CS611 Lecture 13 **Domain Constructions II** 26 September 2003

Scribe: David Martin Lecturer: Radu Rugina

1 Operations on Function Spaces

Previously, we saw that if $(D, \sqsubseteq), (E, \sqsubseteq)$ are c.p.o.'s, then $([D \to E], \sqsubseteq)$ is a c.p.o.

We will now define operations on (continuous) function spaces, with which we can build complex continuous functions out of simpler ones.

This will provide us with a powerful method of showing that complex functions are continuous. If we can show that a given function can be constructed from a sequence of such operations, then we can immediately infer its continuity.

1.1 Apply

Consider two c.p.o.'s $(D, \sqsubseteq)$ and $(E, \sqsubseteq)$. We define *apply* to be an operation that takes a continuous function and applies it to an element of its domain as follows:

$$\mathit{apply} : ([D \to E] \times D) \to E$$

$$\mathit{apply}(f, d) = f(d)$$

Then *apply* is continuous. (Proof left as an exercise)

1.2 Curry

We define *curry* to be an operation that converts a continuous function on pairs into a high-order function on the first element of the pair as follows:

$$\mathit{curry} : [(D \times E) \to F] \to [D \to [E \to F]]$$

$$\mathit{curry}(g) = \lambda d.(\lambda e.g(d, e))$$

Then, for every continuous function g and any $e \in E$, $\mathit{curry}(g)(e)$ and $\mathit{curry}(g)$ are continous (Proof left as an exercise). Therefore the range of *curry* is correctly defined.

Moreover, *curry* itself, as a function on $[(D \times E) \to F]$, is continuous. (The proof of this is left as an exercise.)

1.3 Composition

We define *compose* to be an operation that composes continuous functions as follows:

$$\mathit{compose} : ([D \to E] \times [E \to F]) \to [D \to F]$$

$$\mathit{compose}(f, g) = f \circ g$$

Recall that if $f : E \to F$ and $g : D \to E$ are continuous, then $f \circ g : D \to F$ is continuous. (We saw this in a previous lecture.) So $\mathit{compose}(f, g)$, as a function on D, is continuous.

Moreover, *compose* itself, as a function on $[D \to E] \times [E \to F]$, is continuous. (The proof of this is left as an exercise.)

1.4 Fixed points

We define *fix* to be an operation that returns the least fixed point of a continuous function from D to D (where $(D, \sqsubseteq, \perp)$ is a c.p.o. with bottom) as follows:

$$\begin{gathered} \mathit{fix} : [D \to D] \to D \\ \mathit{fix}(f) = \text{the least fixed point of } f \end{gathered}$$

The higher-order function *fix* is well-defined because, by the Fixed Point Theorem, every $f \in [D \to D]$ has a least fixed point. And if it exists, the least fixed point is always unique.

Then *fix* is continous. (The proof of this is a homework question.)

1.5 Lifting

Lifting refers to the operation that adds a bottom value to a domain. We extend a c.p.o. $(D, \sqsubseteq)$ to a c.p.o with bottom $(D_\perp, \sqsubseteq, \perp)$ (the lifted c.p.o.) as follows:

$$\begin{gathered} D_\perp = \{\lfloor d \rfloor : d \in D\} \cup \{\perp\} \\ \sqsubseteq = \{(\lfloor d_1 \rfloor, \lfloor d_2 \rfloor) : d_1, d_2 \in D \wedge d_1 \sqsubseteq d_2\} \cup \{(\perp, \lfloor d \rfloor) : d \in D\} \end{gathered}$$

This can be thought of as renaming all the elements of D and adding a $\perp$ symbol. The purpose of renaming is to avoid conflict with a pre-existing $\perp$ symbol in D. (Notice that $\lfloor \perp_D \rfloor \neq \perp_{D_\perp} = \perp$.)

We define a lift operation that extends the domain of continuous function $f : D \to E$ (where E is a c.p.o. with bottom $\perp_E$) to a c.p.o. with bottom as follows:

$$f^* : D_\perp \to E$$

$$f^*(d) = \begin{cases} f(d') & \text{if } d = \lfloor d' \rfloor, d' \in D \\ \perp_E & \text{if } d = \perp \end{cases}$$

(Here, $\lambda f.f^* : [D \to E] \to [D_\perp \to E]$ is the lift operation.)

Then f^*, as a function on $D_\perp$, is continuous. (Proof left as an exercise)

Moreover, $\lambda f.f^*$ itself, as a function on $[D \to E]$, is continuous. (The proof of this is in the textbook.)

1.6 Combining

First we will define the **disjoint sum** of two c.p.o's as follows:

Let $(D_1, \sqsubseteq), (D_2, \sqsubseteq)$ be c.p.o.'s.

