Denotational Semantics and Domain Theory

I am working through these Cambridge notes. The problem numbers don't line up. Sue me.

1 Basics of Denotational Semantics (from Cambridge notes)

Problem 1.1. Consider the function

$$f_{b,c} : (State \rightharpoonup State) \to (State \rightharpoonup State)$$

defined on Slide 11.

a) Show by induction on n that

$$f^n_{b,c}(\bot) = \lambda s \in State. \begin{cases} c^k(s) & \text{if } 0 \le k < n \text{ is such that } b(c^i(s)) = true \\ & \text{for all } 0 \le i < k \text{ and } b(c^k(s)) = false \\ \text{undefined} & \text{if } b(c^i(s)) = true \text{ for all } i \ge 0 \end{cases}$$

b) Let $w_{b,c} : State \rightharpoonup State$ be the partial function defined as

$$w_{b,c} \stackrel{\text{def}}{=} \lambda s \in State. \begin{cases} c^k(s) & \text{if } k \ge 0 \text{ is such that } b(c^i(s)) = true \\ & \text{for all } 0 \le i < k \text{ and } b(c^k(s)) = false \\ \text{undefined} & \text{if } b(c^i(s)) = true \text{ for all } i \ge 0 \end{cases}$$

Show that $w_{b,c}$ satisfies the fixed-point equation

$$w_{b,c} = f_{b,c}(w_{b,c}).$$

c) Describe the function $f_{b,c}$ for $b = [\![\textbf{true}]\!] = \lambda s \in State.true$ and $c = [\![\textbf{skip}]\!] = \lambda s \in State.s$. Which partial functions from states to states are fixed points of this $f_{b,c}$? What is its least fixed point (with respect to the $\sqsubseteq$ ordering defined above)? Does this least fixed point agree with the partial function from states to states determined by the operational semantics of **while true do skip**?

Solution. .

a) In the base case, let $n = 1$ and we have $f_{b,c}(\bot) = \lambda s \in State.\ if(b(s), \bot, s)$. This gives us the definition

$$f_{b,c}(\bot) = \lambda s \in State. \begin{cases} \text{undefined} & \text{if } b(c^0(s)) = true \\ c^0(s) & \text{if } 0 \leq k < 1 \text{ is such that } b(c^i(s)) = true \\ & \text{for all } 0 \leq i < k \text{ and } b(c^0(s)) = false \end{cases}$$

where $g^0(s) = s$ for any function g. For the inductive case, suppose that we have the function $f^n_{b,c}(\bot)$ as defined in the problem statement and we wish to compute $f_{b,c}(f^n_{b,c}(\bot))$. By the definition, we get the function

$$f_{b,c}(f^n_{b,c}(\bot)) = \lambda s \in State. \begin{cases} f^n_{b,c}(\bot)(c(s)) & \text{if } b(s) = true \\ s & \text{if } b(s) = false. \end{cases}$$

Or, unfolding the definition of $f^n_{b,c}(\bot)$,

$$f_{b,c}(f^n_{b,c}(\bot)) = \lambda s \in State. \begin{cases} c^k(c(s)) & \text{if } b(s) = true \text{ and } 0 \leq k < n \\ & \text{is such that } b(c^i(c(s))) = true \\ & \text{for all } 0 \leq i < k \text{ and} \\ & b(c^k(c(s))) = false \\ \text{undefined} & \text{if } b(s) = true \text{ and} \\ & b(c^i(c(s))) = true \text{ for all } i \geq 0 \\ c^0(s) & \text{if } b(s) = false \end{cases}$$

which can be massaged into the definition in the problem statement by folding the additional calls to c into the ranges defined in the conditions and then combining the top and bottom cases.

b) If we expand out the function $f_{b,c}(w_{b,c})$, we get the definition

$$f_{b,c}(w_{b,c}) = \lambda s \in State. \begin{cases} s & \text{if } b(c(s)) = false \\ c^k(c(s)) & \text{if } b(s) = true \text{ and if } k \geq 0 \\ & \text{is such that } b(c^i(c(s)) = true \text{ for} \\ & \text{all } 0 \leq i < k \text{ and } b(c^k(s)) = false \\ \text{undefined} & \text{if } b(s) = true \text{ and} \\ & \text{if } b(c^i(c(s)) = true \text{ for all } i \geq 0 \end{cases}$$

which can be massaged into the definition of $w_{b,c}$ similarly to in part a). Note that this is $\lim_{n\to\infty} f^n_{b,c}(\bot)$.

c) If we plug b and c into the definition of $w_{b,c}$ above, we see that the top branch never triggers because $b(c^k(s)) = true$ will never equal $false$. Thus, the least fixed point is $\perp$. Because $\perp$ has the empty set as its graph and thus approximates any function, we would expect that any function satisfies the fixpoint equation. Indeed, per Slide 11,

$$f_{[\![\mathbf{true}]\!],[\![\mathbf{skip}]\!]} = if(true, w(s), s),$$

so a solution to $w = f_{[\![\mathbf{true}]\!],[\![\mathbf{skip}]\!]}(w)$ is any function such that $w = w$.

