AILP (2016) Report

Keqi Li B065105

December 2016

1 Introduction

This report describes work done in the AILP course. It gives the aims and hypothesis that guided the work; describes the algorithms that were implemented; reports the results of experiments that were run; and analyses these results.

2 Aims and hypothesis

The aim of the assignment is to explore the concept of argumentation theory, and to conduct experiments about how a system, that is built on such theory, solves disputations in real life scenarios.

The intention of the extension implemented is:

> to allow analysis of legal disputations in a way that closely matches what happens in real cases.

Argumentation systems are used in Artificial Intelligence related fields. It allows machines to made decisions and judgments from the premises/facts available. However, the system considers not only validating of a claim, but also rejecting it. It weighs on both sides and comes up with a decision. Examples of such arguments include:

- Options for government policies,
- Legal arguments/judgments,
- Individual decisions, etc.

As argumentation theory described, the system is not to find out the truth but the most possible results given the premises and environment, such that, there is no 'right' or 'wrong' in argumentation systems; there is only proponent and opponent.

Carneades is a model of argumentation system and is made specifically for legal arguments. The significant difference between a normal argumentation problem and a legal one is that, legal arguments bear burdens of proof. A burden of proof can be a burden of

persuasion for the opponent and a burden of production for the opponent. It is that the side that carries the burden of proof should formulate an argument to shift such a burden to the other side.

The assignment is built to extend a Carneades model with basic features. With the first extension to ease out development process and the second for burdens of proof, the system should be able to model some cases in a reality context.

3 Algorithms and implementation

For this purpose the following extensions were carried out

1. reading of input from text files
2. integrating burden of proof into the system

3.1 Reading from text files

This extension allows any user/experimenter to input the propositions, arguments, and everything related to a test case by structuring them into a text file. By doing so, the system can construct a Carneades argumentation system from the input and test the acceptability of some proponents.

The extension goes through the following steps:

- **Syntax design - ease of use**: In order for users to easily input the parameters and comment as they wish, a syntax is designed that, a hash sign # will allow a user to comment the rest of the line, where anything preceding the sign will not be affected. I put special efforts to make the whole syntax space-insensitive.
- **Syntax design - types of input**: The system was programmed to detect different types of input and identifies them. Figure 1 shows an overall structure that the user should follow while using the extension.
- **Implementation and Doctest**: Given a prototype of the input style, the implementation just took functions that can be reusable in later implementation to read files. Doctest helped me smooth the debugging process to ensure outputs from the functions.

There are some test cases used at this stage to test the basic functionality and some of them will be introduced and analyzed in the next section.

Figure 1: A shortened example of how the input file should look like.

Untitled — Edited

```
# Case 1 - murder
PROPOSITIONS:
murder
selfdef
-selfdef

ARGUMENTS:
# Please follow such structure:
# P/D argument_name: [premise, premise, …], ~[exception, exception, …] =>
conclusion
# Please follow the exact order above; however, this is made space insensitive.
P arg1: [killing, malice], ~[exclude187] => murder

ASSUMPTIONS:
killing
malice

WEIGHTS:
arg1: 0.5
arg2: 0.5

PROOF STANDARDS:
# The proof standards available are
# 'scintilla', 'preponderance', 'clear_and_convincing'
# 'beyond_reasonable_doubt', 'dialectical_validity'
selfdef: preponderance

TARGET:
# For Burden of Proof case, just use one single target
# Can use - to mean negation
murder
```

3.2 Second Extension - Burden of Proof

As Carneades is specifically for legal arguments, it is crucial for such a system to implement a way to consider the burdens of proof. The changes made to the previous extension range from modifying the input syntax to creating new functions.

The input syntax is changed in only one way, that is in front of every argument, a **P**(Prosecution) or **D**(Defense) should be indicated. This way we know what this argument belongs to and the arguments that belong to the same side can be grouped into a bracket and the system decides when to introduce them.

Such legal context comes with critical questions that results in shifting the burden of proof. Some of them requires even only asking the questions but some need extra premises to support the argument.

The following process show how my extension of the Carneades system works on a legal argumentation scenario, where an opponent and a defendant both have a set of evidence to put in and then they will work on by claiming arguments to observe the outcome of these evidences.

