Presentation Script

# Title

* Need to introduce Argumentation Logic first and then my project

# Elevator Pitch

* We will see how we can take a natural deduction proof like this, and turn it into an argument like this
* The glue in-between is Argumentation Logic

# Argumentation Logic 101

* Argumentation Logic acts like a bridge between propositional logic natural deduction and argumentation theory
* It establishes some properties and semantics in order to build this connection
  + We have GAP for propositional logic
  + We have NACC for argumentation
* Ultimately it tries to provide grounds for reasoning in inconstant environments but I didn’t have time to go into that

# Propositional Logic Side

* We have normal derivation, and also proving MRA – ie without RA
* RA (or not introduction) is the rule where we
  + Assume something
  + Prove a contradiction
  + Conclude with assumption’s negation
* RAND derivation are derivations with an outermost application of RA rule
* <show example RAND on the picture>

# Genuine Absurdity Property (GAP)

* A derivation follows the GAP iff
  + All children derivations follow the same property (so it’s recursive)
* This establishes a form of relevance since the missing ingredient here must be the hypothesis
* Note that GAP is defined only over conjunction and negation

# Genuine Absurdity Property Examples

???

# Argumentation Theory Side

* We have a simple argumentation framework
  + Each argument is a set containing the theory plus an extension
  + The attack is a binary relation where the attacker union the victim result in a contradiction (MRA!)
* A defence is taking the opposite stance, ie propose the negation of a part of the attack

# Non-Acceptability Semantics (NACC)

* Two things are non-acceptable with each other iff
  + They are different and
  + There is an attack which
    - Either uses your own arguments against you or
    - All defences against it are useless – ie non-acceptable

# Non-Acceptability Semantics Example

???

# The Point of it All

* These two things are the same for consistent theories

# Argumentation Logic Toolkit – Plan

* The original plan consisted of 7 steps
* <one line per step>
* But the last 3 were scrapped in order to provide an overall better experience for the previous ones
* That is why the new steps are in
* <one line per step>

# Flowchart

???

# Solution Architecture

* The end product is split into three parts
* The core contains implementations of all the previously mentioned steps
* It’s built in Prolog
* The server imports the core and uses it to respond to client queries
* It’s built in Prolog too!
* The client offers an easier experience by wrapping around the core
* It’s built in HTML, JS and CSS

# Step 1&2: Theorem Prover

* The theorem prover is the mule we task with proving or disproving goals given a theory
* <prove T:a, G:a>
* The prover does not prune the search space in order to not exclude potentially useful proofs
* This round-about proof is indeed important in Argumentation Logic
* <save proof>
* <prove T:a&b, G:a>
* But as you can see the effect of this is that we get a ton of results that we need to go through

# Step 1+: Proof Builder

* Thus we saw the importance of being able to input proofs yourself
* This is what the proof builder does, with the aid of a parser and rule checker
* <input proof>
  + !((a&!b)&!c)
  + !(a&b)
  + !(a&c)
  + a; hypothesis
  + b; hypothesis
  + a&b; &I(3,4)
  + \_; \_I(1,5)
  + !b; !I(4,6)
  + c; hypothesis
  + a&c; &I(3,8)
  + \_; \_I(2,9)
  + !c; !I(8,10)
  + a&!b; &I(3,7)
  + (a&!b)&!c; &I(12,11)
  + \_; \_I(0, 13)
  + !a; !I(3,14)
* We can drag this proof to save it
* <save proof>

# Step 3: GAP Check

* This uses the theorem prover to check whether the given proof follows the GAP
* <check GAP>

# Step 3+: Extended GAP Check

* However, GAP does not work with shortcuts
* Shortcuts are very frequent so I thought it would be nice to support them
* Thus I extended the GAP property and proved they are equivalent
* Sibling conclusions can now be taken into account
* The GAP-ness of the copies is not affected hence we don’t need to check it

# Extended GAP Check Proof Sketch

???

# Step 4: Visualizing GAP Proofs

* The Argumentation Logic paper does not indicate how proofs can be visualized
* So after a few shots of vodka I came up with this mapping
* <explain mapping>
* <explain algorithm using picture>
* <explain ironing out>
* <visualize proof>

# Step 4+: Extract Proofs from Arguments

* Instead of moving to Step 5, I thought of closing the loop by providing an inverse function
* <explain algorithm using picture>
* <extract proof>

# Step 4++: Argument Builder

* This complements the proof builder
* It allows a different entry point to the cycle by allowing arguments to be a starting point
* It plays like a mini-game where the user is the opponent and the computer is the proponent
* <build argument>

# Complete Ecosystem

* So this is how everything is connected
* We have the loop made using the visualizer and extractor
* We have two entry points
  + the proof builder (and theorem prover)
  + the argument builder

# Contributions

* Apart from building a toolkit that allows one to learn more about Argumentation Logic I also cooked up a couple of things
* I extended the GAP to work with proofs with shortcuts
* I found a way to visualize proofs by establishing a mapping
* I made the inverse algorithm as well

# Future Work

* This involves mainly improving the theorem prover and possibly changing its focus
* Completing the steps left out from the original plan

# Thank You for Watching!

* Thank you for watching!
* I will now answer any questions you might have!