Let $\begin{array}{l} \mathrm{in}_1 : D_1 \to D_1 + D_2, \\ \mathrm{in}_2 : D_2 \to D_1 + D_2 \end{array}$ be injective functions with disjoint ranges. I.e.

$$\forall x \in D_1 . \forall y \in D_2 .\ \mathrm{in}_1(x) \neq \mathrm{in}_2(y)$$

Then $(D_1 + D_2, \sqsubseteq)$ is a c.p.o., where

$$\sqsubseteq = \{(\mathrm{in}_1(d), \mathrm{in}_1(d')) : d, d' \in D_1 \wedge d \sqsubseteq d'\} \cup \{(\mathrm{in}_2(d), \mathrm{in}_2(d')) : d, d' \in D_2 \wedge d \sqsubseteq d'\}$$

(In the above definition, the domain of the various $\sqsubseteq$'s should be inferred from context.)

We shall extend the definition of disjoint sums to be over several c.p.o.'s rather than just two.

We now define a combine operation that *combines* continuous functions $f_1 : E_1 \to D, \cdots, f_k : E_k \to D$ into a single continuous function over the disjoint sum of the individual domains as follows:

$$\begin{gathered} [f_1, \cdots, f_k] : (E_1 + \cdots + E_k) \to D \\ [f_1, \cdots, f_k](\mathrm{in}_i(e)) = f_i(e), \forall e \in E_i \end{gathered}$$

Then the combined function $[f_1, \cdots, f_k]$ is continuous. (The proof of this is left as an exercise).

1.7 Conditionals

Let $U = (\{u\}, =)$ be a c.p.o. (U is said to have a unit domain, which is one that contains a single element.)

Let $T = U + U$, where $\begin{cases} \text{in}_1(u) = \mathit{true} \\ \text{in}_2(u) = \mathit{false} \end{cases}$

Then $T = \{\mathit{true}, \mathit{false}\}$.

We now define *cond* to be an operation that uses the value of the first argument to choose between the second and third arguments as follows:

$$\mathit{cond} : (T, E, E) \to E$$
$$\mathit{cond}(t, e_1, e_2) = [\lambda x_1.e_1, \lambda x_2.e_2](t)$$

where

$$\lambda x_1.e_1 : U \to E$$
$$\lambda x_2.e_2 : U \to E$$

Then *cond* is continuous. (This is immediate from the definition and because combining is continuous).

1.8 Case Constructs

We define *case* to be an operation which uses the domain from which the first argument originates chooses between the second and third arguments as follows:

$$\mathit{case} : ((D_1 + D_2) \times E \times E) \to E$$
$$\begin{array}{r} \mathit{case}\ e \text{ of } \text{in}_1(x_1).e_1 \\ |\text{in}_2(x_2).e_2 \end{array} \equiv [\lambda x_1.e_1, \lambda x_2.e_2](e)$$

If the functions $\lambda x_1.e_1$ and $\lambda x_2.e_2$ are continuous, then *case* is continuous.

2 Denotational Semantics of while

We now return to the goal of showing that our definition of denotational semantics for **while** was well-defined.

Recall:

$$\mathcal{C}[\![\textbf{while}\ b\ \textbf{do}\ c]\!] = \mathit{fix}(\Gamma)$$

where

$$\Gamma : (\Sigma \to \Sigma_\perp) \to (\Sigma \to \Sigma_\perp)$$
$$\Gamma(\gamma) = \lambda\sigma.\text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (\text{if } \mathcal{C}[\![c]\!]\sigma = \perp \text{ then } \perp \text{ else } \gamma(\mathcal{C}[\![c]\!]\sigma)) \text{ else } \sigma$$

We can now reduce Γ to operations on function spaces as follows:

$$\begin{aligned} \Gamma(\gamma) &= \lambda\sigma.\text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (\text{if } \mathcal{C}[\![c]\!]\sigma = \perp \text{ then } \perp \text{ else } \gamma(\mathcal{C}[\![c]\!]\sigma)) \text{ else } \sigma \\ &= \lambda\sigma.\text{if } \mathcal{B}[\![b]\!]\sigma \text{ then } (\gamma^* \circ \mathcal{C}[\![c]\!])\sigma \text{ else } \sigma \\ &= \lambda\sigma.\mathit{cond}(\mathcal{B}[\![b]\!]\sigma, (\gamma^* \circ \mathcal{C}[\![c]\!])\sigma, \sigma) \\ &= (\mathit{cond} \circ \langle \mathcal{B}[\![b]\!], \gamma^* \circ \mathcal{C}[\![c]\!], \text{id}\rangle)(\sigma) \end{aligned}$$

Thus $\Gamma = \mathit{cond} \circ \langle \mathcal{B}[\![b]\!], \gamma^* \circ \mathcal{C}[\![c]\!], \text{id}\rangle$ (where $\text{id} : \Sigma \to \Sigma_\perp$ is the identity function).

So, assuming that $\mathcal{B}[\![b]\!], \mathcal{C}[\![c]\!]$ are continuous, we immediately have $\Gamma(\gamma)$ continuous, for any continuous γ.

Note, however, that this does not immediately imply that Γ itself is continuous. (It will eventually turn out that Γ is indeed continuous, but we will have to do more work to see this.)