Written Assignment 5
Due November 13th 2017 at 12:30pm

You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work and you must acknowledge the students you work with. Written assignments must be turned in at the start of lecture on the indicated due date. If you choose to use any late days on this assignment, you must email your assignment to the TA.

1. (20 points) Consider the following language:

$$S \quad \rightarrow \quad \textit{integer constant} \mid S_1 * S_2 \mid S_1 / S_2 \mid \textit{let } x = S_1 \textit{ in } S_2$$

   Assume that the only run-time error occurs from division by zero (i.e., S_2 is zero in a division)

   (a) (6 points) List three programs in this language

   (b) (14 points) Give the natural large-step operational semantics for this language that "get stuck" if a program divides zero.

   (c) (15 points) Using the types `zero` and `non-zero`, give sound typing rules for this language that prevent all run-time errors without being overly restrictive.

   (d) (10 points) Proof preservation of your type system with respect to your operational semantics

   (e) (10 points) Proof progress of your type system with respect to your operational semantics

2. (35 points) Consider the language given by the following grammar:

$$S \quad \rightarrow \quad \lambda id.S \mid (S_1 S_2) \mid id \mid c$$

   where id is any identifier and c is an integer constant. Here, $\lambda id.S$ defines an anonymous function and $(S_1 S_2)$ is a function application.

   (a) (6 points) Assuming *call-by-name* semantics, write three distinct programs using (in total) *all* syntactic constructs and give their meaning.

   (b) (12 points) Give large-step operational semantics using call-by-name semantics. Only use an environment if necessary.

   (c) (6 points) Give a full derivation of its meaning of all three programs you wrote in part (a)

   (d) (7 points) Give new operation semantics that use call-by-value semantics.

   (e) (4 points) List one advantage of call-by-value and one advantage of call-by-name semantics and explain your reasoning.

3. (35 points) Consider again the language from Question 1 with *call-by-name* semantics:

$$S \quad \rightarrow \quad \lambda id.S \mid (S_1 S_2) \mid id \mid c$$

   (a) (4 points) List all run-time errors that are possible in this language and give an example program for each possible error.

(b) (6 points) To prevent all run-time errors, you decide to design a sound type system. For this, you modify the language as follows to allow for type annotations:

$$S \quad \rightarrow \quad \lambda id : \tau.S \mid (S_1 S_2) \mid id \mid c$$

Give your language of types and their concretization

(c) (17 points) Give *sound* typing rules that prevent all run-time errors, but still allow the following program to type check $((\lambda x.x)7)$. *Hint:* You may find a type environment helpful.

(d) (5 points) Give a type derivation of the following program:

$$(\lambda x.x\ 7)$$

(e) (3 points) Explain if your type system is polymorphic or not.

4. (20 points) Consider the following system of type constraints generated from type inference in the lambda language from lecture:

$$\begin{array}{lcl} \alpha_1 & = & Int \rightarrow \alpha_2 \\ Int \rightarrow \alpha_2 & = & \alpha_3 \\ \alpha_3 & = & \alpha_4 \rightarrow \alpha_4 \end{array}$$

(a) (14 points) Draw the final type DAG after calling unify() on each constraint. For each equivalence class, indicate the representative of this equivalence class by double-circling it. For this, you probably have to draw all intermediate steps, but you only have to show the final graph.

(b) (6 points) Give the type solution for $\alpha_1, \alpha_2, \alpha_3$ and α_4 using your union-find DAG.