Scribe: Bryant Adams and Olga Belomestnykh Lecturer: Andrew Myers

1 Continuous functions form a CPO

1.1 Motivation

In a pointed CPO, we have a way to get our hands on least upper bounds. In our metalanguage, everything we work with is either a CPO or a continuous function. For the things that are already CPOs, taking fixed points is straightforward. What about for our operations, however? It would be nice to know that continuous functions themselves form a CPO. And, indeed, they do.

Essentially, we want to show that $\bigsqcup_n f_n$ is continuous (i.e. given chains f_n, d_m that $(\bigsqcup_n f_n)(\bigsqcup_m d_m) = \bigsqcup_m((\bigsqcup_n f_n)d_m)$.)

1.2 Attempt

It would be nice to try to argue as follows:

$$\begin{aligned} \textit{Start with} & (\bigsqcup_n f_n)(\bigsqcup_m d_m) \\ \textit{Since LUB is defined pointwise} &= (\bigsqcup_n (f_n \bigsqcup_m d_m)) \\ \textit{By continuity of } f_n &= (\bigsqcup_n (\bigsqcup_m f_n(d_m))) \\ \textit{By wishful thinking} &= (\bigsqcup_m (\bigsqcup_n f_n(d_m))) \\ \textit{Since LUB is defined pointwise} &= \bigsqcup_m ((\bigsqcup_n f_n) d_m) \end{aligned} \tag{1}$$

Alas, wishful thinking proves little. If we could just see that joins ($\bigsqcup$) commute, though... or, at least, that they commute when dealing with monotonic functions f_n (which our continuous functions are)... then we'd be in good shape.

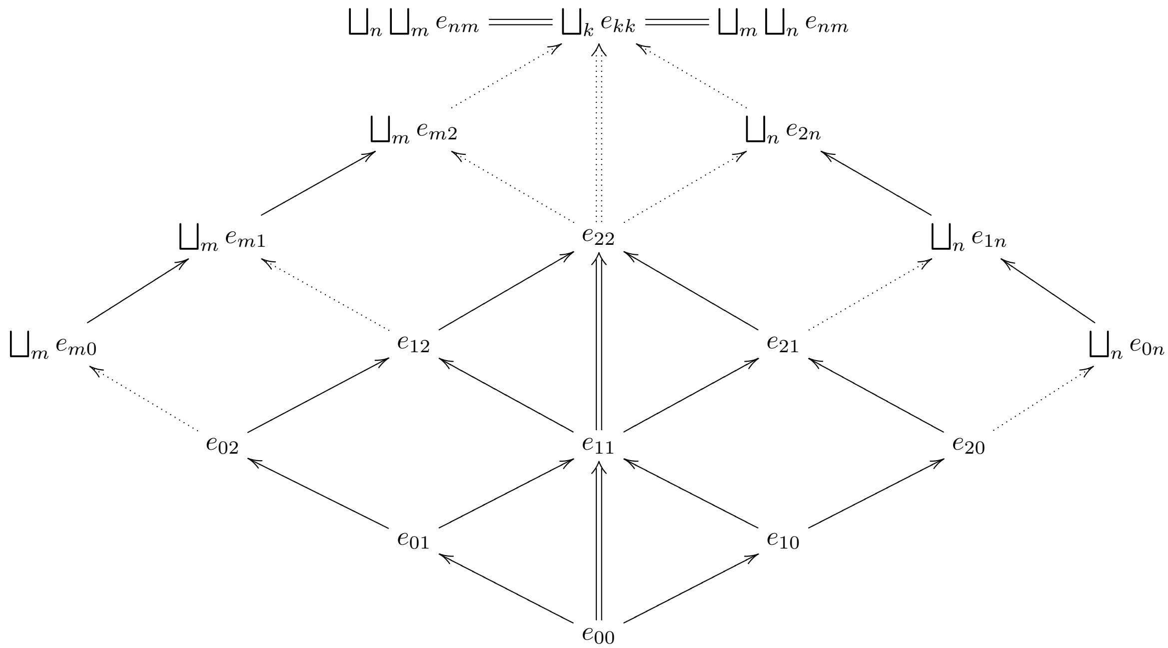
1.3 Proof

Theorem 1.1. *Given a chain of monotonic functions f_n and a chain of arguments d_m, it is the case that $\bigsqcup_n \bigsqcup_m f_n(d_m) = \bigsqcup_m \bigsqcup_n f_n(d_m)$.*

