1 Reading

Please read Chapter 5 in your textbook.

2 Church Numerals

Using the “fulluntyped” check, write functions with the following behavior:

1. plus2 Take a church numeral and return the church numeral two more.
2. double Take a church numeral and return the church numeral twice as big.
3. square Take a church numeral and return the church numeral representing its square.
4. power2 Take a church numeral and return the church numeral representing two to this power.

Do not simply use the existing functions in your answers (though you may use your earlier definitions); In each case ensure that your definition doesn’t include the church numeral for 2 in it.

Put your definitions in homework4.f in your repository (in the f directory) along with the definition of the church numerals from 0 to 9, the definition of realnat, and several test cases for each function showing that it works.

You will either need to do this part on andrew.cs.uwm.edu (using special ssh port 53211) or on your own machine. The fulluntyped checker is in

/afs/cs.uwm.edu/users/classes/cs732/checkers/fulluntyped/f

3 Proofs

Prove the following theorems in SASyLF:

1. Prove that multi-step full (unrestricted) beta-reduction of \( \omega \) where

\[
\omega = (\lambda x \cdot x.x)(\lambda x \cdot x.x)
\]

never reaches a “value,” (a lambda abstraction). To do this, you need to specify multi-step full beta-reduction. If you use the same technique (reflexive, transitive, inclusion) for multi-step evaluation as in Homework #3, you will require another lemma. It is easier if you define multi-step evaluation to be right recursive.

2. Prove that call-by-value evaluation is deterministic.
4 Graduate Students

For each of the following situations, give a valid\(^1\) pure lambda-calculus term that has the given properties:

1. diverges under normal evaluation and under call-by-value evaluation, but not under call-by-name
2. diverges under normal evaluation but not under call-by-value
3. diverges under call-by-value but not under normal evaluation

You may use \(\omega\) defined above. Please explain your answers!

Call-by-need is used in lazy languages such as Haskell. Explain how it differs from the three: call-by-value, call-by-name and normal evaluation. Explain which of the three diverges on exactly the same terms as call-by-need, while still not being the same. Explain on paper.

5 Files

As usual, there is a repo for this homework. Please turn in your SASyLF proof by pushing committed changes to the repo.

---

\(^1\)no unbound variables please