

AI Large Practical

Alan Smaill

School of Informatics

Oct 5, 2016

- ▶ The second assignment is now on-line on the course web page.

Today, we will look through what you are being asked to do as part of this assignment.

As we saw last time, the general area is that of **argumentation systems**, and we will build on an implementation.

- ▶ While the assessment here is on your submitted programs, it will be a good idea to look at some of the associated literature now in preparation for part 3, which will be assessed primarily on a report of further extensions and experiments.

- ▶ One approach to argumentation is with the Carneades framework, which by now has a fair sized literature; this is specifically designed for **legal argumentation**, and different standards of argument that are applied when reaching legal judgements.
- ▶ A bunch of material gathered together (look at sections 1–5):
<http://www.sciencedirect.com/science/article/pii/S0004370207000677>
- ▶ And on argumentation in general, some slides by Besnard and Hunter are useful in giving a bigger picture:
<http://www.ecsqaru.org/ECSQARU2007/elements.pdf>
NB, they start from a more conventional logical approach.

Reminder:

Assignment	Issue	Due	Weight
A1	28 Sep	Fri 14 Oct, 16:00	0%
A2	5 Oct	Fri 11 Nov, 16:00	50%
A3	9 Nov	Wed 21 Dec, 16:00	50%

There is an account of Haskell version of initial system in this paper:

`www.cs.nott.ac.uk/~bmv/Papers/tfp2012_abstract.pdf`

- ▶ Does a good job of distilling the Carneades framework into abstractions that lend themselves to implementation.
- ▶ `https://github.com/ewan-klein/carneades` is a reimplement of the Haskell code in Python; still beta. Probably helpful for you to look at the Haskell version as well as the Python.
- ▶ If (when?) you find bugs in the Python, raise an issue on GitHub or even better, fix the bug and we can incorporate the fix.



- ▶ Initial docstring of module `caes.py` constructs a argumentation model — a Carneades Argument Evaluation Structure (CAES) — and evaluates some arguments in that structure.
- ▶ To use the model to investigate a series of problems, and to set up experiments, it would be convenient to load the information required for initialising a CAES from a file.
- ▶ Assignment 2: Develop extra functionality so that propositions, arguments and other configuration data required for a CAES instance can all be provided via text files.

Our basic notion of argument:

- ▶ Propositions are atomic statements, possibly negated; they can be represented as simple strings, with a boolean flag.
- ▶ An argument consists of a set of propositions as *premises*, a further set of *exceptions* and a single *conclusion*.
- ▶ We read this as saying that **if** the premises are all established, **and** none of the exceptions can be established, **then** the conclusion is justified.
- ▶ Furthermore, an audience is modeled as a set of assumptions, and an assignment of *weights* to each argument.
 - ▶ A premise is established if it belongs to the set of assumptions; some of the proof standards make appeal to the weights of arguments.

- ▶ Design a string-based way of expressing argumentation data, rather than always having to call explicit constructors.
- ▶ You will need to be able to deserialise your input data into the CAES data structures; your choice of how to do this may affect the design of your input syntax.
- ▶ Priority is to make life easier for person trying to write the input data!
- ▶ Design choices:
 - ▶ add classes / functions to caes.py
 - ▶ create new module that imports from caes.py

You should provide 3 test files with between 5 and 30 arguments, and indicate (in comments) what your system returns on selected queries.

It is also a good idea to include a README file in your directory; this will make the tester's life easier.

- ▶ No lecture;
- ▶ Drop-in sessions for queries.