AI Large Practical

Stuart Anderson

School of Informatics

Sept 29 2015

AILP: Assignment 1

- Assignment 1 is now available on the course web page.
- Not required for Assignment 1 but . . .
- look at some of the argumentation literature now, so as to prepare for Assignment 2, which will involve a written report.

Starting point

- Carneades framework:
  - specifically designed for legal argumentation;
  - permits different standards of 'proof' that are applied when reaching legal judgements.
- Main reference (mentioned last week) is Gordon et al (2007): `http://www.sciencedirect.com/science/article/pii/S0004370207000677`
- For argumentation in general, slides by Besnard & Hunter are useful in giving a bigger picture (albeit with a strong logic slant): `http://www.ecsqaru.org/ECSQARU2007/elements.pdf`

A Computational Framework

- van Gijzel, B. and Nilsson, H (2012) 'Haskell Gets Argumentative', `www.cs.nott.ac.uk/~bmv/Papers/tfp2012_abstract.pdf`
- Does a good job of distiling the Carneades framework into abstractions that lend themselves to implementation.
- `https://github.com/ewan-klein/carneades` is a reimplementation 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 send me a pull request.

The task: initialising a Carneades structure from text input

- 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.
- Assigment 1: Develop extra functionality so that propositions, arguments and other configuration data required for a CAES instance can all be provided via text files.

Arguments with exceptions

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.

Assignment continued

- 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`

Future meetings

- There will be **no Wednesday lecture next week**.
- There **will** be a drop-in lab session on Tuesday:
  - 13:00-14:00,
- Use the Wiki to ask questions, respond to questions or make a comment.