Project 1 (Description Logics)

The deadline for the Description Logics project will be 23 November, at 6pm (Amsterdam time) . This deadline is *hard*: out of fairness to other students, if you hand in after the deadline your mark will be capped to the lowest pass mark. Details of how to hand will be announced closer to the deadline.

You will have to work in groups of 3 students, which will be randomly assigned by us. The project consists of 3 main parts, some of which can only be started once the corresponding teaching material has been made available. Some parts can already be addressed earlier - I marked the weeks in which you should look at them **in purple.**  
  
The final submission will consist of a report in PDF format, as well as a zip file with your implementation (ontology and source code). **You should not wait until the end with writing the report** - rather, I would recommend already documenting what you have done in the respective parts after you have finished them.

**Part I: An Ontology for a Restaurant**

The story: You want to develop an intelligent system so support restaurants that are specialised in one type of dish (e.g. sushi, soups, hamburgers, bowls, etc.). In particular, that system should have an overview about the different dishes the restaurant has to offer, as well as some knowledge about them (ingredients, properties of these ingredients, etc.). This way, it should be possible to intelligently answer specific queries from customers that want to order something. In particular, such a query might involve dietary requirements, preferences in taste, etc. You have recently attended a very interesting lecture on knowledge representation, which convinced you that description logic ontologies are the perfect formalism to represent and reason about such knowledge.  
  
The aim of Part I is to develop an ontology with Protege that can be used to answer such queries using appropriately defined concepts. You can choose the type of restaurant (hamburgers, salads, tacos, etc.), but you cannot build an ontology about pizzas (for obvious reasons). In fact, you are invited to use the pizza ontology for inspiration.

**Before week 2**of the lecture, you will not yet be assigned to a group, but you can already think about what type of restaurant you would be interested in, what kind of questions you would like to ask an intelligent system about the menu of that restaurant, and what things would be relevant to formalise.  
  
**During week 2**of the lecture, you will already have been assigned to a group, learned about Protege (during the Monday session), and you should start and finish your ontology. Make sure all the logical constructors you have learned about are used in the ontology, and are relevant for classification results. Your ontology should contain at least 20 classes in total, and at least 5 classes with complex class expressions in their logical description.  
  
**During later weeks**you will learn about additional constructors that can be used in your ontology, and should extend your ontology to also use those constructors.

Protege can be downloaded here: [https://protege.stanford.eduLinks to an external site.](https://protege.stanford.edu)

**Part II: Your own EL Reasoner**

Now that you have an ontology, you will also need a reasoner to reason with it. The full expressivity of ALC is a bit challenging to support, and therefore in Part II of this project, you will implement a reasoner for the description logic EL, using the algorithm you have learned in he lecture. This part should be finshed **during week 3**of the lecture (since you learn about the EL reasoning algorithm in that week). Ontology tools are commonly developed in Java. Since not everyone of you might be good at programming in Java, I have prepared a little library called *dl4python* that lets you work with ontologies from Python. Find more about it [**here**](https://canvas.vu.nl/courses/70849/pages/project-1-dl4python). If you rather want to use Java or even Scala, the library will also help you.

The reasoner should be able to compute all subsumers for a given class name. In particular, I should be able to call it from the command line using:

**PROGRAMM\_NAME ONTOLOGY\_FILE CLASS\_NAME**

and then get in the output all class names from the ontology that are subsumers of CLASS\_NAME wrt. the ontology in ONTOLOGY\_FILE. The precise output should be: **1 class name per line, no other output.**

This part of the project will be evaluated using an automated tool, it is therefore important that your software can be used in**exactly the way as described.**

You can test your reasoner with your own ontology, as well as with an additional set of ontologies that can be downloaded [**here**](https://canvas.vu.nl/courses/70849/pages/project-1-ontologies)**.** Some of these ontologies may use expressivity that goes beyond EL, and the ontologies differ also in size. Thus your implementation will not always be able to infer all the subsumers for any given class. In your report you should also mention how you deal with concepts/axioms that are not supported by your reasoner.

While this project is to be worked on in week 3 of the lecture, you should already install and test the library [dl4python](https://canvas.vu.nl/courses/70849/pages/project-1-dl4python) in **week 2**, to avoid any problems in week 3.

**Part III: The Creative Part**

This is the part of your project where you can finally do something on your own. In Part III of the project, you come up with your own research question or hypothesis. You should then set up an experiment, possibly using the [**provided ontologies,**](https://canvas.vu.nl/courses/70849/pages/project-1-ontologies) to evaluate your research question/hypothesis. Examples of an idea could be: a modification of the algorithm to increase reasoning speed (you would then evaluate whether/how the performance is improved), or an extension to support more expressivity (you could then evaluate how many additional subsumers you can find), or a comparison with the reasoning output of other reasoners on the set of provided ontologies. Extending the expressivity of the reasoner is probably the most tricky of these examples - be aware that you won't be able to implement a full ALC reasoner in the provided time. Extensions that are easier to deal with are role inclusion axioms, transitivity axioms or the bottom concept.

Your idea can also be something completely else, if you have a good idea, for example, do something with the interpretations that are a side product of the algorithm. If you are unsure, feel free to ask. Note that you can also use dl4python to access other DL reasoners: ELK which supports EL with some extensions, and HermiT which supports all the constructs covered in the lecture.

This should be implemented in the last two weeks of the project (**week 3+4** of the lecture).

**Part IV: The Report**

The report should be 10 pages long and cover the three components of the project. If you use latex, use the standard article template. If you use another tool, the format should be roughly as follows:

* 10 point font size with a font like Times
* 4.4 cm margins on each side

Make sure to not only describe what you have done, but also explain the reasons. For the ontology part, you might want to point out some complex class expressions and how they are used to infer something interesting. For Part II, also mention challenges you had to solve while implementing, and how you dealt with them.

**Part V: Peer Reviewing**

In the week after the project, each student will peer review the reports of two other projects. Instructions will be given then. Note that you are not grading the other projects, but rather give the students feedback on what you think about their project.