## 1 Semantics of while

Let's return to the denotational semantics of the **while** loop. We previously defined the function

$$\begin{aligned} F &: (\Sigma \to \Sigma_\perp) \to (\Sigma \to \Sigma_\perp) \\ F &\triangleq \lambda w \in \Sigma \to \Sigma_\perp.\, \lambda\sigma \in \Sigma.\, \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (w)^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma. \end{aligned}$$

Any function $\Sigma \to \Sigma_\perp$ is continuous, since chains in the discrete space $\Sigma$ contain at most one element, thus the space of functions $\Sigma \to \Sigma_\perp$ is the same as the space of continuous functions $[\Sigma \to \Sigma_\perp]$. Moreover, the lift $(w)^* : \Sigma_\perp \to \Sigma_\perp$ of any function $w : \Sigma \to \Sigma_\perp$ is continuous.

By previous arguments, the function space $[\Sigma \to \Sigma_\perp]$ is a pointed CPO, and $F$ maps this space to itself. To obtain a least fixpoint by the Fixed-Point Theorem, we need to know that $F$ is continuous.

Let's first check that it is monotonic. This will ensure that, when trying to check the definition of continuity, when $C$ is a chain, $\{F(d) \mid d \in C\}$ is also a chain, so that $\bigsqcup_{d\in C} F(d)$ exists. Suppose $d \sqsubseteq d'$. We want to show that $F(d) \sqsubseteq F(d')$. But for all $\sigma$,

$$\begin{aligned} F(d)(\sigma) &= \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (d)^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma \\ &\sqsubseteq \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (d')^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma \\ &= F(d')(\sigma). \end{aligned}$$

Here we have used the fact that the operator $(\cdot)^*$ is monotonic, which is easy to check.

Now let's check that $F$ is continuous. Let $C$ be an arbitrary chain. We want to show that $\bigsqcup_{d\in C} F(d) = F(\bigsqcup C)$. We have

$$\begin{aligned} \bigsqcup_{d\in C} F(d) &= \bigsqcup_{d\in C} \lambda\sigma.\, \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (d)^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma \\ &= \lambda\sigma.\, \bigsqcup_{d\in C} \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (d)^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma \\ &= \lambda\sigma.\, \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } \bigsqcup_{d\in C} (d)^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma \\ &= \lambda\sigma.\, \text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (\bigsqcup C)^*(\mathcal{C}[\![c]\!]\sigma) \text{ else } \sigma \;=\; F(\bigsqcup C), \end{aligned}$$

since $\mathcal{B}[\![b]\!]\sigma$ does not depend on $d$ and since the lift operator $(\cdot)^*$ is continuous.

## 2 A Metalanguage for Domain Constructions

Last time we did several constructions that required us to check that various domains were CPOs and that various associated operations were continuous. How can we avoid doing this kind of check over and over again? One solution is to create an abstract metalanguage consisting of some basic operations that will allow us to do domain constructions (like function spaces, direct products, etc.) and that will ensure that the domains that are constructed are CPOs and the associated functions are continuous. We can compose these constructions to create more complicated domains from simpler ones and always be assured that the desired mathematical properties hold.

The simplest objects will be the discrete CPOs $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{U}$ for the integers, the natural numbers, and the unit domain, respectively. The unit domain contains a single element $unit$. Operations on discrete CPOs are automatically monotonic and continuous because any chain on a discrete CPO can only contain one element. This is why we can use operations like $+$ and (if $\cdot$ then $\cdot$ else $\cdot$) in writing denotational semantics.

For any domain $A$, we can construct a new domain $A_\perp$, which is $A$ adjoined with a new element $\perp$ below all the previous elements. Note that $\perp$ is intended to be a new element, so we can actually iterate this operation. The associated operations are the natural embedding $\lfloor\cdot\rfloor : D \to D_\perp$ and the lifting operation $(\cdot)^* : (D \to E_\perp) \to (D_\perp \to E_\perp)$ defined by

$$\begin{aligned}(d)^*(\lfloor x\rfloor) &\triangleq d(x)\\ (d)^*(\perp) &\triangleq \perp\end{aligned}$$

Both these operations are continuous, and when $(\cdot)^*$ is applied to a continuous function, the result is a continuous function.