1. One side uses an argumentation scheme, the burden that this side bears is to
   (a) give evidence for the ordinary premises, or
   (b) resolve the challenged assumptions (not implemented in my system).

If this side successfully shifts the burden of proof, the burden will be on the other side.

2. Such side can find some ways to defeat the arguments

   (a) by giving evidence to the exceptions of the opposite side's argument, or

   (b) by giving evidence to a counter-argument

   The burden successively shifts until the side with the burden runs out of arguments or could not come up with any argument to defeat the opposite side.

This process is a high level description of my implementation. In my code, the process runs in a more detailed way:

- At the beginning, find the evidence from the proponent as the burden of proof lies with the proponent.
- From this stage, the bearer propose argument in this order (**Selection Criterion**) to give evidence:

  1. Prove the ordinary premises of its own argument.
  2. Try rebutting by proving a counter argument, although this way the results will be dependent on the assigned weight of the premises.
  3. Try to prove the exceptions of the opposite side's arguments.

  This selection criterion is followed so it is a completely automatic system that can always find the argument to step forward, instead of successively listing the arguments and restricting the user.

4 Experiments and results

4.1 Choice of experiments

For the first extension I chose some scenarios to test the functionality of the extension. These can be represented in a set of arguments to give a look at, with additional information attached to the right. The first case is a personal decision in Table 1.

arg1: $[finishTasks, sleepy, free, feelGood] \Rightarrow go$	go to bed
arg2: $[task1, task2] \Rightarrow finishTasks$	finished all tasks
arg3: $[tired] \Rightarrow sleepy$	feeling sleepy
arg4: $[coffee]$, $[smallAmount] \Rightarrow -sleepy$	drank coffee, not sleepy
arg5: $[kidAsleep]$, $[kidGrownUp] \Rightarrow free$	kids asleep, free to sleep
arg6: $[hydrated, warm] \Rightarrow feelGood$	feeling good

Table 1: Extension 1 test case 1

The second case is a scenario of U.S. government asking congress to declare war against an enemy in Table 2.

arg1: $arg1 : [budget, reasonable, public\_approve] \Rightarrow\Rightarrow approve$	Congress approves war
arg2: $[security, good\_conseq] \Rightarrow reasonable$	reasonable war
arg3: $[gdp\_growth, assets] \Rightarrow budget$	enough budget
arg4: $[gdp\_slow] \Rightarrow\Rightarrow -budget$	predicted not enough budget
arg5: $[threat\_clear, harmless]$, $[riot] \Rightarrow good\_conseq$	good consequence for the country
arg6: $[-reasonable] \Rightarrow -approve$	not approve
arg7: $[bad\_conseq, -security] \rightarrow -reasonable$	how can it be unreasonable

Table 2: Extension 1 test case 2

The assumptions and the environment will be explained along with the results in the next subsection. The next case in Table 3 is for **extension 2** and it is extracted from Gordon et al., section 6. It describes a typical legal argument scenario and with the same input. I would like to produce two different results, using different weights, because the weights affect the system such that the shifting of burdens of proof would be different given two sets of weights.

P/D?	argument	explanation
P	$arg1 : [killing, malice]$, $[exclude187] \Rightarrow murder$	murder's argument for *murder*
D	$arg2 : [selfdef]$, $[exclude197] \Rightarrow exclude187$	defendant declares self defense
D	$arg3 : [w1def]$, $[w1unreliable] \Rightarrow selfdef$	defendant proving selfdef as an ordinary premise
P	$arg4 : [w2run]$, $[w2unreliable] \Rightarrow -selfdef$	proponent proving non-self-defense

Table 3: Extension 2 test case (different weights for results)

It is clear that in this scenario, the weights for arg3 and arg4 can affect the results. The proof standard here for *selfdef* is *preponderance* and the available assumptions are *killing, malice, w1def, w2run.*

4.2 Experimental results

- **Extension 1 test case 1**: Since we have *task1,task2,coffee,tired,kidAsleep* and *warm* are the assumptions, and *feelGood* is not present or provable, this conclusion will fail on proving *go* which means going to bed is not supported by the evidence. Note that the proof standards for *sleep* is clear_and_convincing, although it does not make any difference in this scenario.

