Journal Report 2
9/9/19-9/13/19
Rahul Mittal
Computer Systems Research Lab
Period 2, White

---

# Daily Log

## Monday September 9

Change sample proofs from being string constants inside my program to being file resources packed into the jar, change the formulae in the proofs to being inside quotes (and also change them from being in the random format I wrote them in to being in a format approximately like what I expect to actually use), run the parser on the new proofs, the output constituency parse doesn't have the formulae grouped nicely under the quotes like I was hoping, now try with the parentheses, it's equally bad, decide to use quotes since parentheses don't seem to offer any advantages, try looking at dependency parse.

## Tuesday September 10

Try to start writing the function to separate sentences into pieces that perform actions, go look at the proofs I found online, realize that it's really difficult to do without actually trying to figure out the actions being performed, scrap that plan, create function to just do the action identification based on the dependency parse looking at the verb nodes, decide for now that four actions exist in the world: defining, claiming, stating a premise, and making a deduction (make an Action enum w/ those four), try looking up how to get the root verb of an IndexedWord (the class for the nodes in the dependency parse), fail, add simple code to recognize verbs that define things and verbs that claim things.

## Thursday September 12

Get tired of Stanford parser taking so long to load every time, modify my program to just load Stanford parser and then read in sentences from the command line since I'm not even looking at relationships between sentences right now, waste 10 minutes because I accidentally entered a sentence with a newline in the middle of it and it messed up the parsing of the next sentence I entered and I didn't realize that and got confused, add case in my identifyActions function where a math verb with can as an auxillary will be recognized as a premise (or a deduction - right now the function tests if the thing is a premise and then if it is also checks if there are things that would make it a deduction, like the words 'so' or 'therefore'), now go back to 'Theorem' part of sample proof, the 'is' verb which is the main verb of the sentence isn't the root of the dependency graph, it's a graph that can apparently have multiple roots so try looking at all the roots, nope it's just the one, try calling the dependencies() function, that gives an NPE, but there's a setRoot function in the dependency graph so try BFSing for the first verb and setting that as the root, nope just gave me a tree with only the verb and then two more trees for the rest of the graph.

## Timeline

| Date | Goal | Met |
|---|---|---|
| September 2 | Run Stanford parser on my sample proof | No, didn't import the proper model files or run it with enough memory |
| September 9 | Run Stanford parser on a couple of proofs | No, but I ran it on mine and looked at a couple of proofs |
| September 16 | Run Stanford parser on more proofs, write a function trying to separate sentences into clauses with different actions | No, abandoning that idea completely |
| September 23 | Finish prototype of function identifying actions and test it on more sample proofs to see what I didn't account for | |
| September 30 | Revise function identifying actions and continue process to the point that 95% of random proofs on the Internet produce the correct output | |

## Reflection

This week went way off of what I expected. To start, on Monday, I found out that Stanford parser doesn't group things in quotes nicely together in a way where they are under their own node that can be separated off from the rest of the sentence, but instead essentially treats them as part of the sentence.

This is my current version of the sample proof, where I've made written the formulae in essentially the format that I expect to actually use, and also put them inside quotes:

Theorem: For integers ”b,u” such that ”b ¡= u”, the amount of ways to place ”b” labelled balls into ”u” labelled urns such that each urn contains at most one ball is ”factorial u / factorial (u - b)”.

Proof: We can choose ”b” urns which will contain a ball in ”u ¿¿ b” ways, and the balls can be matched up with the urns in b factorial ways. Therefore, the total amount is ”(u ¿¿ b) * factorial b = (factorial u / (factorial b * factorial (u - b))) * factorial b = factorial u / factorial (u - b)”.

You can see the formulae essentially look like code, because that's what I plan to use them as - code that is compiled and produces as output a JSON version of the formulae, though that's not relevant to what I'm working on right now.