As a convenient syntactic sugar, we write let $x \in D = e_1$ in $e_2$ as a shorthand for $(\lambda x \in D\,.\,e_2)^*\ e_1$. That is, let is *strict* in that if $e_1 = \perp$, the let is $\perp$ too.

### 2.1 Products

Given CPOs $D$ and $E$, we can form the product $D \times E$ consisting of all ordered pairs $< d, e >$ with $d \in D$ and $e \in E$, ordered componentwise. This is the set-theoretic Cartesian product of $D$ and $E$ with $< d, e > \sqsubseteq < d', e' >$ iff $d \sqsubseteq d'$ and $e \sqsubseteq e'$. This is a CPO, and it is easy to check that $\bigsqcup_{d\in X,\ e\in Y} < d, e > = < \bigsqcup X, \bigsqcup Y >$. Along with the product constructor come the projections $\pi_1$ and $\pi_2$ defined by $\pi_1(< d, e >) = d$ and $\pi_2(< d, e >) = e$, which are continuous. If $f : C \to D$ and $g : C \to E$, then the function $< f, g >: C \to D \times E$ defined by $< f, g > \triangleq \lambda x.\ < f(x), g(x) >$ is continuous if $f$ and $g$ are. This is the unique function satisfying the equations $f = \pi_1 \circ < f, g >$ and $g = \pi_2 \circ < f, g >$. The binary product can be generalized to an arbitrary product $\prod_{x\in X} D_x$ with associated projections $\pi_y : \prod_{x\in X} D_x \to D_y$.

### 2.2 Sums

Given CPOs $D$ and $E$, we can form the *sum* (or *coproduct*) $D + E$, corresponding to the disjoint union of $D$ and $E$. We would like to take the union of the sets $D$ and $E$, but we need to mark the elements to make sure we can tell which set they originally came from in case $D$ and $E$ have a nonempty intersection. To do this, we define

$$D + E \triangleq \{\mathsf{in}_1(d) \mid d \in D\} \cup \{\mathsf{in}_2(e) \mid e \in E\},$$

where $\mathsf{in}_1$ and $\mathsf{in}_2$ are any one-to-one functions with disjoint ranges; for example, we could take $\mathsf{in}_1(x) = < 1, x >$ and $\mathsf{in}_2(x) = < 2, x >$. We define $\mathsf{in}_i(x) \sqsubseteq \mathsf{in}_j(y)$ iff $i = j$ and $x \sqsubseteq y$. Any chain in $D + E$ must be completely contained in $\{\mathsf{in}_1(x) \mid x \in D\}$ or $\{\mathsf{in}_2(x) \mid x \in E\}$, so $D + E$ is a CPO. The associated operations are the injections $\mathsf{in}_1 : D \to D + E$ and $\mathsf{in}_2 : E \to D + E$, which are continuous. If $f : D \to C$ and $g : E \to C$, then we can combine $f$ and $g$ into a function $f + g : D + E \to C$ using a *case* construct:

$$f + g \triangleq \lambda x.\ \mathit{case}\ x\ \mathit{of}\ \mathsf{in}_1(y) \to f(y) \mid \mathsf{in}_2(y) \to g(y)$$

This is continuous if $f$ and $g$ are, and it is the unique function satisfying the equations $f = (f + g) \circ \mathsf{in}_1$ and $g = (f + g) \circ \mathsf{in}_2$. As with products and projections, the binary coproduct can be generalized to an arbitrary coproduct $\sum_{x\in X} D_x$ with associated injections $\mathsf{in}_y : D_y \to \sum_{x\in X} D_x$.

### 2.3 Continuous functions

Finally, given CPOs $D$ and $E$, we can define the CPO $[D \to E]$ of all continuous functions from $D$ to $E$ with the pointwise ordering.

It is not obvious that continuous functions themselves form a CPO. Essentially, we want to show that $\bigsqcup_n f_n$ is continuous (i.e. given chains $f_n, d_m$ that $(\bigsqcup_n f_n)(\bigsqcup_m d_m) = \bigsqcup_m((\bigsqcup_n f_n)d_m)$.)

