

EXERCISES INF3580 SPRING 2010 WEEK 8

Martin G. Skjæveland

19 March 2010

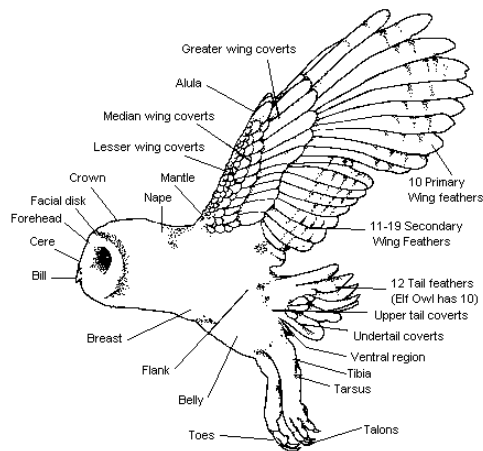


Figure 1: A bird.

This document contains exercises made for INF3580. Please send any comments, errors, bug or improvement reports to this exercise set to martige@ifi.uio.no¹. Feedback is most welcome! Alphabetically thanks to Audun Stolpe, Espen H. Lian, Martin Giese and Rune Dahl for feedback.

The main curriculum for INF3580 spring 2010 is *Semantic Web Programming* by John Hebelers et al., Wiley Publishing, 2009. They have a website with additional articles and all source code used in the book at <http://semwebprogramming.org/>. Auxiliary curriculum is the book *Foundations of Semantic Web Technologies* by Hitzler, Krützsch, Rudolph, CRC Press 2009.

Keep all the work you do for these exercises in a safe place. Setting up a version control system like cvs, svn or git for the work you do is smart. You can create a svn repository on IfI's svn server², see their help section³ for more information. There is also a walk-through⁴ from old INF3120 on how to set up a svn repository and connect it to Eclipse, but news is that you'll need the plug-in subclipse⁵ to make it work. Please contact me if you have any smart tips to share.

¹<mailto:martige@ifi.uio.no>

²<https://www.ifi.uio.no/system/svn/>

³<https://www.ifi.uio.no/system/svn/help.cgi>

⁴http://www.uio.no/studier/emner/matnat/ifi/INF3120/h06/studentarbeider/Prosjektoppgave/SVN_i_Eclipse.pdf

⁵<http://subclipse.tigris.org/>

8 OWL

Read

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

Supplementary reading:

- Protégé OWL Tutorial⁶
- OWL Pizzas: Practical Experience of Teaching OWL-DL:Common Errors & Common Patterns⁷

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.

8.1 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://www.co-ode.org/ontologies/pizza/2007/02/12/>
- <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>

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.

8.1.1 Exercise

Open the pizza ontology⁸ 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.

8.1.2 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?

⁶<http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>

⁷http://www.cs.man.ac.uk/~rector/papers/common_errors_ekaw_2004.pdf

⁸<http://www.co-ode.org/ontologies/pizza/pizza.owl>

8.1.3 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?

8.1.4 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.

8.1.5 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.

8.1.6 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.

8.2 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 Reference⁹, which

⁹<http://www.w3.org/TR/owl-ref/>

may be a valuable resource for these exercises. OWL Web Ontology Language Overview¹⁰ 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 OWL¹¹.

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 validator¹² and try also regularly to open your file in Protégé. If you have problems using Protégé, consult the Protégé OWL Tutorial¹³.

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 Overview¹⁴. 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.

8.2.1 Exercise

Make a new ontology file. Give it the namespace

`http://www.ifi.uio.no/INF3580/v10/ex/family.owl#`

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

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

¹⁰<http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2>

¹¹<http://www.w3.org/2004/OWL/>

¹²<http://www.mygrid.org.uk/OWL/Validator>

¹³<http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/>

¹⁴<http://www.w3.org/TR/2004/REC-owl-features-20040210/>

file to OWL. Changing all instances of `rdfs:Class` to `owl:Class` and instances of `rdf:Property` to either `owl:ObjectProperty` or `owl:DatatypeProperty` should take care of most conversion problems. Protégé converts from RDFS to OWL just fine it seems.

8.2.2 Exercise

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

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.

Tip 2 My solution (yours may be different) as a DL axiom:

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

8.2.3 Exercise

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

8.2.4 Exercise

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

8.2.5 Exercise

State that nothing can be both male and female.

