Practical Session 5 – Ontologies

Today we are going to extend a small ontology about courses at IDMC to see how to add to a simple vocabulary some logical expressions and see how this affects inferences, i.e. the new information that can be derived from those logical expressions.

Note: We will use blazegraph again today. If you are having trouble making it work, please ask for help! A shared version of blazegraph is still available at `http://212.227.190.14/blazegraph/`.

Note2: We will also use Protégé today and it will be necessary to install it.

T1: In Blazegraph, create a new namespace with inference (you can, for example, call it "IDMC" or "SWLab5") and load into it (make sure the new namespace is selected in Blazegraph) the file at `https://mdaquin.github.io/d/idmc.ttl`. Have a look at what the file contains as Blazegraph shows it to you, and using a SPARQL describe query, inspect the entities:

- `<https://idmc.univ-lorraine.fr/data/mathieu>`
- `<https://idmc.univ-lorraine.fr/data/SWLecture5>`

T2: At `https://mdaquin.github.io/d/idmc_voc.ttl` you will find an RDF Schema vocabulary for the small graph previously loaded. Open it in a text editor and look at what it says. Draw a quick diagram of the classes and of the way properties relate them.

T3: Load the vocabulary onto the previously created namespace in Blazegraph. Inspect again the two entities from T1. What changed and why?

T4: We are now going to use Protégé to edit the vocabulary and add some logical definitions in OWL. Go to `https://protege.stanford.edu/`, download and install Protégé. Start it and open the `idmc_voc.ttl` file. Check how things from the vocabulary appear on the Protégé interface.

T5: Below is a list of things to change in the ontology using OWL operators (don't forget to save):

- Declare that the property `givenBy` is the inverse of the property `gives`: In the entities tab, subtab 'Object Properties', select `givenBy` and add `gives` as its inverse.
- Create a new property `attendedBy` as the inverse of `attends`.
- Define that the class `SCLectureInEnglish` is the class of things which are at the same time a `SCLecture` and `LectureInEnglish`: In the class subtab, select `SCLectureInEnglish` and add an equivalent class. Use the class expression editor to declare it using the `and` operator between `SCLecture` and `LectureInEnglish` (note that ctrl+space activates autocompletion in Protégé).
- Define `LectureInEnglish` as a `Lecture` that has for language (property `inLanguage`) English (`value english`)
- Define `NLPLecture` as a `Lecture` attended by someone (`some`) enrolled in some NLPSemester.
- Define `SCLLecture` as a `Lecture` attended by someone (`some`) enrolled in some SCSemester.
- Make `NLPLecture` a subclass of `LectureInEnglish` to indicate that an NLP lecture is necessarily in English.
- Say, by adding an expression of which `Lecture` is a subclass, that a lecture is given by exactly one person and that lectures are given only by people who have for role (property `hasRole`) lecturer (`value`).

T6: Open the file containing the ontology (`idmc_voc.ttl` if you did not rename it) and check how the expressions you have entered in Manchester syntax were translated in turtle.

T7: Load the ontology into the namespace created at T1 and check the same entities as before. What changed and why?

T8: Some things should have changed, but because the inference capabilities of blazegraph are limited, they did not. Do you know what they are? The file at `https://mdaquin.github.io/d/idmc_ontoindi.ttl` contains a clean version of the ontology together with the elements in the original file (`idmc.ttl`). Load it into Protégé and activate one of its reasoners (e.g. Pellet) to check what it can infer what Blazegraph could not.

Project

For the project, you are asked to build a knowledge graph of children stories. The idea is that your knowledge graph could be used by someone researching a particular theme, plot, type of character, or other aspects of children stories to find relevant ones, to compare stories with each other on those aspects, or to analyze trends in the way stories have evolved over time and cultures. As the lectures and practical sessions go, we will learn more about how that could be done and how we could use it.

At the end of all the practical sessions, you will have to submit:

1. The RDF code of the knowledge graph.
2. A short report briefly describing the steps you have gone through, the choices you have made, any SPARQL query you have used, and a description of any code you might have written.

So, keep notes of what you do and find!

The following task is to be started at the end of this practical session and completed before the next one.

You should have, last time, made an initial diagram of the schema of your knowledge graph. This should be the basis for an ontology for your knowledge graph.

1. Start by taking the classes and properties in this diagram and representing them in a new ontology in Protégé.
2. Write down some (logical) definitions of the classes and add them to the ontology in Protégé.
3. Look at your properties. Make sure they have valid domains and ranges and identify whether they are symetric, functional, or transitive in Protégé.
4. Try adding individuals and running a reasoner to see if the expected inferences are actually realized.