Building ontologies hands-on

– BarCamp lab at ISAO 2023 –

C. Maria Keet
University of Cape Town, South Africa

The aim of this lab is to introduce the participant to creating an ontology, 'debugging' it, and exploring improving the ontology. It focuses on the following tasks:

1. Adding and editing knowledge to an ontology;
2. Familiarising oneself with the ontology language features and the automated reasoner;
3. Finding the root cause of mistakes, which may be both logical ones and undesirable deductions, and correcting them;
4. Exploring methods and tools to improve the representation of the knowledge.

No prior ontology development experience is necessary, though helpful. Since any summer school is a 'crash course', the difficulty of the exercises in Sections 1 and 2 increases quickly and, depending on your knowledge and skills, may take more time to complete than the 1.5h timeslot of this lab. It is possible, though not optimal, to jump straight to the exercises in Sections 3.

Preliminaries

Exercise 0.1. Install your ontology development environment of choice, if not already done so, and acquaint yourself with the software. Look at the menu options and editing panels. If you installed Protégé, you also can add new tabs and view (Click 'Window' - 'View' or 'Window' - 'Tabs'; for views, the mouse pointer will change and you'll need to click to place that view somewhere on the open tab to be able to see it).

1 Adding knowledge to an ontology

Exercise 1.1. The following tasks aim to assist with exploring adding content to a new 'test ontology' that uses different language features and therewith different OWL species.

a. Create a new ontology, give it a name, and save it in RDF/XML.
b. Add the knowledge that bicycles are vehicles; i.e., create two classes, bicycle and vehicle, and make the former a subclass of the latter.
c. Add $\mathsf{bicycle} \sqsubseteq \exists \mathsf{hasComponent.Wheel}$, or: that each bicycle has 'at least one' ('some' in Protégé) wheel as component.
d. Save the file in RDF/XML format (the official exchange format), and also each time before you run the reasoner (if you're using Protégé, as newer versions might freeze at times).
e. Download the OWL classifier from `https://github.com/muhummadPatel/OWL_Classifier`, open it, open your ontology ('File' - 'load..') and inspect the OWL species. (note: this may or may not work fully, depending on whether you have installed Java and can run `.jar` files and if so, which version.)
f. Add $\mathsf{Bicycle} \sqsubseteq \geq 2\ \mathsf{hasComponent}.\top$. (note: the $\top$ in Description Logics (DL) notation is the same as `owl:Thing` in OWL)
g. Reload your ontology in the OWL classifier and inspect the OWL species.
h. Update the previous axioms with the following one: $\mathsf{Bicycle} \sqsubseteq =2\ \mathsf{hasComponent.Wheel}$. (note: the $=$ in DL is the same as `exactly` in Protégé).
i. Reload the ontology in the OWL classifier and inspect the OWL species and violations. What is the main difference, if any?
j. Declare hasComponent to be transitive. Save your ontology, and either run the reasoner ('Reasoner' - 'start reasoner') or check the file again with the OWL Classifier and try to explain the tool's behaviour/output. How is this different from the previous outputs?

Exercise 1.2. The following tasks are intended to experiment with adding content to the 'test ontology' and to try out automated reasoning. If you're using Protégé: it includes an 'explanation' feature that lists the axioms involved in deducing what it deduced, which shows up highlighted in yellow, and will help pinpointing errors later on as well, which can be accessed by clicking on the '?' element.

a. First, uncheck the 'transitive' checkbox on hasComponent so that we can use the DL reasoner again.
b. Add $\mathsf{Cycle} \equiv \exists \mathsf{hasComponent.Wheel}$. Run the reasoner and observe the deductions. Can you explain what was deduced and why? (note: if you have used the reasoner before in Protégé in this session, then click 'synchronize reasoner'.)
c. Declare vehicle to be the domain of hasComponent.
d. Add the class motorised vehicle as a subclass of vehicle and declare it disjoint from bicycle.
e. Create a new class electric bicycle and declare it to be a subclass of both bicycle and motorised vehicle. Run the reasoner and try to explain the deductions in your own words.
f. Add $\mathsf{Wheel} \sqsubseteq \exists \mathsf{hasComponent.Spoke}$. Run the reasoner and try to explain the deductions, i.e., describe why it deduced what it did.

2 Finding and correcting mistakes

Exercise 2.1. In this exercise, we're going to fix the 'mess' we created in the previous exercise in such a way that the ontology is within OWL 2 DL expressivity, is not incoherent (i.e., has no unsatisfiable classes), and has no undesirable deductions.[1]

a. Let's first fix the unsatisfiable class, electric bicycle. What is the root cause of it being unsatisfiable? Consider:
   - Which axiom(s) is (are) the culprit(s)? Several valid arguments are possible.
   - We know that electric bicycles do exist (and your lab facilitator rides one, even!), so they deserve to be in the ontology somehow. Write down at least two options how to represent it differently, which need not necessarily be equally good yet not implausible either.

   Then discuss with your lab partner what the best solution would be, and why, and implement it in the ontology.[2]

b. We'll tackle the undesirable deduction now: Wheel $\sqsubseteq$ Vehicle is clearly wrong with respect to reality, even though logically correct given what we have represented in the ontology. This means we either made a mistake in our understanding or we understood it right but did not add at least some of the knowledge in a way that we should have done. In investigating the issue, consider:
   - Which axiom caused the deduction?
   - Does this axiom hold in all possible worlds?
     - If yes: then how should the property of Wheel be changed?
     - If no: then how should the property be changed?

   Discuss with your lab partner what the best solution would be, and why, and implement it in the ontology.

