WallA2 354 Critical Appraisal

Aidan Wall

December 2021

1 Progress And Roadblocks

I had issues with the .cabal file as it was giving me many errors. I had to meet with Dan separately and set up the Docker container he created in order to solve this error. Until that I struggles with being able to test my results. I struggled a little with this assignment as it was difficult to implement some of the functions because my Haskell skills are still not perfect. I also struggled to submit my assignment as I got lost along the way because I was using different versions and struggled with version control as a whole. I was working on a different version on my local computer than was uploaded to github and had many stupid capitalization errors.

2 #:# does not parse, but (#):# does. Why? Does that mean that it would make sense to change the grammar?

In LambdaNat5 we look at the usage of the character/indicator #. #:# does not parse. This is because when trying to parse this expression, it will look at the grammar we have defined for it, and in the grammar we defined the order for things to be evaluated with parenthesis. This expression does not have any parenthesis. When parenthesis are added to the function and it looks like (#):# the interpreter now understands that this is Exp16 as seen in the grammar file (.cf file).The machine is being told to do something with the parenthesis, and what is inside of it.

3 In the critical appraisal explain the results for sum x:2:3:4:# and sum 1:2:3:x:#.

For the first case, sum x:2:3:4:# when x is at the beginning of the list, the function outputs x+9. It calculates the sum of everything after x, which is 2+3+4 which is 9. When x is at the end of the function, it returns 1 +(2+(3+(x+0))) because of recursive addition in the function.

4 In the critical appraisal explain what happens if one replaces the 2 in plustwo by a free variable y.

The function plustwo takes a variable and returns x+2. If the 2 in plustwo was replaced by a free variable y, the function would then be x+y instead of x+2. This would then make the function pointless because it would be adding another variable and not 2. You would also have to make the function reference another variable y so that it knows what to add.

5 Reflect on the differences between LambdaNat5 and Haskell. In your experience from this assignment, how does writing code in LambdaNat5 and Haskell compare? How far did we come in implementing a functional programming language?

Writing code in LambdaNat5 and Haskell was different because we were writing LambdaNat5 in Lambda Calculus instead of Haskell. I thought it was easier to write the code in Haskell than in Lambda Calculus because we have done more coding in it throughout the semester. Since the theory behind lambda calculus is what Haskell is based on it was not too difficult to make the jump, but still challenging syntactically. Both Haskell and Lambda Calculus are based off of creating/defining functions and applying arguments to them, which is pretty much what we have done in this assignment. We applied the fundamentals of Lambda calculus to our own syntax to create/implement a functional programming language.