### 2.4 Attempt

It would be nice to try to argue as follows:

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

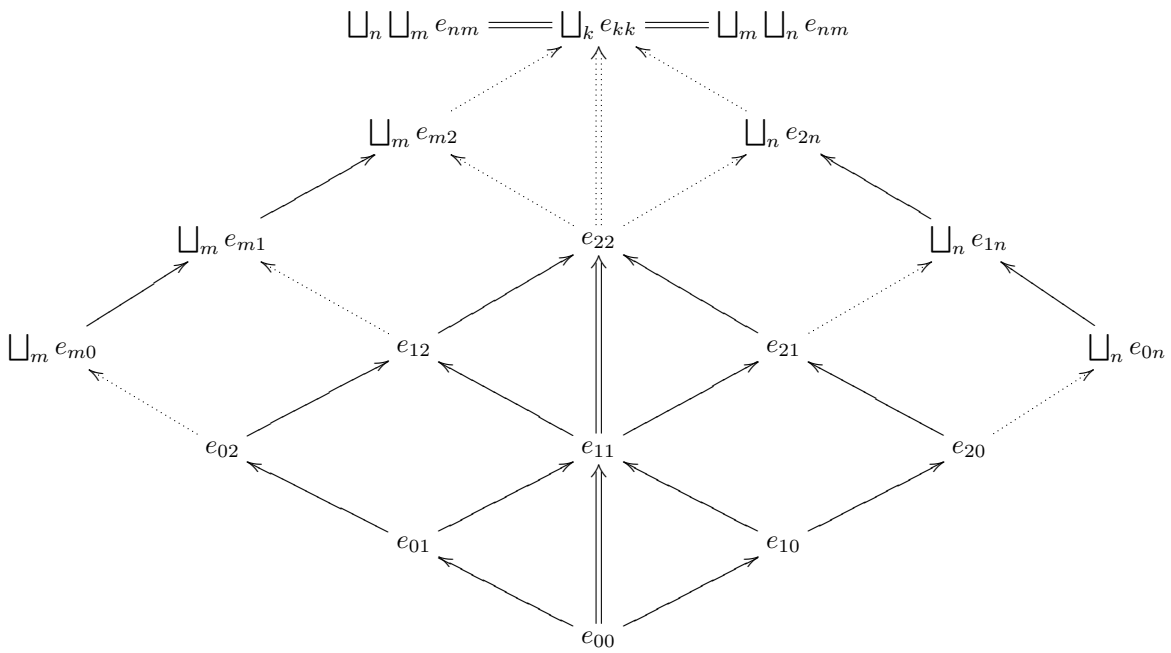
Alas, wishful thinking proves little. We need to show that joins ($\bigsqcup$) commute, at least, that they commute when dealing with continuous functions $f_n$.

### 2.5 Proof

**Theorem 2.1.** *Given a chain of monotonic functions $f_n$ and a chain of arguments $d_m$, we have $\bigsqcup_n \bigsqcup_m f_n(d_m) = \bigsqcup_m \bigsqcup_n f_n(d_m)$.*

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

**Lemma 2.1.** *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 $\bigsqcup_n \bigsqcup_m e_{nm} = \bigsqcup_k e_{kk} = \bigsqcup_m \bigsqcup_n e_{nm}$*




*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 symmetric, 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. □

The operations on continuous functions are:

1. $apply : [D \to E] \times D \to E$ that applies a given function to a given argument;
2. $compose : [E \to F] \times [D \to E] \to [D \to F]$;
3. $curry : [D \times E \to F] \to [D \to [E \to F]]$;
4. $uncurry : [D \to [E \to F]] \to [D \times E \to F]$; and most importantly,
5. $fix : [D \to D] \to D$, defined by $\lambda g \in [D \to D]. \bigsqcup g^n(\bot)$, that takes a function and returns its least fixpoint. To apply **fix**, $D$ must have a bottom element $\bot$.

All these functions are continuous, which can be shown without too much effort.