- **Extension 1 test case 2**: The assumptions here are *security, gdp_growth, public_approve, assets, threat_clear, harmless.* It is clear that this is a all-proponent assumptions such that it should yield True for declaring war. It did decide war to be True. This case combined with the previous scenario showed that the system is not a dummy system that cannot produce sensible outputs choosing from True and False.

- **Extension 2 test case with Weight Set 1**: We already know that the assumptions are *killing, malice, w1def, w2run.* Now the deciding factor will be the weights on arg3 and arg4. In this Weight Set 1, I set the weight of arg3 to be 0.5 and arg4 to be 0.4. As in the implementation of my system, the dialogue will be recorded and printed out after the argumentation. Figure 2 is the dialogue of this run. It shows the shifting of burdens of proof. As the weight shows a clear preference for self-defense, given the process, it does not surprise me that it ends up not accepting the murder.

```
Dialogue State 0:
Current arguments:
Accept murder? ==> False
Burden of Proof: P

Dialogue State 1:
Current arguments:
[killing, malice], ~[exclude187] => murder
Accept murder? ==> True
Burden of Proof: D

Dialogue State 2:
Current arguments:
[killing, malice], ~[exclude187] => murder
[selfdef], ~[exclude197] => exclude187
Accept murder? ==> True
Burden of Proof: D

Dialogue State 3:
Current arguments:
[killing, malice], ~[exclude187] => murder
[w1def], ~[w1unreliable] => selfdef
[selfdef], ~[exclude197] => exclude187
Accept murder? ==> False
Burden of Proof: P

Dialogue State 4:
Current arguments:
[killing, malice], ~[exclude187] => murder
[w1def], ~[w1unreliable] => selfdef
[w2run], ~[w2unreliable] => -selfdef
[selfdef], ~[exclude197] => exclude187
Accept murder? ==> False
Burden of Proof: P
```

Figure 2: A screenshot of the dialogue output from the system when weight for arg3 is 0.5 and that for arg4 is 0.4.

- **Extension 2 test case with Weight Set 2**: While keeping all the other setup identical, in this set we changed the weight of arg4 to 0.6 so the system will show preference for non-selfdef. The result in Figure 3 shows that *murder* is accepted based on the last argument to shift the burden of proof.

```
Dialogue State 0:
Current arguments:
Accept murder? ==> False
Burden of Proof: P

Dialogue State 1:
Current arguments:
[killing, malice], ~[exclude187] => murder
Accept murder? ==> True
Burden of Proof: D

Dialogue State 2:
Current arguments:
[selfdef], ~[exclude197] => exclude187
[killing, malice], ~[exclude187] => murder
Accept murder? ==> True
Burden of Proof: D

Dialogue State 3:
Current arguments:
[selfdef], ~[exclude197] => exclude187
[killing, malice], ~[exclude187] => murder
[w1def], ~[w1unreliable] => selfdef
Accept murder? ==> False
Burden of Proof: P

Dialogue State 4:
Current arguments:
[selfdef], ~[exclude197] => exclude187
[killing, malice], ~[exclude187] => murder
[w2run], ~[w2unreliable] => -selfdef
[w1def], ~[w1unreliable] => selfdef
Accept murder? ==> True
Burden of Proof: D
```

Figure 3: A screenshot of the dialogue output from the system when weight for arg3 is 0.5 and that for arg4 is 0.6.

After all the inspections on the extension 2 (burden of proof), I realized using *preponderance* as the proof standard for *self defense* might not be suitable here. More differentiating proof standards such as *clear and convincing* and *beyong reasonable doubt* can use a margin to prevent a tiny weight difference from changing the results dramatically.

5 Discussion and Conclusion

The system works nicely to simulate legal arguments and can apparently reduce the subjective in the court. However, the design of weight is still largely subjective in my implementation. A possible improvement to design the weight might be machine learning. From the history cases, it is possible to learn the weight parameters that yields higher performance than a subjectively defined weight systems. In conclusion, with a working system like this, the design of weight factors and proof standards still has room for improvement.

6 References

1. Alan Smaill, 'AILP 2016 lecture and material set', 2016, The University of Edinburgh

2. Gordon, T. F., Prakken, H. & Walton, D. (2007), The Carneades model of argument and burden of proof, Artificial Intelligence 171, 875896. URL: http://www.sciencedirect.com/science/article/pii/S0004370207000677