□

Problem 1.2. Show that the relation $\sqsubseteq$ is a partial order with least element $\perp$.

Solution. This follows trivially from the fact that $\subseteq$ is a partial order with least element $\emptyset$, and that two functions are equal if their graphs are equal. □

Problem 1.3. Prove that

$$w = f_{b,c}(w) \Longrightarrow w_{b,c} \sqsubseteq w$$

for all $w \in (State \rightharpoonup State)$.

Solution. We wish to prove that, for any $s, s' \in State$, if $w_{b,c}(s) = s'$ then $w(s) = s'$. From the definition of $w_{b,c}$ in Problem 1.1, we see that if $w_{b,c}(s) = s'$ then $s' = c^k(s)$ for some $k \geq 0$ and $b(c^i(s)) = true$ for all $0 \leq i < k$ and $b(c^k(s)) = false$. We proceed by induction on k. If $k = 0$, then $b(s) = false$ and so $w = if(false, w(c(s)), s) = c^0(s)$. If $k = n + 1$, we see that $w = if(true, w(c(s)), s) = w(c(s))$, and by the inductive hypothesis this must equal $c^n(c(s)) = c^{n+1}(s)$. □

2 Least Fixed Points

Problem 2.1. Verify the claims implicit in Slide 24: that the relation $\sqsubseteq$ defined there is a partial order; that f is indeed the lub of the chain $f_0 \sqsubseteq f_1 \sqsubseteq \ldots$; and that the totally undefind partial function is the least element.

Solution. We proved that $\sqsubseteq$ is a partial order in Problem 1.2. By construction, every element of the chain $f_i \sqsubseteq f$, so f is clearly an upper bound of the chain. Given any other upper bound g, the domain of g must be at least the union of the domains of elements f_i, so $f \sqsubseteq g$. □

Problem 2.2. Prove the claims in Slides 25 and 27.

Solution. Let D be a cpo. First, we prove that for $d \in D, \bigsqcup_n d = d$. By reflexivity, $d \sqsubseteq d$, so d is an upper bound of the chain. For any other upper bound d', by definition we must have $d \sqsubseteq d'$, so d is less than all other upper bounds of the chain. Thus, d is the least upper bound.

Next, we prove that for every chain $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ in D,

$$\bigsqcup_i d_i = \bigsqcup_i d_{i+N}$$

for all $N \in \mathbb{N}$. Let d be $\bigsqcup_i d_i$. Every element of the righthand chain is also an element of the lefthand chain because it just skips the first N elements, so d must be an upper bound of the righthand chain. For any other upper bound d', we must also have that d' is an upper bound of the lefthand chain because $d_{i+N} \sqsubseteq d'$ for all i, and $d_j \sqsubseteq d_N$ for all $j < N$. Thus, by the definition of a least upper bound, $d \sqsubseteq d'$, so d is also a least upper bound of the righthand chain.

Finally, we prove that for every pair of chains $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ and $e_0 \sqsubseteq e_1 \sqsubseteq \cdots \sqsubseteq e_n \sqsubseteq \ldots$ in D, if $d_n \sqsubseteq e_n$ for all $n \in \mathbb{N}$ then $\bigsqcup_n d_n \sqsubseteq \bigsqcup_n e_n$. First, we show that $\bigsqcup_n e_n$ is an upper bound of the chain of ds. By transitivity of the partial order, because $e_i \sqsubseteq \bigsqcup_n e_n$ and $d_i \sqsubseteq e_i$ for all i, we have that $\bigsqcup_n e_n$ is an upper bound of the d chain. Then, by definition of the least upper bound, $\bigsqcup_n d_n \sqsubseteq \bigsqcup_n e_n$. □

Problem 2.3. Verify the claim made in Example 2.3.9 that $f_{b,c}$ is continuous. When is it strict?