Proof. We will introduce a lemma which does all the real work of the theorem:

Lemma 1.2. *Given a bi-indexed infinite chain e_{nm} such that $e_{nm} \sqsubseteq e_{n'm'}$ iff ($n \leq n'$ and $m \leq m'$), it is the case that $\bigsqcup_n \bigsqcup_m e_{nm} = \bigsqcup_k e_{kk} = \bigsqcup_m \bigsqcup_n e_{nm}$*

The picture we are working within is an infinite 'square' (if we were in a finite case, we could have a rectangle, but if we were in a finite case, the least upper bounds would just be the maxima and there'd be nothing to worry about.)



Proof. Consider some k. Certainly, $e_{kk} \sqsubseteq \bigsqcup_n e_{nk}$ for each k, and so $\bigsqcup_m e_{mm} \sqsubseteq \bigsqcup_m \bigsqcup_n e_{nm}$.

Now, note that for all n and m, there exists k such that $k \geq m$ and $k \geq n$ (in particular, $k = max(m, n)$) and thus for each n, m there is k such that $e_{nm} \sqsubseteq e_{kk}$, so we also have $\bigsqcup_n e_{nm} \sqsubseteq \bigsqcup_k e_{kk}$.

¿From this, we see that $\bigsqcup_m \bigsqcup_n e_{nm} \sqsubseteq \bigsqcup_m e_{mm}$, and thus conclude that $\bigsqcup_m e_{mm} = \bigsqcup_m \bigsqcup_n e_{nm}$

The case with reversed indices is identical, so the lemma is proved. □

To complete the proof of the theorem, we let $e_{nm} = f_n(d_m)$, and note that for $n \geq n', m \geq m'$ we have $e_{nm} \geq e_{n'm'}$ by d_m and f_n being chains, and the f_n being monotonic, so we may apply the lemma. □

2 Denotational Semantics for REC

1. REC Language

$$
\begin{array}{lll}
p & ::= & \textbf{let } d \textbf{ in } e \\
d & ::= & f_1(x_1, \ldots, x_{a_1}) = e_1 \\
 & & \vdots \\
 & & f_n(x_1, \ldots, x_{a_n}) = e_n \\
e & ::= & n \mid x \mid e_1 \oplus e_2 \mid \textbf{let } x = e_1 \textbf{ in } e_2 \mid \textbf{if}_p\ e_0 \textbf{ then } e_1 \textbf{ else } e_2 \mid f_i(e_1, \ldots, e_{a_i})
\end{array}
$$

It is reasonable to expect that under most semantics **let** $f_1(x_1) = f_1(x_1)$ **in** $f_1(0)$ is likely to infinite loop, but **let** $f_1(x_1) = f_1(x_1)$ **in** 0 to return 0.

Example:

$$
\begin{array}{l}
\textbf{let} \\
\quad f_1(n, m) = (n - m * m) \wedge ((n = m * (n \textbf{ div } m)) \vee f_1(n, m + 1)) \\
\quad f_2(n) = f_1(n, 2) \\
\quad f_3(n) = \textbf{if}_p\ f_2(n) \textbf{ then } n \textbf{ else } f_3(n + 1) \\
\textbf{in} \\
\quad f_3(1000)
\end{array}
$$

In this REC program $f_3(n)$ finds the first prime number p such that $p \geq n$.

2. CBV Denotational Semantics for REC

The meaning function is $[\![e]\!] \in$ **FEnv** $\rightarrow$ **Env** $\rightarrow \mathbb{Z}_\perp$. First we must define two environments: one for variables and one for functions.

$$\begin{array}{lll} \rho & \in & \textbf{Env} = Var \rightarrow \mathbb{Z} \\ \phi & \in & \textbf{FEnv} = (\mathbb{Z}^{a_1} \rightarrow \mathbb{Z}_\perp) \times \ \ldots \ \times (\mathbb{Z}^{a_n} \rightarrow \mathbb{Z}_\perp) \end{array}$$

Var is a countable set of variable names. $\mathbb{Z}$ is a set of possible bindings. $\mathbb{Z}^n = \underbrace{\mathbb{Z} \times \mathbb{Z} \times \ldots \times \mathbb{Z}}_{n\ times}$.

