Journal Report 15
1/13/20-1/17/20
Rahul Mittal
Computer Systems Research Lab
Period 2, White

Daily Log

Monday January 13

Start writing recursive dfs, replace handleStatement function with dfs function, add if/else for is MetaStatement, yeah i should make commas important, finished when case for HangingTherefore, finished when case for And, move up implications variable soI can reference it in the dfs, adding concept of a tag id, using UUID just for overkill, now i can get rid of Unit from ThereforeTag, but I could have done that anyway because i'm making them not data classes, but actually I'm making id a field of the specific classes rather than StatementTag so they are data classes again.

Tuesday January 14

Done with dfs stuff, adding adjacent implications, finished adjacent implications (from "Therefore"), modified tokenizing code so that all punctuation now counts as a token (not including apostrophes), now time for finding similar phrases, really need to refactor this code, created getStatementsAndImplications(sentenceStatements) -¿ (statements, implications) function (tags aren't returned because they're only used internally for finding implications).

Thursday January 16

Writing a general Trie implementation, made trie file, added Trie¡K, V¿ class, added children = map from K to Trie¡K, V¿ field, added terminal = list of V field, making add function, finished add function and therefore finished Trie implementation, add findSimilarSubstrings(statements) function, realize that I need the actual tokens, keep tokenized List¡String¿s instead of throwing them away, get tokenizations that i actually need, add tokenizations argument to those two functions in proof structure, add SentenceTag subclass of StatementTag, convert initial loop in getStatementAndImplications to being index based, do the adding SentenceTag thing, realize that I need to modify how the trie works, adding depth field to Trie, finished, add tags argument to substrings function, finish code adding everything to the trie (by that i mean calling Trie.add with the correct arguments, not implementing Tree.add).

Timeline

Date	Goal	Met
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	Yes, yes, yes but haven't used them for anything yet
January ??	Be able to find all implications between statements in Colorado proofs	
January ?? + 7	Start work on combinatorics foundation in Coq	

Reflection

This week was fairly straightforward in terms of what I did. Starting with the ”flattening” of the sentence trees to statements, I realized quickly (probably not quickly enough) that it would be much easier to use recursive DFS here after my years of only using stack-based DFS. Once I switched to using recursion, writing the DFS became fairly simple.

In terms of the StatementTags, I did realize that I needed to be able to distinguish when two StatementTags belonging to different Statements were the same tag, or more precisely existed for the same reason. This became an issue in the hypothetical of ”A. Therefore B and C.” - B and C would both have a ThereforeTag, so in my previous idea for how I would use the ThereforeTags in finding adjacent implications, A would be found to imply B, and B would be found to imply C. This is obviously not what we want, so by allowing my code to distinguish whether two StatementTags are the ”same”, I can then write (and did write) code that finds a contiguous subsegment of statements with the same ThereforeTag, and then states that they are all implied by the statement preceding the subsegment. I used UUIDs to distinguish the StatementTags, which is overkill, but easy to use/write so it's fine.

On the commas thing - I realized that you could have a very complicated sentence with nested ”and”s and ”so”s, so I decided that it might be useful later on to be able to use commas for clarification, and thus modified my tokenization code so that commas (and most other punctuation) are counted as their own tokens. As I write this, I now realize that I didn't modify the code parsing syntax into List¡SyntaxPiece¿s to allow for use of commas in the syntax, so that modification is currently useless, but with that in mind I will do that first thing in my next work period.

The last thing is the finding similar phrases in different statements. Sentences are small enough that it's completely efficient enough to just compare every substring of every sentence to every

substring of every other sentence (technically subsegments of tokens rather than substrings, since I want similar phrases), but I still used a trie to find this because it was fairly easy to write, seemed like it might be easier to query, and, for example, would allow my code to easily see when the same phrase appeared in more than two statements.

Now that I have the similar phrases (really same phrases right now), I need to figure out how to use them to find implications. A first step I was thinking of might be to, now that my code has the phrases which appear in multiple sentences, find the MathThings which are contained in those phrases (more likely, the MathThings contained in those phrases which aren't contained by other MathThings contained in those phrases, which ideally will be disjoint). I could after that (or instead) create a scoring/ranking system where having a common phrase of length l tokens gives two sentences l^2 commonality points or something, but it's not clear how I would use the commonality score, because two sentences having a lot of phrases in common doesn't immediately imply that one implies the other.

My goals are also kind of weird right now. I didn't change my goal for next week because I don't really know how it's going to go as far as finding those implications, even though it's this week and I should know whether I can achieve it. My goal for two weeks out is also very vague and not really a proper goal because that's moving into a completely different portion of my project, so I don't really know what kind of subgoals I should be achieving. That should become clear as I start work on that, but for right now that goal is vague.