Solution. First, we prove that $f_{b,c}$ is monotone. Let w_1 and w_1 be partial functions $State \rightharpoonup State$ such that $w_1 \sqsubseteq w_2$. We wish to show that $\lambda s \in State.if(b(s), w_1(c(s)), s) \sqsubseteq \lambda s \in State.if(b(s), w_2(c(s)), s)$. This is trivial by case analysis on $b(s)$. If $b(s) = true$, then the problem reduces to verifying $w_1(c(s)) \sqsubseteq w_2(c(s))$, which is true by our choice of w_1 and w_2. If $b(s) = false$, then clearly $s \sqsubseteq s$.

Next, to show that $f_{b,c}$ is continuous it suffices to show that for each chain $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ in D,

$$f_{b,c}\left(\bigsqcup_n d_n\right) \sqsubseteq \bigsqcup_n f_{b,c}(d_n).$$

which is true if you think aabout it.

It is strict if $b(s)$ is always true. □

3 Constructions on Domains

Problem 3.1. Verify that the constructions given on Slide 32, in Definition 3.2.3, and on Slides 35 and 31 do give CPOs and domains. Give the proofs of Propositions 3.1.1 and 3.2.2.

Solution. First, we verify the construction of a product CPO. Let D_1 and D_2 be CPOs and their product is the set $D_1 \times D_2$ with partial order

$$(d_1, d_2) \sqsubseteq (d_1', d_2') \overset{\text{def}}{\Leftrightarrow} d_1 \sqsubseteq d_1' \,\&\, d_2 \sqsubseteq d_2'.$$

First, we verify that this is a partial order. It is reflexive because for any $(d_1, d_2) \in D_1 \times D_2$, we have $d_1 \sqsubseteq d_1$ and $d_2 \sqsubseteq d_2$ by the reflexivity of the partial

orders on D_1 and D_2, so $(d_1, d_2) \sqsubseteq (d_1, d_2)$. Next we verify antisymmetry. For any two elements $(d_1, d_2), (d_1', d_2') \in D_1 \times D_2$, if $(d_1, d_2) \sqsubseteq (d_1', d_2')$ and $(d_1', d_2') \sqsubseteq (d_1, d_2)$, then by definition of the partial order, $d_1 \sqsubseteq d_1'$ and $d_1' \sqsubseteq d_1$, so by antisymmetry of the partial order on D_1, we must have $d_1 = d_1'$. Similarly, we have $d_2 = d_2'$, so $(d_1, d_2) = (d_1', d_2')$. Finally, we verify transitivity. If $(a_1, a_2) \sqsubseteq (b_1, b_2)$ and $(b_1, b_2) \sqsubseteq (c_1, c_2)$ then by transitivity of the underlying partial orders, $a_1 \sqsubseteq c_1$ and $a_2 \sqsubseteq c_2$; therefore, $(a_1, a_2) \sqsubseteq (c_1, c_2)$.

Next, we verify that $\bigsqcup_{n \geq 0}(d_{1,n}, d_{2,n}) = \left(\bigsqcup_{i \geq 0} d_{1,i}, \bigsqcup_{j \geq 0} d_{2,j}\right)$. Let d_1 and d_2 by te component-wise suprema. By definition of the partial order, $(d_{1,n}, d_{2,n}) \sqsubseteq (d_1, d_2)$ for all $n \geq 0$ so it is indeed an upper bound. Given any other upper bound (d_1', d_2'), we know by the fact that d_1 and d_2 are suprema that $d_1 \sqsubseteq d_1'$ and $d_2 \sqsubseteq d_2'$, so (d_1, d_2) is the least upper bound. Thus, because D_1 and D_2 are CPOs and have least upper bounds on all their chains, $D_1 \times D_2$ must also have a least upper bound for each of its chains, constructed component-wise as above. Finally, we verify that the bottom element is constructed component-wise which is like the same thing as everything else so.

Next, we verify dependent products. It is again the same proof as above—by induction, take one product at a time and note that it is a domain, so its product with another domain is again a domain.

I'm tired of this problem. □

Problem 3.2. Let X and Y be sets and $X_\perp$ and $Y_\perp$ the corresponding flat domains. Show that a function $f : X_\perp \to Y_\perp$ is continuous if and only if one of (a) or (b) holds:

(a) f is strict

(b) f is constant.

