Journal Report 14
1/6/20-1/10/20
Rahul Mittal
Computer Systems Research Lab
Period 2, White

Daily Log

Monday January 6

Look at proofs for phrasings used to describe implications, starting with proof 1, add HangingTherefore subclass of MetaStatement, add "therefore %st%" metastatement template, looking at "by definition" in proof 1, for now just naively adding ByDefinition subclass of MetaStatement and "%st% by definition" metastatement template, moving to proof 2, covered by "therefore" and preexisting "since", has "By the multiplication principle,", slightly less naively adding ByConcept enum, enum has DEFINITION and MULTIPLICATION_PRINCIPLE, got rid of ByDefinition and added By(ByConcept, Statement) subclass of MetaStatement, made necessary fix in syntax template section, added "(by mp %st%—%st% by mp)" metastatement template, added "by definition %st%" option to other template.

Tuesday January 7

Not in school.

Thursday January 9

Looked through proofs 3 to 5, added "%st% so %st%" metastatement template for proof 5, ok now need to figure out howtf to connect disparate statements together, um, first going through and writing what connections should exist, copied proofs from input directory into input_playground directory so I can add non-input stuff to their files, now going through to try to "write down" implications in each proof between statements on different lines, change And to being a subclass of MetaStatement, finished implications for proofs 1 and 2, doing some backofhand calculations on total amount of possible "implication graphs", with minimal assumptions looks like $2^{\binom{n}{2}}$ with n = amount of statements, counting how many statements (not intents) are in each proof (two statements joined by an and in the same sentence count as different statements): 4, 6, 6, 9, 7, finished implications for proofs 4 and 5, and 3 now.

Sunday January 12

Change current main function to main6, copy contents to new main function, replace output file creation with creation of list of statements, create ProofStructure file, create Implication(Int, Int) data class, create list of implications, add j - 1 =¿ j if statement j contains therefore, realize that I need to account for multiple statements in the same sentence, create StatementTag sealed class, create ThereforeTag(Unit) subclass of StatementTag, create ByTag(ByConcept) subclass of StatementTag, create AndTag(Int) subclass of StatementTag, delete j - 1 =¿ j code, change statements to statements1, actually to sentenceStatements, now doing DFS on metastatement structure for each sentence, add addTag and handleStatement inner helper functions, realize that DFS needs to be more complicated to set proper indexes for implications arising from "x and y so z" and similar.

Timeline

Date	Goal	Met
December 16	Finish Colorado proofs, prepare code to accept whole proof as input for demonstration	No, Colorado proofs unfinished, yes, code prepared
Winter Goal	Take in as input a combinatorial proof, with no modifications other than expressing formulae with my formula syntax, and output the claim and all steps of the proof as instances of Statement.	See Journal 12
January 13	Forgot to set a goal	Added some MetaStatement subclass + templates, started conversion of sentences to proper statements and implication finding
January 20	Finish conversion of sentences to proper statements, finish adjacent implications from "Therefore", find similar phrases used in different statements	
January ??	Be able to find all implications between statements in Colorado proofs	

Reflection

So this week was fairly productive considering that it was the first week back from winter break. My forgetting to set a goal for this week drastically affected my ability to meet my goal for this week, but I was able to finish some things and start some important things regardless.

The first thing I did was to make my current collection of MetaStatements and metastatement templates more complete. For some reason, I started working on metastatements during the previous portion of the project even though they weren't relevant to that portion of the project, but they're relevant now so I guess that's fine. Metastatements are based in the lazy but justifiable idea that statements describing relations between other statements are still statements, meaning that I don't need to put more effort creating a new class that has to be parsed. At this point, my catalog of metastatements includes "statement by (definition—multiplication principle—etc.)", "statement and statement", "therefore statement", "statement so statement", and other things equivalent to those. It is important to keep in mind, though, that metastatements are different from other statements, and when my program describes relations between different statements, those statements won't be metastatements, so my program does need to be able to easily differentiate between metastatements and other statements which is the purpose of the MetaStatement class.

The next thing is that I ~~wasted~~ spent some time thinking about the very theoretical idea of just trying all possible sets of implications to see if they created a sensible proof. It's reasonable to assume that if one statement comes before the other in a proof, then only the former can imply the latter, so if you have n statements, making no other assumptions about the structure of the implications, there are $2^{\binom{n}{2}}$ possible sets of implications. The most amount of sentences any of the Colorado proofs had was 9, which would mean 2^{36} possibilities, which is way too much to

reasonably brute-force considering that the creation of each proof will require outputting a proof using IO and running the proof checker on it. That number can probably be decreased considerably by making justifiable assumptions about the amount of implications, the way implications are structured (it might be reasonable to assume that something like 1 =¿ 3, 2 =¿ 4 is not practically going to appear, though I'm not even sure about that), but it's probably better to stick to trying to generate the implication structure for now.

The last thing is what I got started on and need to finish: the conversion of the collection of sentence-statements, some of which will be metastatements, into a "flattened" structure with the non-meta statements, and then the broader finding of implications between those non-meta statements. At this point I know how I'm doing the "flattening" - the data classes that I'm using to represent all of my MathThings (Statements, Quantities, MathObjects) can be though of as a tree, so I can just DFS in the structure of each sentence until I reach the non-meta statements. The identitities and contents of the metastatements are important, so I'm also converting those into "statement tags", where each statement will have certain tags according to what metastatements contained them. For example, a statement contained by HangingTherefore would have a ThereforeTag(Unit), and the 7th statement contained by And(7, 8) would have an AndTag(8) (while the 8th would have an AndTag(7)), etc.

More generally, what I'm working towards is the generation of the relations between statements, the most important component of which is the implications between statements. Immediately after finishing the flattening the first case I'll account for is the simple idea that if a statement is contained by therefore, then the previous statement probably implies it. After that it gets more complicated to figure out however, and so I allocated back time for finding similar phrases used in different statements because I think that's important for determining whether there's an implication there or not.

This isn't representative of most of the work that I did this week, but here's the last sentence in Colorado proof 1:

Therefore the right hand side also counts the desired quantity.

And here's the output for the last sentence in Colorado proof 1 where you can see how the therefore is integrated as a MetaStatement:

```
ParsedStatement(
    statement=HangingTherefore(
        conclusion=ParsedStatement(
            statement=Equals(
                q1=ParsedQuantity(
                    quantity=SideOfEquation(
                        side=RIGHT
                    ),
                    context=ParseContext(
                        from=1,
                        to=5
                    )
                ),
                q2=ParsedQuantity(
                    quantity=HangingAmount(
                        unit=kotlin.Unit
                    ),
                    context=ParseContext(
```

```
                        from=7,
                        to=10
                    )
                )
            ),
            context=ParseContext(
                from=1,
                to=10
            )
        )
    ),
    context=ParseContext(
        from=0,
        to=10
    )
)
```