1 Real PCF−

We give below the denotational and operational semantics for Real PCF−. The types are as follows:

\[ \sigma, \tau, \ldots ::= \text{unit} \]
\[ | \ I \]
\[ | \ \sigma \rightarrow \tau \]

\( S = \{ \perp, \top \} \) with \( \perp < \top \). \( I = [0, 1] \) with the usual order.

\( \llbracket \text{unit} \rrbracket = S \quad \llbracket I \rrbracket = I \quad \llbracket \sigma \rightarrow \tau \rrbracket = [\llbracket \sigma \rrbracket \rightarrow \llbracket \tau \rrbracket]. \)

\[
\llbracket x \rrbracket \rho = \rho(x) \\
\llbracket uv \rrbracket \rho = \llbracket u \rrbracket \rho(\llbracket v \rrbracket \rho)
\]

\[
\llbracket \text{fn} \ x_\sigma . u \rrbracket \rho = (V \in \llbracket \sigma \rrbracket \mapsto \llbracket u \rrbracket (\rho[x_\sigma \mapsto V]))
\]

\[
\llbracket \text{letrec} \ x_\sigma = u \ \text{in} \ v \rrbracket \rho = \llbracket v \rrbracket (\rho[x_\sigma \mapsto \llbracket \text{rec} \ x_\sigma = u \rrbracket \rho])
\]

\[
\llbracket \text{rec} \ x_\sigma = u \rrbracket \rho = \text{lfp}(V \in \llbracket \sigma \rrbracket \mapsto \llbracket u \rrbracket (\rho[x_\sigma \mapsto V]))
\]

\[
\llbracket 0.u \rrbracket \rho = \text{add}_0(\llbracket u \rrbracket \rho) \\
\llbracket \text{t}_10u \rrbracket \rho = \text{rem}_0(\llbracket u \rrbracket \rho)
\]

\[
\llbracket \text{pif} \ u \ \text{then} \ v \ \text{else} \ w : \tau \rrbracket \rho = \begin{cases} 
\llbracket v \rrbracket \rho, & \text{if } \llbracket u \rrbracket \rho = \top \\
\llbracket v \rrbracket \rho \land \llbracket w \rrbracket \rho, & \text{if } \llbracket u \rrbracket \rho = \perp
\end{cases}
\]

\[
\llbracket u > 1/2 \rrbracket \rho = \begin{cases} 
\top, & \text{if } \llbracket u \rrbracket \rho > 1/2 \\
\perp, & \text{otherwise}
\end{cases}
\]

\[
\llbracket \ast \rrbracket \rho = \top
\]

where \( V \in X \mapsto f(V) \) denotes the function which to all \( V \) in \( X \) associates \( f(V) \), and where:

\[
\text{add}_0(a) = a/2 \quad \text{add}_1(a) = (a + 1)/2
\]

\[
\text{rem}_0(a) = \min(2a, 1) \quad \text{rem}_1(a) = \max(2a - 1, 0)
\]
Contexts (type constraints omitted):
\[
C ::= \_ | Cv | t1_0C | t1_1C | C > 1/2 | C > 0 | \text{pif } C \text{ then } v \text{ else } w | \text{pif } u \text{ then } C \text{ else } w | \text{pif } u \text{ then } v \text{ else } C
\]

Operational semantics. We only apply a rule under a context \(C\) of the above form, i.e., \(u \rightarrow v\) if and only if \(u = C[\ell]\) and \(v = C[r]\), where \(C\) is a context (the types being respected), and \(\ell \rightarrow r\) is one of the rules below.

\[
\begin{align*}
\text{(fn } x_\sigma u) v & \rightarrow u[x_\sigma := v] \\
\text{letrec } x_\sigma = u \text{ in } v & \rightarrow v[x_\sigma := \text{letrec } x_\sigma = u \text{ in } u] \\
\text{t1}_a(u) & \rightarrow u \\
\text{t1}_0(u) & \rightarrow 1 \\
\text{t1}_1(u) & \rightarrow 0 \\
(1.u) > 1/2 & \rightarrow u > 0 \\
(1.u) > 0 & \rightarrow * \\
(0.u) > 0 & \rightarrow u > 0 \\
\text{pif } * \text{ then } v \text{ else } w & \rightarrow v \\
\text{pif } u \text{ then } v \text{ else } * & \rightarrow v \\
\text{pif } u \text{ then } 0.v \text{ else } 1.w & \rightarrow 0.v \\
\text{pif } u \text{ then } a.v \text{ else } a.w & \rightarrow a.\text{pif } u \text{ then } v \text{ else } w \\
& \quad (a \in \{0,1\})
\end{align*}
\]

Exercise 1:
Recall that for all \(u : \tau, [u]\) is a well-defined function, Scott-continuous from \(Env \overset{\text{def}}{=} \prod_{x \text{variable}, [\sigma]} [\sigma] \rightarrow [\tau]\).

