Final Report

CS525

Jiarui Fu

12/13/2021

The intuition of my final project is to first utilize my midterm solutions. Translate the t1erm into t2erm, use the tinfer function to type check the term and then interpret it to obtain the environment. Lastly, using this newly generated t2erm, produce boxed c-representation code.

One of the first things I tried is to produce a file that has access to all three portions of the midterm, unfortunately I was only able to compile the file with trans12 and interp.

In the compiler\_sol.dats file, the solution of the compiler starts roughly on line 1211. Where tt\_instr\_wrap & ftt\_instr\_wrap are wrapper functions of the instruction\_const function which constructs the instructions. The wrapper functions translate the t1erm to t2erm (and in an ideal world type infers it to make sure that everything type checks). The instruction\_const and finstruction\_const, which only differ by file output location, assumes that type checking has already been done by the wrapper function.

The finstruction\_const, pattern matches the t2erm and “emits” boxed representation c-code accordingly. Out of the lambda2 terms, the base terms all functions correctly, with the caveat that for T2Mstr, I wasn’t able to let the print function print quotation marks as part of the string. This is the part where I mentioned on my readme file about the lambda testcase where the single quotation mark is manually fixed to double quotation

I also modified interp into interp0\_env2, where it instead of just the t2val it also returns its last iteration of the environment, in attempts to use this environment to decipher T2Mvar variables in the construction const. However this attempt is not 100% successful and only parts of it are implemented. Also lambda2 app also handles T2Mvar as a separate case for reasons already mentioned in the comments of the file. The point of having the environment is also to implement lambda\_lifting. Unfortunately I understand the concept, basically providing the environment for inner functions in lambda, which needs to be flattened for C, but I could not implement it

In addition I could not get the section of code for lambda app to compile, so it has to be commented out in the final version. The idea is that if the for T2Mapp(func, arg) where the func is just a lam or fix, the code would print out the full definition of the function first and then make a function call. However if func is a T2Mvar, which most likely happens when the function is already declared, for example in a fix term, printing out the definition of the function would not be ideal. Therefore it would only emit code that makes the function call, for example in terms of a recursion call. T2Mfix also utilizes the first t2var term to name the function in the emitted code, however due to the lack of implementation of T2Mapp recursion could not be demonstrated.

The runtime.h file is based on the file with the same name from assignment 04. For the purpose of this project several more structs are added. Unfortunately I was unable to produce a working boxed representation of the ref\_new, ref\_get and ref\_set. Therefore all three of these t2erms are unimplemented on instruction\_const.