Solution. In the forward direction, first suppose f is strict. Then if we have two elements $x_1, x_2 \in X_\perp$ such that $x_1 \sqsubseteq x_2$, we know that either $x_1 = x_2$ or $x_1 = \perp_X$. If $x_1 = x_2$, then clearly $f(x_1) = f(x_2)$ so $f(x_1) \sqsubseteq f(x_2)$. If $x_1 = \perp_X$, then $f(x_1) = \perp_Y$ because f is strict, so by the definition of a least element, $f(x_1) \sqsubseteq f(x_2)$. Next, we need to prove that f preserves the suprema of chains. Looking at the definition of the partial order for a flat domain, we see that all chains must consist of zero or more instances of $\perp$ followed by zero or more instances of the same element. Because it is strict, applying f to the individual elements of the chain will preserve this structure, thus preserving the supremum.

Otherwise, suppose f is constant and maps everything to some $y \in Y_\perp$. By reflexivity of the partial order, it must be monotone, and similarly it must preserve suprema of chains because it maps everything to y and the supremum of a constaint chain $y \sqsubseteq y \sqsubseteq \ldots$ is y.

In the reverse direction, suppose f is continuous. Then it is monotone, so $f(\perp_X) \sqsubseteq f(x)$ for all $x \in X_\perp$. By the definition of the partial order, this only holds if $f(\perp_X) = \perp_Y$ (f is strict) or if $f(\perp_X) = f(x)$ for all x, which is only possible if f is constant. □

Problem 3.3. Let $\{\top\}$ be a one-element set and $\{\top\}_\perp$ the corresponding flat domain. Let Ω be the domain of 'vertical natural numbers.' Show that the function domain $\Omega \to \{\top\}_\perp$ is in bijection with Ω.

Solution. The underlying set of the function domain $\Omega \to \{\top\}_\perp$ consists of all continuous functions $\Omega \to \{\top\}_\perp$. Because $\{\top\}_\perp = \{\top, \perp\}$ is a two-element set and continuous functions must be monotone, we can think of these as functions that pick an element of Ω and send everything above that element to $\top$ and everything below to $\perp$ (hence the bijection). These functions are clearly monotone, and functions that do not fit that description are clearly not monotone. The only sticking point is that we are not sure whether the selected element should be sent to $\top$ or $\perp$, where either selection seems to leave out one function (either the constant function that maps to $\perp$ or the constant function that maps to $\top$). However, I think there is some subtlety with the natural numbers that I don't yet know, like we can think of the spaces between numbers, of which there are contably many, so the bijection is still there? It's a Hilbert's Hotel situation. □

Problem 3.4. Prove the Proposition on Slide 37.

Solution. The proposition states that for CPOs D, E, and F, the composision function $\circ : ((E \to F) \times (D \to E)) \longrightarrow (D \to F)$ defined by $g \circ f = \lambda d \in D.g(f(d))$ is continuous. We first prove monotonicity. For any $d_1, d_2 \in D$ such that $d_1 \sqsubseteq d_2$, we desire that $(g \circ f)(d_1) \sqsubseteq (g \circ f)(d_2)$ for all $g \in (E \to F)$ and $f \in (D \to E)$. By the continuity of f, we know that $f(d_1) \sqsubseteq f(d_2)$, so it follows by the continuity of g that $g(f(d_1)) \sqsubseteq g(f(d_2))$.

Next we prove that suprema of chains are preserved by the composition function. For all chains $f(d_1) \sqsubseteq f(d_2) \sqsubseteq \cdots \in E$,

$$\bigsqcup_{n\geq 0} g(f(d_n)) = g(\bigsqcup_{n\geq 0} f(d_n)) = g(f(\bigsqcup_{n\geq 0} d_n)).$$

Thus, the composition of two continuous functions is continuous. □

4 Scott Induction

Problem 4.1. Answer all the "Why?"s above in the building of chain-closed subsets.

Solution. Let D be a CPO. First, we verify that the subset $\{(x, y) \in D \times D \mid x \sqsubseteq y\}$ is chain-closed. For any chain $(d_1, d_1') \sqsubseteq (d_2, d_2') \sqsubseteq \ldots$ such that $d_n \sqsubseteq d_n'$ for all n, we can take the component-wise suprema so that the least upper bound is $(\bigsqcup_n d_n, \bigsqcup_n d_n')$. Because every element of the right-projection chain is component-wise greater than the left-projection, $\bigsqcup_n d_n'$ must also be an upper bound of the left-projection chain. Then by the definition of least upper bound, $\bigsqcup_n d_n \sqsubseteq \bigsqcup_n d_n'$, so the least upper bound is in the subset.

It is a more straightforward verification that the subset $\{(x, y) \in D \times D \mid x = y\}$ is chain-closed. Clearly, the component-wise least upper bounds must be equal because the both projections are equal, so the least upper bound is in the subset.

