Homework 3 of CS520 Theory of Programming Languages
Deadline: 6:00pm on 27 November (Monday)

Submit your solutions in KLMS. (Reminder: We adopt a very strict policy for handling dishonest behaviours. If a student is found to copy answers from fellow students or other sources in his or her homework submission, she or he will get F.)

The numbers in the questions refer to exercise questions in the textbook of the course, i.e. "Theories of Programming Languages" by John C. Reynolds.

Question 1

Solve the part (a) of 5.4 after making the following four modifications. First, the part (a) mentions the type of command meanings $[\![c]\!]$ that we did not cover in the lectures. Instead of this type of $[\![-]\!]$, use the following one, which extends the semantics that we studied in the lectures:

$$[\![-]\!] \in \big[\langle\text{comm}\rangle \to \Sigma \to \Omega\big].$$

Here

$$\hat{\Sigma} \overset{\text{def}}{=} \Sigma \cup (\langle\text{label}\rangle \times \Sigma) \qquad \text{and} \qquad \Omega \simeq \big(\hat{\Sigma} + (\mathbb{Z} \times \Omega) + (\mathbb{Z} \to \Omega)\big)_\perp, \tag{1}$$

and the isomorphism $\simeq$ refers to the existence of two continuous functions

$$\phi : \Omega \to \big(\hat{\Sigma} + (\mathbb{Z} \times \Omega) + (\mathbb{Z} \to \Omega)\big)_\perp \qquad \text{and} \qquad \psi : \big(\hat{\Sigma} + (\mathbb{Z} \times \Omega) + (\mathbb{Z} \to \Omega)\big)_\perp \to \Omega$$

such that

$$\psi \circ \phi = \text{id} \quad \text{and} \quad \phi \circ \psi = \text{id}.$$

Second, extend the semantics in my hand-written notes (the same as the one in Section 5.6), instead of the direct semantics in Section 5.1. Third, you can ignore the variable declaration when extending the semantics. This means that you need to define the semantics of the following language:

$$\begin{aligned}
\langle\text{comm}\rangle ::= {} & \langle\text{var}\rangle := \langle\text{intexp}\rangle \mid ?\langle\text{var}\rangle \mid !\langle\text{intexp}\rangle \mid \langle\text{comm}\rangle ; \langle\text{comm}\rangle \\
\mid {} & \textbf{if } \langle\text{boolexp}\rangle \textbf{ then } \langle\text{comm}\rangle \textbf{ else } \langle\text{comm}\rangle \\
\mid {} & \textbf{while } \langle\text{boolexp}\rangle \textbf{ do } \langle\text{comm}\rangle \\
\mid {} & \textbf{fail } \langle\text{label}\rangle \mid \textbf{catch } \langle\text{label}\rangle \textbf{ in } \langle\text{comm}\rangle \textbf{ with } \langle\text{comm}\rangle
\end{aligned}$$

Finally, while defining the semantics, especially the one for sequential composition, you may use the extension of a function $f : \Sigma \to \Omega$ to $f_* : \Omega \to \Omega$, which we discussed and is also defined in the textbook. (20 marks)

Question 2

Solve 5.10. (20 marks)

Question 3

You need to do two things for this question. First, solve the parts (a), (b) and (c) of 10.1. Among the three expressions in 10.1, ignore the third and solve these parts with the first two expressions. Second, for these two expressions, write down the canonical forms that you would get if those expressions are evaluated according to the eager evaluation.

[Hint 1] Using the following abbreviations made it much easier for me to do the required calculations.

$$N \overset{\text{def}}{=} \lambda b.\, \lambda x.\, \lambda y.\, b\ y\ x, \qquad T \overset{\text{def}}{=} \lambda z.\, \lambda w.\, z,$$

$$D \overset{\text{def}}{=} \lambda d.\, d\ d, \qquad F \overset{\text{def}}{=} \lambda f.\, \lambda x.\, f\ (f\ x).$$

[Hint 2] N is the standard encoding of the negation of booleans in the lambda calculus, and T that of true. The expression F takes a function f and composes it with itself. Thus, the first expression composes the negation operation with itself, and then applies it to true. The second expression applies F to F. Intuitively, this should lead to the composition of the second F with itself because of the first F. (40 marks)

Question 4

Solve 10.12. You may assume the substitution lemma stated in 10.11. Using this assumption, you have to prove that

$$[\![(\lambda v.\, e)\ z]\!]\,\eta = [\![e/(v \to z)]\!]\,\eta$$

for all variables v, expressions e, canonical expressions or variables z, and environments η. Recall that an environment η is an element in $V^{\langle \text{var} \rangle}$ (i.e. function from the set of variables $\langle \text{var} \rangle$ to V), and that every canonical form is a lambda expression $\lambda w.\, e'$. As a result, if z is a canonical expression or a variable,

$$[\![z]\!]\,\eta \neq \bot.$$

(20 marks)