Kiri Wagstaff

1 More on cpo's

Recall that $\mathcal{C}[\![\textbf{while}\, b\, \textbf{do}\, c]\!]\sigma = \mathit{fix}(\lambda f.\lambda\sigma.\text{ if }\mathcal{B}[\![b]\!]\sigma\text{ then }f(\mathcal{C}[\![c]\!]\sigma)\text{ else }\sigma)$. But we need to know that this function is continuous so that we can find its fixed point. So let's construct a toolbox of 'safe' constructs that will form a meta-language which is guaranteed to provide continuous functions.

The simplest cpo is a *flat* or *discrete* cpo. This means that it consists of a set ordered by $=$, with no $\perp$ element. Because there are no chains, such a cpo has no fixed point. Let U be the cpo with only one element, $\top$, ordered by $=$, i.e.

$$U = \{\top\}, =$$

It is possible to *lift* a domain by adding $\perp$, so that

$$U_\perp = \{\top, \perp\}, \sqsubseteq = \{(\perp, \top), (\top, \top), (\perp, \perp)\}$$

This $U_\perp$ is now a *pointed flat cpo*. In general the domain $D_\perp$ has all of the ordering relations that D does, and in addition $\forall d \in D_\perp \,.\, \perp \sqsubseteq d$. Sometimes we write $\lfloor d \rfloor$ to denote the element in $D_\perp$ that corresponds to the element d in D. This is useful when we want to distinguish between elements of the two domains.

In order to convert some $f : \Sigma \to \Sigma$ to $f' : \Sigma_\perp \to \Sigma_\perp$, we define

$$f'(\sigma) = \begin{cases} f(\sigma) & \text{if } \sigma \neq \perp \\ \perp & \text{if } \sigma = \perp \end{cases}$$

This means that for $\lambda f.\lambda\sigma.\text{ if }\mathcal{B}[\![b]\!]\sigma\text{ then }f(\mathcal{C}[\![c]\!]\sigma)\text{ else }\sigma$ (we have an implicit extra if test so that $f(\perp) = \perp$).

2 Domain constructors

There are five parts to the specification of a domain:

1. What are the elements?
2. What is the ordering relation $\sqsubseteq$?
3. Does $\perp$ exist?
4. What operations are defined on the elements?
5. If the domain is a cpo, what is the lub of all chains?

2.1 Product Domain

We can create domains that allow us to have records, which are known as *product domains*. Given a set of domains $D_1 \ldots D_n$, $D_1 \times D_2 \times \ldots \times D_n$ is a product domain.

1. Elements: $(d_1, d_2, \ldots, d_n)$
2. Ordering: $(d_1, d_2, \ldots, d_n) \sqsubseteq (d_1', d_2', \ldots, d_n') \overset{\text{def}}{\Leftrightarrow} \forall i.(d_i \sqsubseteq_{D_i} d_i')$
3. $\perp$: if $\forall D_i.\exists \perp_i$, then $\perp = (\perp_1, \perp_2, \ldots, \perp_n)$
4. Operations (both are monotonic and continuous):
   - Tuple construction
   - Projection: $\pi_i(D_1 \times D_2 \times \ldots \times D_n) \to D_i$
5. If the domain is a cpo, what is the lub of all chains? (Not done in lecture)

Note $D \to E$ denotes a continuous function from D to E, i.e. $\{(d_1, e_1), (d_2, e_2), \ldots\}$ where each element of D occurs once and only once.

2.2 Function Domain

All functions $f_i : D \rightarrow E$ form a domain.

1. Elements: f_i
2. Ordering: $f_i \sqsubseteq f_j \overset{\text{def}}{\Leftrightarrow} \forall d \in D. f_i(d) \sqsubseteq f_j(d)$.
3. $\bot$: $\bot = \lambda d : D.\bot_E$
4. Operations (monotonic and continuous):
   - Application: $((D \rightarrow E) \times D) \rightarrow E$
5. If the domain is a cpo, what is the lub of all chains?

$$\bigsqcup_n f_n = \lambda d : D. \bigsqcup_n f_n(d) \quad where f_0 \sqsubseteq f_1 \sqsubseteq f_2 \ldots$$

2.3 Sum Domain

We would like to be able to merge domains together. This is accomplished by creating a *sum domain*. Given a set of domains $D_1 \ldots D_n$, $D_1 + D_2 + \ldots + D_n$ is a sum domain.