Next, we verify that for all continuous functions $f : D \to E$, CPOs D and E, and chain-closed subsets $S \subseteq E$,

$$f^{-1}S = \{x \in D \mid f(x) \in S\}$$

is a chain-closed subset of D. Because S is chain-closed, the image of the least upper bound of every chain in $f^{-1}S$ under f lies in S. Then because f is continuous, it preserves least upper bounds, so the reverse image of the least upper bound of the chain in S must be the least upper bound of the chain in D. Thus, it lies in the subset.

Next, let S and T be chain-closed subsets of the CPO D. We verify that their union and intersection are chain-closed subsets. For the intersection, this is trivial because every chain in $S \cap T$ is also a chain in S and T, so its least upper bound is in S and T. We observe that any chain in $S \cup T$ can have any elements not in S removed so that it becomes a chain in S, and likewise for T. The least upper bound of the former chain lies in S, and the latter in T. The larger of these two least upper bounds can be taken as the least upper bound of the entire chain, which thus lies in $S \cup T$. Finally, we verify that the intersection of a family of chain-closed subsets of D is chain-closed, which follows trivially from the proof that the intersection of two subsets is chain-closed if you don't care about infinity. □

Problem 4.2. Give an example of a subset $S \subseteq D \times D'$ of a product CPO that is not chain-closed, but which satisfies both of the following:

(a) for all $d \in D$, $\{d' \mid (d, d') \in S\}$ is a chain-closed subset of D'; and

(b) for all $d' \in D'$, $\{d \mid (d, d') \in S\}$ is a chain-closed subset of D.

Solution. TODO: i dont know □

5 PCF

Problem 5.1. Carry out the suggested proof of Proposition 5.4.1.

Solution. The proposition states that evaluation in PCF is deterministic: if both $M \Downarrow_\tau V$ and $M \Downarrow_\tau V'$, then $V = V'$. The proof is by induction on the evaluation rules. For the $\Downarrow_{\text{val}}$ case, the proposition holds by reflexivity.

For $\Downarrow_{\text{succ}}$, if $\mathbf{succ}(M) \Downarrow_{nat} \mathbf{succ}(V)$ and $\mathbf{succ}(M) \Downarrow_{nat} \mathbf{succ}(V')$, we have by the induction hypothesis that $V = V'$. Then by functional equality, $\mathbf{succ}(V) = \mathbf{succ}(V')$.

For $\Downarrow_{\text{pred}}$, if $\mathbf{pred}(M) \Downarrow_{nat} V$ and $\mathbf{pred}(M) \Downarrow_{nat} V'$, we have by the induction hypothesis that $\mathbf{succ}(V) = \mathbf{succ}(V')$. Because $\mathbf{succ}$ is injective, this implies that $V = V'$.

For $\Downarrow_{\text{zero1}}$ and $\Downarrow_{zero2}$, we see by the induction hypothesis that M evaluates to either $\mathbf{0}$ or $\mathbf{succ}(V)$ but not both, so $\mathbf{zero}(M)$ evaluates to either $\mathbf{true}$ or $\mathbf{false}$ but not both.

Similarly for $\Downarrow_{\text{if1}}$ and $\Downarrow_{\text{if2}}$, the condition is either $\mathbf{true}$ or $\mathbf{false}$ but not both, and the triggered branch is deterministic, so the if statement is deterministic.

For β-reduction, we have determinism of substitution (modulo α-equivalence). Finally, $\Downarrow_{\text{fix}}$ uses determinism of β-reduction and substitution. □

Problem 5.2. Recall that Church's fixpoint combinator in the untyped lambda calculus is $Y \stackrel{\text{def}}{=} \lambda f.(\lambda x.f(x\,x))(\lambda x.f(x\,x))$. Show that there are no PCF types τ_1, τ_2, τ_3 so that the typing relation

$$\emptyset \vdash \mathbf{fn}\, f : \tau_1.(\mathbf{fn}\, x : \tau_2.f(x\,x))(\mathbf{fn}\, x : \tau_2.f(x\,x)) : \tau_3$$

is provable from the axioms and rules in Figure 2.

