



Project 2 (Argumentation)

The deadline for the Argumentation project is on 14 December, 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 should work in groups of 3-4 students that you choose yourself. Please use the provided groups in Canvas to make the assignment (join one of the existing empty groups). This time, the project consists of 2 main parts. The aim is to develop your own system for handling abstract argumentation frameworks.

The Input Format

You assume argumentation frameworks to be given in the form of JSON files as in [this example](https://canvas.vu.nl/courses/70849/files/7008663?wrap=1) (<https://canvas.vu.nl/courses/70849/files/7008663?wrap=1>)  (https://canvas.vu.nl/courses/70849/files/7008663/download?download_frd=1) . (<https://canvas.vu.nl/courses/70849/files/7008663?wrap=1>) You are encouraged to create some more examples like this to test your implementation, looking also for interesting borderline cases. If you want to use Python, parsing JSON files with Python is not difficult, see for example [here](https://docs.python.org/3/library/json.html)  (<https://docs.python.org/3/library/json.html>) . But there are parsers available also for many other programming languages.

Part I: Preferred Discussion Games

The first part considers the discussion game presented in the lecture. Namely, you will develop a game that the user can play against the computer, based on a loaded argumentation framework.

The tool is initiated with the file name of a JSON file and a string denoting a specific argument, referred to as the claimed argument. It then starts a *preferred discussion game* that is played between the computer and the user, where the computer plays the role of a proponent defending the given argument, and the user plays the role of the opponent trying to attack the given argument. Proponent wins the game if and only if there exists a preferred extension that contains the claimed argument.

In each round, the proponent outputs an argument, and the user can choose between different options to attack this argument. At the beginning, the proponent outputs the argument given as parameter to the program. The rules of the game are then as follows:

1. The opponent can only choose arguments that attack another argument previously outputted by the proponent. The attacked arguments can be from the previous round, but also from an earlier round.
2. The proponent always has to answer with an argument that attacks the argument that the

opponent selected in the directly preceding round.

3. The opponent is not allowed to use the same argument twice. (The proponent however can.)

The game is over and a winner selected based on the following conditions:

1. If the opponent uses an argument previously used by the proponent, then the opponent wins (because he has shown that the proponent contradicts itself).
2. If the proponent uses an argument previously used by the opponent, then the opponent wins (for similar reasons as in the previous point).
3. If the proponent is unable to make a move, then the opponent wins.
4. If the opponent has no choices left, then the proponent wins.

Part II: Semantics of AFs

In Part 2, you are supposed to implement credulous decision problems on one of the semantics from the lecture. First you pick a semantics to implement. Again, your program is started with two parameters, the first the filename to the AF, the second an argument . It then outputs whether the given argument can be accepted under the selected semantics or not. That is, the aim is to show whether the given argument is credulously acceptable under the respective semantics in a given AF.

Part III: The Report

The final submission will consist of a report in PDF format, as well as a zip file with your implementation. As for Project 1, the report should have at most 10 pages.

The report should motivate the tool in an introduction, describe the implementation - how you implemented it, if applicable, challenges you faced and how you solved them, design decisions and ideas - and illustrate the tool using some example argumentation framework. If you use GPT, specify the tasks for which it was employed and how it contributed to your work. Furthermore, include clear and concise instructions on how to use your tool.

Different to the previous report, there is no room for any research questions or quantitative evaluation. Instead, we ask you to illustrate the correct functioning of your implementation using your own examples, by looking at the relevant cases.

Part IV: 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.