This is part of the constituency tree produced by the ”Theorem” part of that proof (the part corresponding to 'integers ”b, u” such that ”b ¡= u”'):

```
|   |   |   | (NP
|   |   |   |   | (NP
|   |   |   |   |   | (NNS integers))
|   |   |   |   | (`` ``)
|   |   |   |   | (NP
|   |   |   |   |   | (NN b))
|   |   |   |   | (, ,)
|   |   |   |   | (NP
|   |   |   |   |   | (NP
```

```
|   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   | (NN u))
|   |   |   |   |   |   | (’’ ’’)
|   |   |   |   |   |   | (PP
|   |   |   |   |   |   |   | (JJ such)
|   |   |   |   |   |   |   | (IN that)
|   |   |   |   |   |   |   | (`` ``)
|   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   | (NN b))))
|   |   |   |   |   | (NP
|   |   |   |   |   |   | (JJR <)
|   |   |   |   |   |   | (JJ =)
|   |   |   |   |   |   | (NN u)))
|   |   |   |   | (’’ ’’)))
```

The annoying thing here is that the 'such that' has become interspersed in the part of the constituency tree corresponding to that formula, and it seems like 'u" such that "b ¡= u"' as a whole has become recognized as a noun phrase. This particular case probably won't be a problem for my current task of identifying actions, but it might be a problem with other proofs and most likely will be a problem for doing my future tasks.

Now onto the more important part of what happened this week, which is the shift in how I'm trying to approach the problem of identifying the actions. Last week when I decided to first split sentences into the pieces that performed actions before identifying those actions, it seemed like a good way to divide the work that I was doing. Unfortunately, it seems to be really difficult to decouple identifying which parts of sentences perform distinct actions from other parts and identifying what those actions actually are. Essentially, at least the way I was doing it, I had to try to determine whether a particular part of a sentence was performing an action or not, but since there isn't really a dichotomy between actions and things that aren't actions that I care about, but it's really more like there are fragments that do A, fragments that do B, fragments that do C, and fragments that don't do anything that I care about/fragments that contribute to other fragments doing A/B/C/D/F/X, identifying which ones are of the last type became essentially equivalent to identifying the actions being performed.

Having finished Thursday, I now have a pretty clear idea of how I'm doing the action identification: go down the constituency tree, and when you hit a verb phrase, try to identify which action is being performed by it, possibly with the result that no action is being performed by it. If there is an action being performed, add that node w/ the action to a result list, and keep going down the tree.

I say constituency tree, however in the daily log you'll notice that my work on identifying actions this week was using the dependency graph. The reason I was using the dependency graph, at least I think, was because I seemed to think that it dealt better with the formulae. Right now I don't really know why I thought that, but regardless there are a few problems with the dependency graph that are making me switch back to using the constituency tree: I've been saying dependency graph rather than dependency tree, and that's because the dependency graph is not guaranteed to be a tree. Rather, edges represented dependencies between words, and this means that the entire format of the dependency graph is much less consistent than that of the constituency tree because what's at the top is determined by how dependencies worked out rather than the sentence structure. This manifested itself on Thursday in that all of the sentences I had been looking at up until then had a verb as the "root" of the dependency graph, but trying the following

variant of the 'Theorem' part of my proof:

Theorem: The amount of ways to place "b" labelled balls into "u" labelled urns, where "b,u" are integers and "b ¡= u", such that each urn contains at most one ball, is "factorial u" divided by "factorial (u - b)".

The dependency graph yielded the following:

```
-> Theorem/NNP (root)
  -> :/: (punct)
  -> amount/NN (dep)
    -> The/DT (det)
    -> ways/NNS (nmod:of)
      -> of/IN (case)
      -> place/VB (acl:to)
```

With the actual 'is' verb occurring at depth 8 in the "tree", while the constituency tree yielded the following:

```
(ROOT
 | (S
 |   | (NP
 |   |   | //stuff
 |   | (VP
 |   |   | (VBZ is)
 |   |   | (NP
 |   |   |   | (NP
```

Which is much better.