8.2.6 Exercise

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

8.2.7 Exercise

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

8.2.8 Exercise

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

8.2.9 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.

8.2.10 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.

8.2.11 Exercise

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

8.2.12 Exercise

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

8.2.13 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.

8.3 The Simpson family contd.

Now we shall use the extended family vocabulary with the Simpsons file.

8.3.1 Exercise

Open the `simpsons.owl` file you might have created in a previous exercise. Import the Family OWL ontology.

When editing your ontology, remember to choose the correct file to work on in the “Ontologies” menu if you are using Protégé as editor.

8.3.2 Exercise

State that all the named Simpsons characters, i.e., not the anonymous individuals, in the `simpsons.rdf` file are distinct.

8.3.3 Exercise

Explain what the unique name assumption, UNA¹⁵ for short, is.

Assuming our family and simpsons data were represented both in a formalism having a UNA and one without a UNA, write questions/queries that would return different answers in the two formalisms.

Tip In OWL there is no UNA, while relational databases have a UNA. If you are comfortable using SPARQL and SQL as query languages, please do.

8.3.4 Exercise

Explain what the open world assumption is.

8.3.5 Exercise

Apply reasoning and answer the following questions.

- Why is not Maggie inferred as the Daughter of Homer and/or Marge?
- Why is not Homer inferred as the husband of Marge?

8.4 OWL metrics

8.4.1 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.

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.

¹⁵Note: this is not an official acronym.

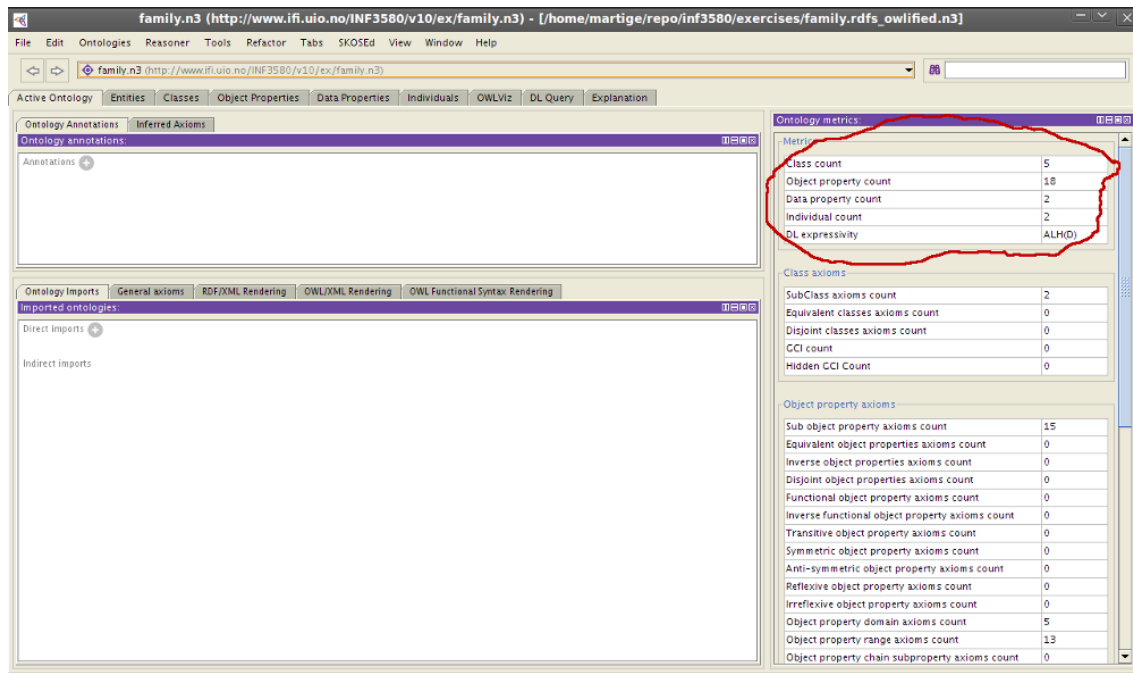


Figure 2: Protégé screenshot with its relevant metrics section marked.

8.4.2 Exercise

Test the metrics of your family ontology.

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 earlier exercise for this week, or you can open the file in Protégé and save it, which should convert it to OWL.