Solution. Suppose for the sake of contradiction that there do exist τ_1, τ_2, and τ_3 such that the typing relation holds. By application of the $:_{\text{fn}}$ rule, we see that $\tau_3 = \tau_1 \to \tau$ for some τ and $f \mapsto \tau_1 \vdash (\mathbf{fn}\, x : \tau_2.f(x\,x))(\mathbf{fn}\, x : \tau_2.f(x\,x)) : \tau$. Next, by the $:_{\text{app}}$ rule, we get that $f \mapsto \tau_1 \vdash \mathbf{fn}\, x : \tau_2.f(x\,x) : \tau_2 \to \tau$ and $f \mapsto \tau_1 \vdash \mathbf{fn}\, x : \tau_2.f(x\,x) : \tau$. However, both premises use the same term, so Proposition 5.3.1(i) tells us that $\tau_2 \to \tau = \tau$, which is impossible. □

Problem 5.3. Define the following PCF terms:

$$\begin{aligned} plus \stackrel{\text{def}}{=}\ & \mathbf{fix}(\mathbf{fn}\, p : nat \to (nat \to nat).\mathbf{fn}\, x : nat.\mathbf{fn}\, y : nat. \\ & \quad \mathbf{if}\ \mathbf{zero}(y)\ \mathbf{then}\ x\ \mathbf{else}\ \mathbf{succ}(p\, x\, \mathbf{pred}(y))) \end{aligned}$$

$$\begin{aligned} times \stackrel{\text{def}}{=}\ & \mathbf{fix}(\mathbf{fn}\, t : nat \to (nat \to nat).\mathbf{fn}\, x : nat.\mathbf{fn}\, y : nat. \\ & \quad \mathbf{if}\ \mathbf{zero}(y)\ \mathbf{then}\ \mathbf{0}\ \mathbf{else}\ plus(t\, x\, \mathbf{pred}(y))\, x) \end{aligned}$$

$$\begin{aligned} fact \stackrel{\text{def}}{=}\ & \mathbf{fix}(\mathbf{fn}\, f : nat \to (nat \to nat).\mathbf{fn}\, x : nat. \\ & \quad \mathbf{if}\ \mathbf{zero}(x)\ \mathbf{then}\ \mathbf{succ(0)}\ \mathbf{else}\ times\, x(f\, \mathbf{pred}(x))). \end{aligned}$$

Show by induction on $n \in \mathbb{N}$ that for all $m \in \mathbb{N}$

$$\begin{aligned} plus\, \mathbf{succ}^m(\mathbf{0})\, \mathbf{succ}^n(\mathbf{0}) &\Downarrow_{nat} \mathbf{succ}^{m+n}(\mathbf{0}) \\ times\, \mathbf{succ}^m(\mathbf{0})\, \mathbf{succ}^n(\mathbf{0}) &\Downarrow_{nat} \mathbf{succ}^{m*n}(\mathbf{0}) \\ fact\, \mathbf{succ}^n(\mathbf{0}) &\Downarrow_{nat} \mathbf{succ}^{n!}(\mathbf{0}) \end{aligned}$$

Solution. For the base case, we see that

$$\begin{aligned} plus\, \mathbf{succ}^m(\mathbf{0})\, \mathbf{0} \Downarrow_{nat}\ & \mathbf{if}\ \mathbf{zero}(\mathbf{0})\ \mathbf{then}\ \mathbf{succ}^m(\mathbf{0}) \\ & \quad \mathbf{else}\ \mathbf{succ}(plus\, \mathbf{succ}^m(\mathbf{0})\, \mathbf{pred}(\mathbf{0})) \\ = \mathbf{succ}^{m+0}(\mathbf{0}), & \end{aligned}$$

so the statement holds. For the inductive case,

$$\begin{aligned} plus\,\mathbf{succ}^m(\mathbf{0})\,\mathbf{succ}^n(\mathbf{0}) \Downarrow_{nat} &\mathbf{if}\ \mathbf{zero}(\mathbf{succ}^n(\mathbf{0}))\ \mathbf{then}\ \mathbf{succ}^m(\mathbf{0}) \\ &\quad \mathbf{else}\ \mathbf{succ}(plus\,\mathbf{succ}^m(\mathbf{0})\,\mathbf{pred}(\mathbf{succ}^n(\mathbf{0})) \\ &= \mathbf{succ}(plus\,\mathbf{succ}^m(\mathbf{0})\,\mathbf{succ}^{n-1}(\mathbf{0})) \\ &= \mathbf{succ}(\mathbf{succ}^{m+n-1}(\mathbf{0})) = \mathbf{succ}^{m+n}(\mathbf{0}). \end{aligned}$$

