 Exercise

State that a person is either a man or a woman, but not both.

2.9 Exercise

Explain what inverse properties are. For all the properties that exist in our ontology, add the correct inverse property axioms. You are not supposed to add new properties, only state that a property is the inverse of an other property if they already exist in the ontology.

2.10 Exercise

Explain what it means for a property to be transitive or symmetric.

For all the properties in our ontology, if it is natural, state that they are transitive and/or symmetric.

There is no standard way of asserting characteristics for properties in DL, so you may skip this part. The more or less *common* way of assering that a property P is asymmetric, symmetric, reflexive, reflexive or transitive in DL literature is $\mathsf{Asym}(P)$, $\mathsf{Sym}(P)$, $\mathsf{Ref}(P)$, $\mathsf{Irr}(P)$ or $\mathsf{Tra}(P)$, respectively.

To say that two properties P_1 and P_2 are disjoint is commonly done in DL literature with $\mathsf{Dis}(P_1, P_2)$.

2.11 Exercise

Is a subproperty of a transitive property necessarily also transitive? Explain why / why not?

2.12 Exercise

Is a subproperty of a symmetric property necessarily also symmetric? Explain why / why not?

2.13 Exercise

Explain what it means for a property to be inverse functional.

For all properties in our ontology, state that they are inverse functional if you believe that is correct.

3 OWL metrics

3.1 Exercise

Make a java program which loads an OWL ontology and lists

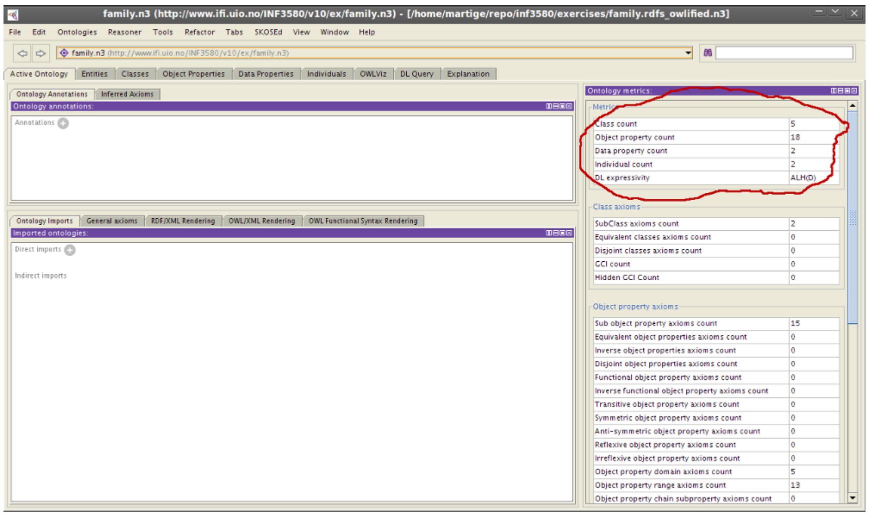
- the number of classes,
- the number of object properties,
- the number of datatype properties,
- the number of individuals and
- the DL expressivity of the ontology.

Use a Pellet reasoner to do your reasoning.

3.1.1 Tip

You should get the same results from your program as you get when loading an ontology in Protégé.

I used the Pellet API to get hold of the expressivity of the ontology, using the classes `JenaLoader` and `KnowledgeBase`.



Metrics

Class count	5
Object property count	18
Data property count	2
Individual count	2
DL expressivity	ALH(D)

Class axioms

SubClass axioms count	2
Equivalent classes axioms count	0
Disjoint classes axioms count	0
GCI count	0
Hidden GCI Count	0

Object property axioms

Sub object property axioms count	15
Equivalent object properties axioms count	0
Inverse object properties axioms count	0
Disjoint object properties axioms count	0
Functional object property axioms count	0
Inverse functional object property axioms count	0
Transitive object property axioms count	0
Symmetric object property axioms count	0
Anti-symmetric object property axioms count	0
Reflexive object property axioms count	0
Irreflexive object property axioms count	0
Object property domain axioms count	5
Object property range axioms count	13
Object property chain subproperty axioms count	0

Figure 1: Protégé screenshot with its relevant metrics section marked.

3.2 Exercise

Test the metrics of your family ontology.

3.2.1 Tip

Note that if your file uses RDFS class or properties, you can have trouble getting the results you expect from Jena, so it is smart to convert the relevant RDFS constructs to OWL. This is easily done manually, as explained in an eariler exercise for this week, or you can open the file in Protégé and save it, which should convert it to OWL.