Cornell University
Computer Science 6110

Homework 4 DUE: Monday, April 14, 11:59PM

Updates

- 4/9: clarified definition of CPO isomorphism.

What to turn in

Turn in the assignment by midnight on CMSX on the due date, including both code and written problems.

Working with a partner

You may have a partner on this assignment. Both partners are expected to work on all parts of the assignment, and to understand the solution.

1. **Domain isomorphism** [35 pts]

   A set S is *isomorphic* to another set S' if there is a one-to-one and onto function (a bijection) mapping S to S'. Similarly, a domain (for our purposes, a CPO) D is isomorphic to a domain D' if there is a *continuous* bijection mapping D to D'. That is, there is both a continuous function mapping D and D' and a continuous inverse function mapping D' back to D. The two domains must not only have corresponding elements but also corresponding structure. Because the function is continuous, it preserves not only ordering but also least upper bounds (suprema). So it is harder to show that domains are isomorphic than it is for sets. But it also shows a deeper correspondence: in particular, that the mapping between the domains is computable.

   For each of the following pairs of domains, show that they are isomorphic or that they are not.

   (a) The domains $D \times \mathbb{U}$ and D for any CPO D (where $\mathbb{U}$ is the unit domain ($\{\mathit{unit}\}$, =)).

   (b) The domains $D \to E$ and $\{f \in D_\perp \to E_\perp \mid f(\perp) = \perp\}$ (both ordered pointwise), for any CPO's D and E.

   (c) The domains $D \times E \to F$ and $D \to E \to F$ for any CPO's D, E, F.

2. **CPOs** [30 pts]

   In this problem you will prove some facts about the mathematical structure P_ω, which can serve as a model for the untyped lambda calculus. P_ω is a pair $(2^{\mathbb{N}}, \subseteq)$, where $2^{\mathbb{N}}$ is the set of all subsets of $\mathbb{N}$.

   (a) Prove that P_ω is a CPO (in fact, it is a complete lattice). What are the least upper and greatest lower bound operations in P_ω?

   (b) Prove that the continuous functions from P_ω to P_ω are exactly the ones characterized by their values on finite subsets of $\mathbb{N}$, that is functions f such that for all $A \subseteq \mathbb{N}$, $f(A) = \bigcup\{f(B) \mid B \subseteq A, B \text{ is finite}\}$

   (c) **Bonus question (5 pts):** Define a 1–1 continuous function from $[P_\omega \to P_\omega]$ to P_ω. You may find the following functions useful:

   - The bijective pairing function $p : \mathbb{N}^2 \to \mathbb{N}$.
   - The bijective finite set coding function $s : \mathit{Fin}(\mathbb{N}) \to \mathbb{N}$, where $\mathit{Fin}(\mathbb{N})$ denotes the set of all finite sets of natural numbers.

   You don't need to know what these functions are, but for the curious, here are possible definitions:

   - $p(x, y) = \frac{(n+m)(n+m+1)}{2} + m$.
   - $s(\{k_0, \ldots, k_l\}) = \sum 2^{k_i}$.

3. **Standard semantics** [35 pts]

We saw earlier that we could use CPS translation to compactly encode non-local control features. In this problem we'll use this approach to model a **break** statement like that in Java. Consider the IMP language, which already has **while**, and for which we've already seen a direct semantics:

$$
\begin{array}{rcl}
x & \in & \textbf{Var} \\
a & \in & \textbf{Aexp} ::= n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \\
b & \in & \textbf{Bexp} ::= \textbf{true} \mid \textbf{false} \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid a_1 = a_2 \mid a_1 \leq a_2 \\
c & \in & \textbf{Com} ::= \textbf{skip} \mid x := a \mid c_1;\ c_2 \mid \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \mid \textbf{while}\ b\ \textbf{do}\ c
\end{array}
$$

Here we'll develop a continuation semantics which will look very much like the CPS translations we've already been doing, except that the functions we will write are mathematical functions on well-defined domains. The first step is to define the domains that will be the interpretations of various terms. It makes sense to define a separate kind of continuation for each of the three kinds of terms (**Aexp**, **Bexp**, and **Com**. Recall that a continuation semantics involves continuations, which are functions that don't return a value. We introduce a domain *Answer* to represent what continuations return. Since continuations don't return, *Answer* could be equally well be almost anything. By leaving it unspecified, we ensure that we don't use it!

$$
\begin{aligned}
\mathit{ACont} &= \mathbb{Z} \to \mathit{Answer} \\
\mathit{BCont} &= \mathbb{T} \to \mathit{Answer} \\
\mathit{CCont} &= \Sigma \to \mathit{Answer}
\end{aligned}
$$

The three semantic functions then send their results to the appropriate kind of continuation:

$$
\begin{aligned}
\mathcal{A}[\![a]\!] &\in \mathit{ACont} \to \Sigma \to \mathit{Answer} \\
\mathcal{B}[\![b]\!] &\in \mathit{BCont} \to \Sigma \to \mathit{Answer} \\
\mathcal{C}[\![c]\!] &\in \mathit{CCont} \to \Sigma \to \mathit{Answer}
\end{aligned}
$$

Now we can define the meaning of the various terms. For example, the meaning of a variable expression x and a command **skip** is as follows:

$$
\begin{aligned}
\mathcal{A}[\![x]\!] &= \lambda k \in \mathit{ACont}\,.\,\lambda\sigma \in \Sigma\,.\,k(\sigma x) \\
\mathcal{C}[\![\textbf{skip}]\!] &= \lambda k \in \mathit{CCont}\,.\,\lambda\sigma \in \Sigma\,.\,k\sigma
\end{aligned}
$$

We can even use η-reduction to simplify a bit:

$$
\mathcal{C}[\![\textbf{skip}]\!] = \lambda k \in \mathit{CCont}\,.\,k
$$

(a) Finish writing the rest of the continuation semantics for IMP. The **while** command will need to use *fix* just like in the direct semantics. Annotate all λ's with explicit domains, as above. You do not have to use exactly the domains given above, but if you change them you must justify it.

(b) Now suppose that we add a **break** statement to IMP. Informally, **break** causes the closest enclosing **while** statement to immediately terminate. Show how to modify your domain definitions and your semantics to support this new statement.