Journal Report 5
9/30/19-10/4/19
Rahul Mittal
Computer Systems Research Lab
Period 2, White

Daily Log

Monday September 30

Added case for 'We see that ...', Colorado proof 5 finished. Start to create classes for giving more details about sentences, two base classes - Statement and Quantity, Equality class extending Statement, etc., add MathObject base class.

Tuesday September 24

Created Kotlin file for the new method which will replace 'identifyActions', copy/paste identifyActions and start to change it, create StatementNode class, change the if branch for defining into two separate if branches for define and if, think about similarly changing the rest of the function but decide against it, instead decide to use matching trees approach, start doing that, make 'Template' class that stores a constituency tree parsed from a template sentence/phrase that I input and the intended output of Statement/Quantity/MathObject, try out some template sentences to see what output looks like.

Thursday September 26

Changed first template sentence from 'we can choose quantity1 mathobject1 in quantity2 ways' to 'we can choose mathobject1 in quantity1 ways', decide to instead of parsing template sentences into trees, just write out the trees themselves in code, initially try to make functions that can create instances of Stanford CoreNLP's Tree class, that's annoying and stupid, make my own 'Node' class (very creative name), write Python script to autogen functions for each partofspeech and each type of sentence/phrase thing, rewrite existing templates using those functions, add 'TemplateParameter' subclass of Node, finish the rewriting of the existing templates, start to write function to match trees.

Timeline

Date	Goal	Met
September 23	Finish prototype of function identifying actions and test it on more sample proofs to see what I didn't account for	Yes, almost done with making it work for Colorado proof 2
September 30	Revise function identifying actions and continue process to the point that 95% of random proofs on the Internet produce the correct output	No, only finished Colorado proofs
October 7	Modify function identifying actions to include information about statements made, meaning what verb used, what nouns acting/being acted on	No, but layed foundation for doing it
October 14	Create Haskell DSL for converting formulae into JSON and be able to compile/run Haskell program from my main program	
October 21	Get detailing function to work on sample proof + first two Colorado proofs	

Reflection

After this week, I think I have a better approach than before of being able to detail what sentences are doing. My current idea is to determine phrases that have specific meanings in terms of math and combinatorics, see what tree they produce in Stanford parser, then reproduce that tree in my code as a template to be used to check against the constituency trees of actual sentences. This to me seems like a much better approach than the way I was doing the action identification because, among other things, the verbs (and I guess the words in general) used in these proofs have meaning which is extremely dependent on the immediate context (surrounding phrase), as exemplified by the distinction between, say, 'we choose' and 'we can choose'.

To show how I'm planning on writing these templates, consider this clause from my sample proof: we can choose b urns, which will contain a ball, in c ways

This is the tree produced by Stanford parser for that clause:

```
(ROOT
 | (S
 |   * (NP
 |   *   | (PRP we))
 |   * (VP
 |   *   | (MD can)
 |   *   | (VP
 |   *   |   * (VB choose)
 |   *   |   * (NP
 |   *   |   *   | (NP
 |   *   |   *   |   * (NN b)
```

```
|  *  |  *  |  * (NNS urns))
|  *  |  *  | (, ,)
|  *  |  *  | (SBAR
|  *  |  *  |  * (WHNP
|  *  |  *  |  *  | (WDT which))
|  *  |  *  |  * (S
|  *  |  *  |  *  | (VP
|  *  |  *  |  *  |  * (MD will)
|  *  |  *  |  *  |  * (VP
|  *  |  *  |  *  |  *  | (VB contain)
|  *  |  *  |  *  |  *  | (NP
|  *  |  *  |  *  |  *  |  * (DT a)
|  *  |  *  |  *  |  *  |  * (NN ball))))))
|  *  |  *  | (, ,))
|  *  |  * (PP
|  *  |  *  | (IN in)
|  *  |  *  | (NP
|  *  |  *  |  * (NN c)
|  *  |  *  |  * (NNS ways)))))))
```

The most immediate, top level structure I see is 'we can choose %math object% in %quantity% ways', which led me to create this template:

```
Template(S(
    NP(PRP("we")),
    VP(
        MD("can"),
        VP(
            VB("choose"),
            XN(1)
        ),
        PP(
            IN("in"),
            NP(
                QQ(1),
                NNS("ways")
            )
        )
    )
)) { Equals(Amount(Choice(it.xn(1))), it.qq(1)) }
```

Where QQ is an abbreviation for quantity, and XN is an "abbreviation" for math object.

The last thing I want to talk about is my goal for this coming week. I didn't push it back (I actually added something to it) because this is both a rather important part of my project and should be straightforward and easy. I don't think writing the actual Haskell code will even take a whole period, which is why I added the part about being able to do all of the running and compiling of the Haskell code from my main Kotlin program. That's fairly important as I don't want to have to copy and paste my formulae into a separate window running the Haskell code for all of my sample input, and since I currently don't know how to do it and I'll probably need

to delve into fun, new, unexplored parts of the Java API, it'll probably take at least most of the remaining time this week.