Exercise 2.2. We'll explore peculiarities of the language and automated reasoning that beginners generally don't expect, with especially when they have a background in databases. They involve individuals in the ontology.[3]

a. Add the following instances: electric bicycle(eb1), motor(m1) and motor(m2).
b. Relate the instances as follows: hasComponent(eb1,m1) and hasComponent(eb1,m2). Save your ontology and run the reasoner.
c. Why is eb1 seemingly allowed to have two motors even though the class expression states that each electric bicycle has as component exactly 1 motor?
d. Click on the individual m1 or on m2 and examine the inference.
e. Add motor(m3), declare it as different individuals from motor(m1), and add has-

[1] If you did not do the exercises of Section 1, then you may use the OWL file I created for the lab, which is available at `http://www.meteck.org/teaching/ontologies/dubiousbicycle.owl`

[2] If you did not manage to solve the unsatisfiable class issue, or simply want to see a possible (not necessarily the best) solution, then you may inspect the OWL file I created for the lab, which is available at `http://www.meteck.org/teaching/ontologies/dubiousbicycle1.owl`.

[3] If you did not do the exercises of Section 1 or did not manage to resolve the issues in Exercise 2.1, then you may use the OWL file I created for this exercise, being `http://www.meteck.org/teaching/ontologies/dubiousbicycle2.owl`.

Component(eb1,m3). Save your ontology and run the reasoner. What happens? (and read the text; don't click it away!)

f. Remove hasComponent(eb1,m3).

Exercise 2.3. This exercise zooms in on the so-called *RBox*, or things we can do with object properties, and more automated reasoning.[4]

a. Add the following instances: spoke(s1), Wheel(w1) and bicycle(b1).
b. Add partOf as object property and declare it to be transitive. (bonus question: why can't we just reuse hasComponent for that?)
c. Add that spoke is a part of wheel and that wheel is a part of bicycle, and do so likewise for s1, w1, and b1.
d. Run the reasoner and inspect the property assertions of s1. Why did it deduce that?
e. Add participatesIn as object property, the chain partOf $\circ$ participatesIn $\sqsubseteq$ participatesIn (in Protégé, use a lowercase 'o'), the class cycling and instance cycling(tdf1), and then participatesIn(b1,tdf1).
f. Run the reasoner and inspect the property assertions of s1 again. What has changed and why? And what about w1?

3 Additional improvements

There are multiple methods, tools, and guidelines to improve on an ontology. They may involve 'cleaning up' orphan classes that are never used, removing duplicate knowledge, adding more constraints, and more. Since this is only a 1.5h lab, these exercise are but a small sampling. They can be done in any order.[5]

Exercise 3.1. As with software development, good housekeeping of an ontology is better than the alternative. For instance, adding an annotation to a class helps understanding it, especially if not everything could be represented. A well-known set of heuristics for that is OOPS! for which there's a web-based tool that scans the ontology on such matters. Test your ontology with it at `https://oops.linkeddata.es/`. Then examine the output:

a. Do you agree with all the issues OOPS! raised? If not: why not?
b. Which one(s) was (were) most useful to improve your ontology?
c. In case OOPS! is offline or you do not have internet access, then consider the following issues, which are among the ones that came up with my `dubiousbicycle2` file:

[4]If you did not do the exercises of Section 1, did not manage to resolve the issues in Exercise 2.1 or didn't do Exercise 2.2, then you may use the OWL file I created for this exercise, being `http://www.meteck.org/teaching/ontologies/dubiousbicycle3.owl`.

[5]If you did not do the previous exercises and just want to try explore improving an ontology, you may use the test file available at `http://www.meteck.org/teaching/ontologies/dubiousbicycle5.owl`.

i. It's good advice to declare domain and/or range axioms, as OOPS! also mentions, yet there were infelicities like we've seen in Exercise 1.2c and Exercise 2.1b. Enumerate all pros and cons of declaring domain and range axioms.
ii. hasComponent has no inverse declared, yet OOPS! suggests doing so. Why should you; or should you not do so? (Hint: the answer has both a logic and ontological aspect to it.)
iii. The file uses different naming conventions of the vocabulary and not having declared the IRI properly. What is a good naming scheme?

Exercise 3.2. We will now align the test ontology to a foundational ontology, being BFO 2.0. You may use either the diagram or the BFO Classifier tool, which are available from `https://github.com/mkeet/BFO2DecisionDiagram`.

a. Either (1) open the latest version of your test ontology on bicycles in the ontology editor and add some more entities, including at least Tour and Biker, or (2) open the facilitator's cleaned up and extended version, available from `http://www.meteck.org/teaching/ontologies/dubiousbicycle6.owl`
b. List the classes that are a direct subclass of `owl:Thing`. These are the entities that will have to be aligned.
c. If you do not use the BFO Classifier tool, then get BFO v2.0 and import it into your ontology yourself first. If you will use the tool, you still may wish to import it manually, but it is not a requirement (the tool can do it for you).
d. For each entity in the list:
   i. Start at the top and continue answering the questions until either 1) you reached the end and there is no further question, or 2) you don't know what the answer to the question should be.
   ii. Write the entity it aligns to next to the name of the class from your test ontology and any further considerations, such as a copy of the question history or why you couldn't answer the questions or where along the path you may not have been certain about the answer you gave.
   iii. Import the axiom (if using the tool) or add the subclass declaration manually in your ontology (if using the diagram only).
e. Assess the alignments among yourselves in the lab. If you're doing this exercise alone, you may compare to `http://www.meteck.org/teaching/ontologies/dubiousbicycle7.owl`, which definitely has one arguable alignment to probe further and discuss.

This completed the lab. If you'd like more exercises (and answers), then try the exercises in Chapters 4, 5, and 6 of the ontology engineering textbook, which is available at `https://people.cs.uct.ac.za/~mkeet/OEbook/`.