1. Show that the construction \(u > 0\) of Real PCF\(^-\) is redundant. Explicitly propose a definition of an expression Real PCF\(^-\) nonzero, of type \(I \rightarrow \text{unit}\), which does not use the expression of the form \(u > 0\), and whose semantics \([\text{nonzero}]\rho\) is the function to which 0 associates \(\perp\) and to all \(a \in I\) non-zero associates \(\top\). Prove this assertion.

2. Show that the rule tagged with (\(\alpha\)) of the operational semantics is correct, in the sense that \([\text{pif } u \text{ then } v \text{ else } *]\rho = [v]\rho\) for all \(\rho \in Env\).

3. We consider a Real PCF\(^-\) program of the form \text{letrec } x_\sigma = u \text{ in } v, of type \text{unit}.

Show that if \([\text{letrec } x_\sigma = u \text{ in } ]\rho \neq \perp\), then there is an integer \(n \in \mathbb{N}\) such that
\[
[\text{letrec } x_\sigma = u \text{ in } ]\rho = g(f^n(\perp)),
\]
where we use the abbreviations \(g(V) = [v](\rho[x_\sigma \mapsto V])\) and \(f(V) = [u](\rho[x_\sigma \mapsto V])\).
(The \(\perp\) in argument of \(f^n\) is that of \([\sigma]\).) This expresses that a recursive definition (of \(x_\sigma\)) used in a terminating computation (\(v\)) of type \text{unit} will be "expanded" only \(n\) times.
4. Why does the argument from the previous question not work if \( \text{letrec } x_r = u \text{ in } v \) is of type \( \mathcal{I} \)?

5. Recall that \( 0 \overset{\text{def}}{=} \text{letrec } x_1 = 0.x_1 \text{ in } x_1 \). Show that there does not exist a derivation in the operational semantics for

\[ t_1_0(\text{pif } 0 > 1/2 \text{ then } 1.\dot{0} \text{ else } 0.1.1.\dot{0}) > 1/2 \rightarrow^{*} \ast. \]

We can set \( Z \overset{\text{def}}{=} \text{letrec } x_1 = 0.x_1 \text{ in } 0.x_1 \).

6. What can we conclude for the adequacy of the type \( \text{unit} \)? Justify.

7. Any suggestions to complete the operational semantics?

Solution:

1.

\[
\text{rec \ nonzero = fn \ m_1.} \\
\text{\quad pif \ m > 1/2 \text{ then } \ast} \\
\text{\quad else \ nonzero(t_1_0 m)}
\]

Its semantics is the smallest fixed point of the function \( F \) which to \( \phi \in [\mathcal{I} \rightarrow \mathcal{S}] \) associates the function which to \( a \in \mathcal{I} \) associates \( \top \) if \( a > 1/2 \), \( \varphi(\text{max}(2a,1)) = \varphi(2a) \) otherwise. (The \( \land \) is trivial here.)

The iterations of Kleene are \( \phi_0 = \bot \), then \( \phi_1 \) which associates \( \top \) exactly with \( a > 1/2 \), then \( \phi_2 \) which associates \( \top \) exactly with \( a \) such that \( a > 1/2 \) or \( 2a > 1/2 \) (i.e. \( a > 1/4 \)).

By induction on \( n \), we see that \( \phi_n \) associates \( \top \) exactly with \( a > 1/2 \). In effect, \( \phi_{n+1} \) sends all \( a > 1/2 \) to \( \top \), and all \( \text{aleq}1/2 \) to \( \phi_n(2a) \), that is to say to \( \top \) if \( 2a > 1/2^n \) (i.e., \( a > 1/2^{n+1} \)) and to \( \bot \) otherwise.

The smallest fixed point therefore always sends \( 0 \) to \( \bot \), but any number \( a > 0 \) to \( \top \) since there is an \( n \in \mathbb{N} \) from which \( a > 1/2^n \).

2. If \( \| u \| \rho > 1/2 \), the left side is \( \| v \| \rho \). Otherwise, it is worth \( \| v \| \rho \land \| \ast \| \rho = \| v \| \rho \) since \( \| \ast \| \rho = \top \) is the largest element of \( \mathcal{S} \) (and everything happens in \( \mathcal{S} \) given the typing constraints).