There is a small complication here, in that we would like to be able to determine which domain some element d_k is from. This is useful for cases where the sum domain has, for some i and j, $D_i = D_j$. Thus we tag each element with the index i of its original domain, e.g. $in_i(d_k)$ means that d_k comes from D_i.

1. Elements: $\bigcup_i \{in_i(d) | d \in D_i\}$
2. Ordering: $in_i(d) \sqsubseteq in_j(d') \overset{\text{def}}{\Leftrightarrow} i = j \wedge d \sqsubseteq_{D_i} d'$.
3. $\bot$: None.
4. Operations (monotonic and continuous):
   - Case: case e of

$$\begin{array}{ll} & in_1(x_1) \Rightarrow e_1 \\ | & in_2(x_2) \Rightarrow e_2 \\ & \vdots \\ | & in_n(x_n) \Rightarrow e_n \end{array}$$

     where when one case is actually selected, e_i becomes $e_i[d/x_i]$.
   - Constructor: $in_i(d)$
5. If the domain is a cpo, what is the lub of all chains?

$$\bigsqcup_n in_i(d_n) = in_i(\bigsqcup_n d_n)$$

   Note the fact that the d_n are forming a chain implies that they must be in the same domain (for the ordering to exist), so $\bigsqcup_n d_n$ is defined.

For example: $\mathcal{T} = \{true, false\}$ is a flat cpo. We can simulate this by constructing $U + U$ so that $true = in_1(\top)$ and $false = in_2(\top)$. Then

if b then e_1 **else** e_2 =

case b of

$$\begin{array}{ll} & in_1(x) \Rightarrow e_1 \\ | & in_2(x) \Rightarrow e_2 \end{array}$$

3 Continuity

Now we can use these to build expressions; since all of our operations are 'safe' (monotonic and continuous), any result will also be.

For *fix*(Γ) to make sense, Γ must be continuous. $\Gamma = \lambda f.e$ is continuous if e is continuous in its variables. For $\lambda x.e$, $x \notin FV[\![e]\!]$ implies that the expression $\lambda x.e$ is a constant function and therefore necessarily continuous. However, the case where we construct $\lambda x.e$, $x \in FV[\![e]\!]$, needs to be considered for each possible kind of expression e.

Lemma: If a function is continuous with respect to each variable independently, then it is continuous with respect to all of them. [See Winskel for proof]

Now, we can show that all expressions in the meta-language are continuous in their variables by structural induction. We'll just do the base case here of an expression that is a single variable. Consider $e \equiv x$. Showing that this expression is continuous in its variables reduces to showing that $\lambda x.x$ is continuous, since $FV[\![x]\!] = \{x\}$.

- Monotonicity: $d_1 \sqsubseteq d_2 \Rightarrow ((\lambda x.x)\ d_1) \sqsubseteq ((\lambda x.x)\ d_2)$ is clearly true.
- Continuity: $\lambda x.x\ (\bigsqcup_n d_n) = \bigsqcup_n (\lambda x.x)\ d_n$ is also clearly true.

4 Our meta-language

To summarize, we allow the following sorts of expressions, because they are all continuous in their variables and we can therefore take fixed points over expressions that mention them:

- Variables
- Constants
- Tuple (record) constructors
- Projection operators
- Application of a continuous function to an expression
- λ-abstractions ($\lambda x.e$)
- Composition ($f \circ g$)
- Let expressions: **let** $x \leftarrow e_1$ **in** e_2, i.e. $((\lambda x.e_2)\ e_1)$
- *case* and in_i
- *fix*

Note that even the *fix* operator is continuous.

Claim: *fix* is continuous.

Proof: *fix*$(f) = x$ s.t. $f(x) = x$. *fix* : $(D \to D) \to D$. More precisely,

$$\mathit{fix} = \lambda f. \bigsqcup_n f^n(\bot)$$

which is monotonic and continuous, by the following reasoning. The above expression $= \bigsqcup_n \lambda f.f^n(\bot)$ (see Homework 3). Now if f is monotonic and continuous then $\lambda f.f^n(\bot)$ is a pointed cpo (the cpo of continuous functions), so its lub exists and, because it is also in the cpo of continous functions, is continuous as well. Thus, our definition of **while**, which uses *fix*(Γ), is a 'safe' construction too.