You get the idea. □

6 Denotational Semantics of PCF

Problem 6.1. Suppose

$$\Gamma \vdash M : \tau$$
$$\Gamma[x \mapsto \tau] \vdash M' : \tau'$$

(so that by Proposition 5.3.1(ii) we also have $\Gamma \vdash M'[M/x] : \tau'$). Then for all $\rho \in [\![\Gamma]\!]$

$$[\![\Gamma \vdash M'[M/x]]\!](\rho) = [\![\Gamma[x \mapsto \tau] \vdash M']\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]).$$

Solution. We proceed by induction on M'. If M' is a constant (**0**, **true**, or **false**), then x does not appear free in M', so the statement holds trivially. If $M' = x$, then

$$[\![\Gamma \vdash x[M/x]]\!](\rho) = [\![\Gamma \vdash M]\!](\rho) = [\![\Gamma[x \mapsto \tau] \vdash x]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]).$$

If $M' = \mathbf{succ}(N)$, then

$$[\![\Gamma \vdash \mathbf{succ}(N)[M/x]]\!](\rho) = [\![\Gamma \vdash \mathbf{succ}(N[M/x])]\!](\rho)$$
$$= \begin{cases} [\![\Gamma \vdash N[M/x]]\!](\rho) + 1 & \text{if } [\![\Gamma \vdash N[M/x]]\!](\rho) \neq \bot \\ \bot & \text{if } [\![\Gamma \vdash N[M/x]]\!](\rho) = \bot. \end{cases}$$
$$= \begin{cases} [\![\Gamma[x \mapsto \tau] \vdash N]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]) + 1 & \text{if } [\![\Gamma[x \mapsto \tau] \vdash N]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]) \neq \bot \\ \bot & \text{if } [\![\Gamma[x \mapsto \tau] \vdash N]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]) = \bot \end{cases}$$
$$= [\![\Gamma[x \mapsto \tau] \vdash \mathbf{succ}(N)]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]).$$

Using the same logic of passing the substitution into the operation, we can see that the same holds for **pred**, **zero**, and **if** (using the inductive hypothesis on M_1 so see that the same branch of the if statement is triggered and then the inductive hypothesis on that branch to see that they evaluate to the same thing). If $M' = M_1 M_2$,

$$\begin{aligned} &[\![\Gamma \vdash (M_1 M_2)[M/x]]\!](\rho) \\ &= [\![\Gamma \vdash (M_1[M/x])(M_2[M/x])]\!](\rho) \\ &= ([\![\Gamma \vdash M_1[M/x]]\!](\rho))([\![\Gamma \vdash M_2[M/x]]\!](\rho)) \\ &= ([\![\Gamma[x \mapsto \tau] \vdash M_1]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]))([\![\Gamma[x \mapsto \tau] \vdash M_2]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]])) \\ &= [\![\Gamma[x \mapsto \tau] \vdash M_1 M_2]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]). \end{aligned}$$

If $M' = \mathbf{fn}\ y : \tau'.N$ where $y \neq x$ and $y \notin dom(\Gamma)$, then

$$\begin{aligned}
&[\![\Gamma \vdash (\mathbf{fn}\ y : \tau'.N)[M/x]]\!](\rho) \\
&= \lambda d \in [\![\tau']\!].[\![\Gamma[y \mapsto \tau'] \vdash N[M/x]]\!](\rho[y \mapsto d]) \\
&= \lambda d \in [\![\tau']\!].[\![\Gamma[y \mapsto \tau'][x \mapsto \tau] \vdash N]\!](\rho[y \mapsto d][x \mapsto [\![\Gamma[y \mapsto \tau'] \vdash M]\!]]) \\
&= [\![\Gamma[x \mapsto \tau] \vdash \mathbf{fn}\ y : \tau'.N]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]])
\end{aligned}$$

by the initial assumption that y does not appear free in Γ and probably some theorem we have that you can remove a substitution when it doesn't appear free. Finally, if $M' = \mathbf{fix}(N)$,

$$\begin{aligned}
[\![\Gamma \vdash \mathbf{fix}(N)[M/x]]\!](\rho) &= \mathit{fix}([\![\Gamma \vdash N[M/x]]\!](\rho)) \\
&= \mathit{fix}([\![\Gamma[x \mapsto \tau] \vdash N]\!])(\rho[x \mapsto [\![\Gamma \vdash M]\!]]) \\
&= [\![\Gamma[x \mapsto \tau] \vdash \mathbf{fix}(N)]\!](\rho[x \mapsto [\![\Gamma \vdash M]\!]]).
\end{aligned}$$

Phew. □

Problem 6.2. Defining $\Omega_\tau \stackrel{\text{def}}{=} \mathbf{fix}(\mathbf{fn}\ x : \tau.x)$, show that $[\![\Omega_\tau]\!]$ is the least element $\bot$ of the domain $[\![\tau]\!]$. Deduce that $[\![\mathbf{fn}\ x : \tau.\Omega_\tau]\!] = [\![\Omega_{\tau\to\tau}]\!]$.