3. By definition, \( \| \text{letrec } x_r = u \text{ in } v \| \rho = g(\text{lfp} f) \). Using Kleene’s formula, and the Scott-continuity of \( g \), this is sup_{n \in \mathbb{N}}(f^n(\bot)). The dcpo \( \| u \| = \mathcal{S} \) is flat, so this sup is reached for a certain \( n \). Note that we must use the Scott-continuity of \( g \). There is no \( n \in \mathbb{N} \) such that \( \text{fn } (\bot) = \text{lfp} f \) in general, as the next question shows.

4. I does not have the ascending string property. For example, the definition of \( \dot{1} \) produces such an infinite growing chain.

5. Expressions \( 1.\dot{0} \) and \( 0.1.1.\dot{0} \) are in normal form because \( 1.\_ \) and \( 0.\_ \) are not contexts. In fact, we can only start by rewriting \( \dot{0} > 1/2 \) in \( Z > 1/2 \), then in \( 0.Z > 1/2 \), which gives \( t_1_0(\text{pif } 0.Z > 1/2 \text{ then } 1.\dot{0} \text{ else } 0.1.1.\dot{0}) > 1/2 \). But there is no longer any rule applicable to this expression.

6. It fails. Indeed, for any environment \( \rho \), such as \( \| 0.Z > 1/2 \| \rho = \text{add}_0(0) = 0 \),

\[
\| t_1_0(\text{pif } 0.Z > 1/2 \text{ then } 1.\dot{0} \text{ else } 0.1.1.\dot{0}) \| \rho = \text{rem}_0(1.\dot{0}) \rho \land \| 0.1.1.\dot{0} \| \rho \\
= \text{rem}_0(\text{add}_1(0) \land \text{add}_0(\text{add}_1(\text{add}_1(0)))) \\
= \text{rem}_0(1/2 \land 3/8) = \text{rem}_0(3/8) = 3/4
\]

So \( \| t_1_0(\text{pif } 0.Z > 1/2 \text{ then } 1.\dot{0} \text{ else } 0.1.1.\dot{0}) > 1/2 \| \rho = \ast \), but we come to see that the operational semantics does not progress far enough to reach \( \ast \).
7. We can already add the rules:

\[
\text{ti}_a(pif \; u \; \text{then} \; v \; \text{else} \; w) \rightarrow pif \; u \; \text{then} \; ti_a v \; \text{else} \; ti_a w
\]

\[
(pif \; u \; \text{then} \; v \; \text{else} \; w) > 1/2 \rightarrow pif \; u \; \text{then} \; v > 1/2 \; \text{else} \; w > 1/2
\]

\[
(pif \; u \; \text{then} \; v \; \text{else} \; w) > 0 \rightarrow pif \; u \; \text{then} \; v > 0 \; \text{else} \; w > 0
\]

for \( a \in \{0,1\} \). The third form is not essential for the example, but we can see that it will be necessary in general. We could also think of adding rules like \((pif \; u \; \text{then} \; v \; \text{else} \; w)t \rightarrow pif \; u \; \text{then} \; vt \; \text{else} \; wt\), but this is not necessary, because with the adecuation of type \text{unit}, the functions play little role.

**Exercise 2:**

We now assume that a same Real PCF− variable is always labeled with the same type: if we see \( x_\sigma \) and \( x_\tau \), then \( \sigma = \tau \). This amounts to saying that the name \( x \) of the variables is sufficient to distinguish them.

We consider the Real PCF−− language, which is just Real PCF− but without any type index. For example, \( \text{fn} \; x.\sigma.\; u \) and \( \text{letrec} \; x = u \; \text{in} \; v \) are the expressions Real PCF−− corresponding to \( \text{fn} \; x_\sigma.\; u \) and \( \text{letrec} \; x_\sigma = u \; \text{in} \; v \), respectively.

Formally, let \( E \) denote the type erasure function, defined by:

\[
E(\text{letrec} \; x_\sigma = u \; \text{in} \; v) \overset{\text{def}}{=} \text{letrec} \; x = E(u) \; \text{in} \; E(v), \quad E(\text{fn} \; x_\sigma.\; u) \overset{\text{def}}{=} \text{fn} \; x.\; E(u), \quad \text{etc.}
\]

We will say that a Real PCF−− expression \( u \) is typable, of type \( \tau \), if and only if there exists a Real PCF− expression \( u' \), of type \( \tau \), such that \( E(u) = u \).

1. Are all Real PCF−− expressions typable? Justify.

2. Is the type of a Real PCF−− typable expression unique? Justify.

**Solution:**

1. No, for example, \( xx \) is not, neither is \( * + 0 \).

2. No, for example, \( \text{fn} \; x.x \) has all types \( \sigma \rightarrow \sigma \).