Assignment 2
The Simply Typed λ-Calculus

15-814: Types and Programming Languages
David Kahn & Frank Pfenning

Due Thursday, September 16, 2021
65 pts

This assignment is due on the above date and it must be submitted electronically on Gradescope. Please use the attached template to typeset your assignment and make sure to include your full name and Andrew ID. For the written problems, you may also submit handwritten answers that have been scanned and are **easily legible**.

Please carefully read the policies on collaboration and credit on the course web pages at `http://www.cs.cmu.edu/~fp/courses/15814-f20/assignments.html`.

You should hand in two files

- `hw02.pdf` with your written solutions to the questions.
- `hw02.lam` with the code, where the solutions to the problems are clearly marked and auxiliary code (either from lecture or your own) is included so it passes the LAMBDA checker.

1 Y Combinator

Task 1 (L3.1, 15 pts) The Lucas function (a variant on the Fibonacci function) is defined mathematically by

$$\begin{array}{lcl} \mathsf{lucas}\ 0 & = & 2 \\ \mathsf{lucas}\ 1 & = & 1 \\ \mathsf{lucas}\ (n+2) & = & \mathsf{lucas}\ n + \mathsf{lucas}\ (n+1) \end{array}$$

Give an implementation of lucas using the Y combinator.

You may copy the functions from nat.lam to the beginning of your file `hw02.lam` to use as helper functions. Test your implementation on inputs 0, 1, 9, and 11, expecting results 2, 1, 76, and 199.

In the previous homework, you recorded the number of β-reductions taken by your primitive recursive implementation of lucas. Compare this record to your Y combinator implementation. Which of the two implementations is more "efficient" (in the sense of number of β-reductions)?

2 Simple types

Task 2 (L3.2, 10 points) Fill in the blanks in the following typing judgments so the resulting judgment holds, or indicate there is no way to do so. You do not need to justify your answer or supply a typing derivation, and the types do not need to be "most general" in any sense. Remember that the function type constructor associates to the right, so that $\tau \to \sigma \to \rho = \tau \to (\sigma \to \rho)$.

(i) $\boxed{} \vdash y\, x : \alpha$

(ii) $\boxed{} \vdash x\, x : \boxed{}$

(iii) $\cdot \vdash \boxed{} : (\alpha \to \alpha) \to \alpha$

(iv) $\cdot \vdash (\lambda z.\, z)\, (\lambda x.\, \lambda y.\, \lambda p.\, p\, x\, y) : \boxed{}$

(v) $\cdot \vdash \lambda f.\, \lambda g.\, \lambda x.\, (f\, x)\, (g\, x) : (\alpha \to \boxed{}) \to (\alpha \to \boxed{}) \to (\alpha \to \boxed{})$

3 Proof by Rule Induction

Since this is the first time we (that is, you) are proving theorems about judgments defined by rules, we ask you to be very explicit, as we were in the lectures and lecture notes. In particular:

- Explicitly state the overall structure of your proof: whether it proceeds by rule induction, and, if so, on the derivation of which judgment, or by structural induction, or by inversion, or just directly. If you need to split out a lemma for your proof, state it clearly and prove it separately. If you need to generalize your induction hypothesis, clearly state the generalized form.
- Explicitly list all cases in an induction proof. If a case is impossible, prove that is is impossible. Often, that's just inversion, but sometimes it is more subtle.
- Explicitly note any appeals to the induction hypothesis.
- Any appeals to inversion should be noted as such, as well as the rules that could have inferred the judgment we already know. This could lead to zero cases (a contradiction—the judgment could not have been derived), one case (there is exactly one rule whose conclusion matches our knowledge), or multiple cases, in which case your proof now splits into multiple cases.
- We recommend that you follow the line-by-line style of presentation where each line is justified by a short phrase. This will help you to check your proof and us to read and verify it.

Task 3 (L4.1, 20 points) In lecture, we defined the reflexive and transitive closure ($\longrightarrow^*$) of the single-step reduction ($\longrightarrow$) with the following:

$$\frac{}{e \longrightarrow^* e}\ \mathsf{red}^*/\mathsf{refl} \qquad \frac{e_1 \longrightarrow^* e_2 \quad e_2 \longrightarrow^* e_3}{e_1 \longrightarrow^* e_3}\ \mathsf{red}^*/\mathsf{trans} \qquad \frac{e_1 \longrightarrow e_2}{e_1 \longrightarrow^* e_2}\ \mathsf{red}^*/\mathsf{step}$$

However, it is more common to define multistep reduction with only two rules, as is done for the $\Longrightarrow$ judgment below:

$$\frac{}{e \Longrightarrow e}\ \mathsf{reds/refl} \qquad\qquad \frac{e_1 \longrightarrow e_2 \quad e_2 \Longrightarrow e_3}{e_1 \Longrightarrow e_3}\ \mathsf{reds/step}$$

Prove by rule induction that these two definitions are equivalent in the sense that $e \Longrightarrow e'$ iff $e \longrightarrow^* e'$.

Task 4 (L4.2, 20 points) Recall the relation $\longrightarrow^*$ defined in the previous task. Prove by rule induction that if $\Gamma \vdash e : \tau$ and $e \longrightarrow^* e'$ then $\Gamma \vdash e' : \tau$. Here (as in general in the course), you may use theorems we have proved in the course (lecture or notes).