Solution. By unfolding definitions, we see that

$$[\![\Omega_\tau]\!] = [\![\emptyset \vdash \mathbf{fix}(\mathbf{fn}\ x : \tau\,.\,x)]\!](\bot) = \mathit{fix}(\lambda d \in [\![\tau]\!].d).$$

By Tarski's Fixed Point Theorem,

$$\mathit{fix}(\lambda d \in [\![\tau]\!].d) = \bigsqcup_{n\geq 0} (\lambda d \in [\![\tau]\!].d)^n(\bot) = \bigsqcup_{n\geq 0} \bot = \bot.$$

Because denotation is a congruence, we can rewrite $[\![\Omega_\tau]\!] = \bot$ so that

$$[\![\mathbf{fn}\ x : \tau\,.\,\Omega_\tau]\!] = [\![\emptyset \vdash \mathbf{fn}\ x : \tau\,.\,\bot]\!](\bot) = \lambda d \in [\![\tau]\!].\bot.$$

Because it is undefined everywhere, it must be the least element of $[\![\tau \to \tau]\!]$ and thus it is $[\![\Omega_{\tau\to\tau}]\!]$. □

7 Relating Denotational and Operational Semantics

Problem 7.1. For any PCF type τ and any closed terms $M_1, M_2 \in \mathrm{PCF}_\tau$, show that

$$\forall V : \tau\,(M_1 \Downarrow_\tau V \Leftrightarrow M_2 \Downarrow_\tau V) \Rightarrow M_1 \cong_{\mathrm{ctx}} M_2 : \tau.$$

Solution. To prove that $M_1 \cong_{\mathrm{ctx}} M_2 : \tau$, it suffices to show that $M_1 \leq_{\mathrm{ctx}} M_2 : \tau$ and $M_2 \leq_{\mathrm{ctx}} M_1 : \tau$, and by the Proposition on Slide 68 it suffices to show $[\![M_1]\!] \lhd_\tau M_2$ and $[\![M_2]\!] \lhd_\tau M_1$. Both of these are discharged by applying Lemma 7.2.1(iii), instantiating $d_1 = d_2 = [\![M_1]\!]$ and $d_1 = d_2 = [\![M_2]\!]$ for the second. By reflexivity, we have that $[\![M_1]\!] \sqsubseteq [\![M_1]\!]$ and by the Fundamental Property we have $[\![M_1]\!] \lhd_\tau M_1$, and by assumption we have that M_1 and M_2 evaluate to the same value. □

Problem 7.2. Use the statement in 7.1 to show that β-conversion is valid up to contextual equivalence in PCF, in the sense that for all $\mathbf{fn}\ x : \tau\,.\,M_1 \in \mathrm{PCF}_{\tau\to\tau'}$ and $M_2 \in \mathrm{PCF}_\tau$

$$(\mathbf{fn}\ x : \tau\,.\,M_1)M_2 \cong_{\mathrm{ctx}} M_1[M_2/x] : \tau'.$$

Solution. By the above, it suffices to show that

$$\forall V : \tau'\,((\mathbf{fn}\ x : \tau\,.\,M_1)M_2 \Downarrow_{\tau'} V \Leftrightarrow M_1[M_2/x] \Downarrow_{\tau'} V)).$$

We first prove the forward direction by supposing that $(\mathbf{fn}\ x : \tau\,.\,M_1)M_2 \Downarrow_{\tau'} V$. By observing our big step semantics, the only rule that can apply is $\Downarrow_{\mathrm{cbn}}$, giving us the assumptions $\mathbf{fn}\ x : \tau\,.\,M_1 \Downarrow_{\tau\to\tau'} \mathbf{fn}\ x : \tau\,.\,M'$ and $M'[M_2/x] \Downarrow_{\tau'} V$. Because functions are already values, we see from the first assumption that $M' = M_1$, so the second assumption becomes $M_1[M_2/x] \Downarrow_{\tau'} V$, which is exactly what we wished to prove. In the reverse direction, the exact same logic applies, allowing us to apply the $\Downarrow_{\mathrm{cbn}}$ rule. □