CS611 Homework 2 DUE: 09/25/00

What to turn in

Turn in the assignment during class on the due date.

1. Equivalence of semantics (15 pts.)

   Winskel, exercise 4.10. Do only the "only if" direction: prove that any complete execution in the small-step semantics can be computed in the large-step semantics. Show only the cases of the while statement and the sequence of two commands, c_1; c_2. **Note**: only the only-if direction of the sequence lemma needs to be proved.

2. Lambda calculus (30 pts.)

   (a) Free variables
   Identify the free and bound variables in each of the following expressions:

   i. (λx (x y))
   ii. (λ(z x) (x y z))
   iii. (z (λy ((λz (x y)) z)))

   (b) Encodings

   i. Show a sequence of β- and/or η-reductions applied to (*INC* 1), resulting in the standard representation of 2 as a Church numeral.
   ii. Show how to write a lambda term named *EMPTY?* that determines whether a list, (using the implementation given in class), is empty, and returns either *TRUE* or *FALSE*, accordingly. You will need to choose a lambda term to represent the empty list, *NIL*.
   iii. Show how to write *MULT*, *EXPT*, and *DEC* operations that work on Church numerals. Hint for *DEC*: first construct a function that maps a pair $\langle l, r\rangle$ to a pair $\langle l+1, l\rangle$.

   (c) The S, K, I Combinators
   Consider the following definitions:

$$\begin{aligned} S &\equiv (\lambda(\text{x y z})\ ((\text{x z})\ (\text{y z}))) \\ K &\equiv (\lambda(\text{x y})\ \text{x}) \\ I &\equiv (\lambda\text{x x}) \end{aligned}$$

   These are the S, K, and I *combinators*, which Curry experimented with to eliminate the need for variables. A combinator is just another name for a closed term. These three have the remarkable property that any lambda calculus expression containing no free identifiers (a *closed expression*) can be expressed using only applications operating on these three combinators, with no explicit variables or abstraction terms. This property also means that the lambda calculus can be universal with only three distinct identifier names, since no combinator uses more than three identifiers.

   i. Reduce the following expressions to normal form:
      - (K I)
      - (S K)
   ii. Show that the I combinator is superfluous: the S and K combinators can be used to construct an expression with the same normal form.
   iii. Now, we will construct a translation from lambda expressions to expressions containing only applications of the S and K combinators. This translation will be defined in terms of two functions: $C[\![e]\!]$, which converts an expression e into this form, and a function $A[\![x, e]\!]$, which *abstracts* the variable x from the expression e. removing all uses of x within e.

The idea is that $A[\![x, e]\!] = (\lambda\ x\ e)$, in the sense that the two expressions have the same effect when applied to any argument (they are extensionally equal). Using the function A, the function C can be defined simply.

$$\begin{aligned} C[\![x]\!] &\equiv x \\ C[\![e_0\ e_1]\!] &\equiv (C[\![e_0]\!]\ C[\![e_1]\!]) \\ C[\![\lambda\ x\ e]\!] &\equiv A[\![x, C[\![e]\!]]\!] \end{aligned}$$

Because A is only applied to expressions produced by C, it needs to be defined only for expressions that are identifiers and applications. For example, consider $A[\![x, x']\!]$ where $x' \neq x$. We require $(A[\![x, x']\!]e) = (\lambda\ x\ x')e$ for any e, so we obtain the right effect with the following definition:

$$A[\![x, x']\!] \equiv (K\ x') \qquad (x \neq x')$$

Define the remainder of the translation to the S, K, and I combinators. Does this translation result in the most compact equivalent expression using the combinators?

(d) Alpha-renaming
An equivalence relation has the properties of reflexivity ($a \sqsubseteq a$), symmetry ($a \sqsubseteq b \iff b \sqsubseteq a$) and transitivity ($a \sqsubseteq b \wedge b \sqsubseteq c \Rightarrow a \sqsubseteq c$). Show that alpha-equivalence is an equivalence relation on lambda calculus expressions.

3. Fixed Points (10 pts.)

Consider the definition

$$F = \lambda f \in Z_\perp \to Z_\perp\ .\ \lambda x \in Z_\perp.\text{if } (x = 1) \text{ then } 1 \text{ else } f(x-1) + 2x - 1$$

We can define the squaring function as

$$SQUARE \overset{def}{=} \mathit{fix}(F) = \bigsqcup_{n \in \omega} F^n(\perp)$$

(a) Expand $F^2(\perp)$ and $F^3(\perp)$ into normal form for some reasonable definition of normal form.

(b) For what argument values x do these two functions compute x^2 ?

(c) Prove inductively the set of argument values x for which the function $F^n(\perp)$ computes x^2 correctly.