Journal Report 10
11/18/19-11/22/19
Rahul Mittal
Computer Systems Research Lab
Period 2, White

Daily Log

Monday November 18

Use StringBuilder and loop through indices in input string, when '$' reached replace '$...$' with letter, parse formula, and put letter + parsed formula in map, change signature of parseStatement to accept list of tokens + map: Char -¿ Quantity, modify formula and formula statement parts of function to set dp value to actual parsed formula Quantity instead of placeholder Quantity.

Tuesday November 19

Test inclusion of actual parsed formulas, doesn't work, entering infinite loop?, start debugging parseFormula function, uncomment commented print statements, print statements prevent thing from working, but also the expected output lines are not being detected, debug FormulaParser initializer, giving errors while loading Haskell code, fix Haskell code, go back, still hanging, try manually inputting prepared formula input to Haskell code, doesn't work, fix formula, now giving IllegalArgumentException, fix "combination" case to "choose", works, test a couple more cases, works (note: may want to make it so that invalid formulae give an actual error message instead of hanging).

Thursday November 21

Start work on Colorado 1, add Satisfied: Statement, add OfSize: MathObject, add "to exclude" math object template, add "to choose . we can " statement template, made it so that I don't have to explicitly put plural forms for all the math object words, added "there are %xn% %qq%" statement template, added hanging "ways to do this", (note: should add optionals in syntax?), added sides of equation quantity templates, added "desired quantity" as HangingAmount, first Colorado proof done, add RemainingElements: MathObject, add "there are other elements" statement template, add "must be chosen from" statement template.

Timeline

Date	Goal	Met
November 11	Get simplified parser to work on sample proof + first two Colorado proofs	No, but parser itself seems to work so far.
November 18	Make sure simplified parser works on all Colorado proofs	No, focused on restructuring code
November 25	Integrate actual formulae (as opposed to placeholders) into output as Quantity objects, make sure parser works on first 2 Colorado proofs	Yes, but need more functionality to better represent Colorado proof 2
December 9	Make sure parser works on rest of Colorado proofs, parse statements of intent, add optional functionality to syntax, add thing in syntax for "does not"	
December 6	List repetitions of same MathObject, find similar Statements	
Winter Goal	Take in as input a combinatorial proof, with no modifications other than expressing formulae with my formula syntax, and output tthe claim and all steps of the proof as instances of Statement, and an additional data structure representing relations between Statements in thet proof.	

Reflection

This week was kind of the opposite of last week, in that I worked to (and mostly completed) my goal for this week, but also realized that there are some things that I need to do that are important, that I now need to do next week.

Most of this came from working on Colorado proof 2. Aside from me having to make some questionable additions to the Statement and MathObject templates, Colorado proof 2 is problematic because it has statements of intent, like "First, we will count how many subsets of size k include s". I wasn't previously planning on including these statements of intent as things that I parse as a whole (though at some points I was considering treating them similarly to claims), but the problem is that those statements provide additional context that is necessary for the proof to make sense.

For example, after that statement of intent, the proof continues "Since such a subset includes s, there are $k1$ other elements in the subset...". This is a completely unjustified statement without the context that the subset is of size k. Originally, I planned on possibly just trying to extract the MathObjects and Quantitys alone from those statements of intent so that I would have the context, but the way I'm approaching the parsing of this right now, for each substring of the statement, I can potentially have random parsed things that only use a few (or even one) word(s) of the substring and aren't actually what is meant by the statement, and since I'm not just getting the one optimal statement from the whole thing, I don't know which of those parsed things best

represents what is being said.

This makes the solution clear - just treat statements of intent similarly to normal statements, so that they can be parsed as a whole.

The other two additions that I feel are important have to do with the way I'm writing syntax for the templates. Right now, the syntax is pretty simple, just raw text and the "%abbreviation%" things. This is leading to some problems though, in that similar English forms of the same template use slightly different words, and also that some templates have words which don't have to be used but shouldn't really be counted as unused, decreasing the optimality of that match. To fix this, I want to add syntax constructs of the form "[a]" and "(b—c)", where "[a]" indicates that "a" doesn't have to be included but may appear without making the match less optimal, and "(b—c)" indicates that either "b" or "c" can be used there.

The other addition is to add something to the syntax that indicates a certain part of the syntax (which will be optional) for Statement templates that, if it is used, indicates that the Statement should be negated. This is just because variations of "does not" can be included in a statement in different ways and so it doesn't seem possible to have a general "does not" template, meaning that the best way would be an easy way to include the "does not" in each individual Statement template.

To demonstrate the current output of my program, here's a sentence from Colorado proof 1:

To choose a subset of size k, we can instead choose the nk elements to exclude from the subset.

The prepared form:

To choose a subset of size k, we can instead choose the $n k$ elements to exclude from the subset.

And the output:

```
ParsedStatement(
    statement=Equals(
        q1=Amount(
            obj=Choice(
                noun=ParsedMathObject(
                    mathObject=OfSize(
                        obj=ParsedMathObject(
                            mathObject=ArbitraryMathObject(
                                noun=subset
                            ),
                             context=ParseContext(
                                from=3,
                                 to=4
                            )
                        ),
                         size=ParsedQuantity(
                            quantity=Variable(
                                name=k
                            ),
                             context=ParseContext(
                                from=6,
                                 to=7
                            )
                        )
```

```
                ),
                 context=ParseContext(
                    from=2,
                     to=7
                )
            )
        )
    ),
     q2=Amount(
        obj=Choice(
            noun=ParsedMathObject(
                mathObject=Multiple(
                    amount=ParsedQuantity(
                        quantity=Difference(
                            left=Variable(
                                name=n
                            ),
                             right=Variable(
                                name=k
                            )
                        ),
                         context=ParseContext(
                            from=12,
                             to=13
                        )
                    ),
                     obj=ParsedMathObject(
                        mathObject=ArbitraryMathObject(
                            noun=subset
                        ),
                         context=ParseContext(
                            from=18,
                             to=19
                        )
                    )
                ),
                 context=ParseContext(
                    from=11,
                     to=19
                )
            )
        )
    )
),
 context=ParseContext(
    from=0,
     to=19
)
```

)

As to how well I'm progressing toward my winter goal, I'm definitely making progress, but in terms of actually reaching my goal, considering that I now have 3 more things to do next week and that I completely forgot that Thanksgiving break was a thing the last time I wrote a journal, I think the answer is definitely not as well as I thought I was, and I might need to tone down my goal, though I didn't do that in this journal as I'm not completely sure about if/how I should modify it yet.