$$\begin{array}{rcl} [\![n]\!]\phi\rho & = & \lfloor n \rfloor \\ [\![x]\!]\phi\rho & = & \lfloor \rho\ x \rfloor \\ [\![e_1 \oplus e_2]\!]\phi\rho & = & \textbf{let } v_1 \in \mathbb{Z} = [\![e_1]\!]\phi\rho \textbf{ in} \\ & & \textbf{let } v_2 \in \mathbb{Z} = [\![e_2]\!]\phi\rho \textbf{ in} \\ & & \quad \lfloor v_1 \oplus v_2 \rfloor \\ & = & ([\![e_1]\!]\phi\rho) \oplus_\perp ([\![e_2]\!]\phi\rho) \\ [\![e_1 \wedge e_2]\!]\phi\rho & = & \textbf{let } v_1 \in \mathbb{Z} = [\![e_1]\!]\phi\rho \textbf{ in} \\ & & \quad \textbf{if } v_1 \leq 0 \textbf{ then } 0 \textbf{ else } [\![e_2]\!]\phi\rho \\ [\![\textbf{let } x = e_1 \textbf{ in } e_2]\!]\phi\rho & = & \textbf{let } y \in \mathbb{Z} = [\![e_1]\!]\phi\rho \textbf{ in} \\ & & \quad [\![e_2]\!]\ \phi\ \rho[x \mapsto y] \\ [\![\textbf{if}_p\ e_0 \textbf{ then } e_1 \textbf{ else } e_2]\!]\phi\rho & = & \textbf{let } v_0 \in \mathbb{Z} = [\![e_0]\!]\phi\rho \textbf{ in} \\ & & \quad \textbf{if } v_0 > 0 \textbf{ then} [\![e_1]\!]\phi\rho \textbf{ else } [\![e_2]\!]\phi\rho \\ [\![f_i(e_1, \ldots, e_{a_i})]\!]\phi\rho & = & \textbf{let } v_1 \in \mathbb{Z} = [\![e_1]\!]\phi\rho \textbf{ in} \\ & & \vdots \\ & & \textbf{let } v_{a_i} \in \mathbb{Z} = [\![e_{a_i}]\!]\phi\rho \textbf{ in} \\ & & \quad (\pi_i\ \phi)\langle v_1, \ldots, v_{a_i} \rangle \end{array}$$

- Where does ϕ come from?

$$\begin{array}{rcl} \phi & = & \langle F_1, \ldots, F_n \rangle \\ & = & \mathit{fix}\ \lambda\phi \in \textbf{FEnv}\ .\langle \lambda y_1 \in \mathbb{Z}, \ldots, y_{a_1} \in \mathbb{Z}.[\![e_1]\!]\ \phi\ \{x_1 \mapsto y_1, \ldots, x_{a_1} \mapsto y_{a_1}\}, \\ & & \qquad \vdots \\ & & \qquad \lambda y_1 \in \mathbb{Z}, \ldots, y_{a_n} \in \mathbb{Z}.[\![e_n]\!]\ \phi\ \{x_1 \mapsto y_1, \ldots, x_{a_n} \mapsto y_{a_n}\} \rangle \end{array}$$

- Is **FEnv** a pointed CPO?

  FEnv is a product. A product is a pointed CPO when each $(\mathbb{Z}^{a_i} \rightarrow \mathbb{Z}_\perp)$ is a pointed CPO. A function is a pointed CPO when the codomain of that function is a pointed CPO and $\mathbb{Z}_\perp$ is a pointed CPO. Therefore, **FEnv** is a pointed CPO.
- Is the function that we're applying *fix* to continuous?

  It is written using metalanguages, thus it is indeed continuous.

3. CBN Denotational Semantics

The denotational semantics for CBN are the same as those for CBV with two exceptions:

$$\begin{array}{rcl} [\![\textbf{let } x = e_1 \textbf{ in } e_2]\!]\phi\rho & = & [\![e_2]\!]\ \phi\ \rho[x \mapsto ([\![e_1]\!]\ \phi\ \rho)] \\ [\![f_i(e_1, \ldots, e_{a_i})]\!]\phi\rho & = & (\pi_i\ \phi)\langle\ ([\![e_1]\!]\ \phi\ \rho),\ \ldots,\ [\![e_{a_i}]\!]\ \phi\ \rho)\ \rangle \end{array}$$