LECTURE 3: CONSTRUCTIONS ON CPO'S

In lecture 2 we proposed that

* semantic domains are cpo's,
* computable functions are continuous,

and showed how the "CPO fixpoint theorem" allows us to give semantics to recursively defined programs.


In this lecture we introduce natural constructions on cpo's, corresponding to the familiar type-forming operators

\( \times \) (product), \( \rightarrow \) (function space), and \( + \) (sum).

Hence the fixpoint approach of lecture 2 applies uniformly to programs that manipulate compound values.
Recall (lecture 2): if \( A \) and \( B \) are cpo's, then
\[
A \times B = \{ (a,b) \mid a \in A \text{ and } b \in B \}
\]
is a cpo under the ordering
\[
(a,b) \leq (a',b') \text{ iff } a \leq a' \text{ and } b \leq b'
\]
with the cpo structure given by
\[
\bot_{A \times B} = (\bot_A, \bot_B)
\]
\[
\bigcup X = \left( \bigcup \pi_0 X, \bigcup \pi_2 X \right) \quad \text{(for directed } X \in A \times B)\]

Now we look at products in more detail...
PRIMITIVE "FUNCTIONS ON PRODUCTS"

1. Destructing pairs

\[ \pi_0 : A \times B \to A \]  
"project left"

\[ \pi_1 : A \times B \to B \]  
"project right"

are defined by

\[ \pi_0(a, b) = a \quad \text{and} \quad \pi_1(a, b) = b. \]

2. Constructing pairs

If \( f : A \to B \) and \( g : A \to C \) then

\[ \langle f, g \rangle : A \to B \times C \]  
"split"

is defined by

\[ \langle f, g \rangle \alpha = (fa, ga). \]

Note: "picturing" these primitives can be helpful.

\[ \pi_0 \quad \pi_1 \quad \langle f, g \rangle \]
Fact: $\Pi_0$ and $\Pi_1$ are continuous.

Fact: if $f$ and $g$ are continuous, so is $\langle f, g \rangle$.

Proof (in two parts):

1. $\langle f, g \rangle$ is monotonic.

Let $x \leq y$ in $A$. Then

$$\langle f, g \rangle x = (fx, gx)$$

$$\leq (fy, gy)$$

$$= \langle f, g \rangle y \quad \square$$

2. $\langle f, g \rangle$ preserves lubs of directed sets.

Let $X \subseteq A$ be directed. Then

$$\langle f, g \rangle \bigvee X = (f \bigvee X, g \bigvee X)$$

$$= (\bigvee fX, \bigvee gX)$$

$$= \left\{ \sum (fx, gx) \mid x \in X \right\}$$

$$= \bigvee \langle f, g \rangle X \quad \square$$
DERIVED "FUNCTIONS ON PRODUCTS"

For example ...

\[ \text{swap} : A \times B \rightarrow B \times A \]

\[ \text{swap} = \langle \pi_2, \pi_0 \rangle \]

\[ f \times g : A \times C \rightarrow B \times D \quad (\text{for } f : A \rightarrow B \text{ and } g : C \rightarrow D) \]

\[ f \times g = \langle f \circ \pi_0, g \circ \pi_2 \rangle \]

\textbf{Note:} \ x \ is \ now \ overloaded, \ e.g. \ A \times C \ is \ a \ cpo, \ while \ f \times g \ is \ a \ continuous \ function
UNIVERSALITY

\[ \pi_0 \circ \langle f, g \rangle = f \]
\[ \pi_1 \circ \langle f, g \rangle = g \]
\[ \langle \pi_0 \circ h, \pi_1 \circ h \rangle = h \]

Fact: these are together equivalent to

\[ (\pi_0 \circ h = f \text{ and } \pi_1 \circ h = g) \iff h = \langle f, g \rangle. \]

This is the universal property or unique extension property (UEP) for \( \langle -,- \rangle \) and says that \( \langle f, g \rangle \) is the unique \( h : A \to B \times C \) such that \( \pi_0 \circ h = f \text{ and } \pi_1 \circ h = g. \)

Note: the UEP rule is easy to remember as a "commuting diagram":

\[ \begin{array}{c}
A \\
\downarrow f \\
B \\
\pi_0 \\
\downarrow \langle f, g \rangle \\
B \times C \\
\pi_1 \\
\downarrow g \\
C \\
\end{array} \]
DERIVED PROPERTIES

Fact: the UEP rule for \(<f,g>\) can be used to prove all the familiar properties of products, e.g.

\[
\Pi_0 \circ (f \times g) = f \circ \Pi_0 \\
\Pi_1 \circ (f \times g) = g \circ \Pi_1 \\
(f \times g) \circ <h,i> = <f \circ h, g \circ i> \\
<f,g> \circ h = <f \circ h, g \circ h> \\
id_A \times id_B = id_{A \times B} \hspace{1cm} (\star) \\
(f \circ g) \times (h \circ i) = (f \times h) \circ (g \times i) \hspace{1cm} (**)
\]

(thinking "in pictures" illuminates these equations)

A categorical aside:

* Categorically, (\star) and (**) together with the typing rule for X express that X is a bifunctor on the category CPO with cpo's as objects and continuous functions as arrows.

* The UEP rule expresses that LPC has "binary products".
SOME USEFUL ISO’S ON PRODUCTS

Sets $X$ and $Y$ are isomorphic ($X \cong Y$) iff there are functions $f: X \to Y$ and $g: Y \to X$ such that

$$g \circ f = \text{id}_X \text{ and } f \circ g = \text{id}_Y.$$  (i.e. $f = g^{-1}$)

($X \cong Y$ means that $X$ and $Y$ are in one-to-one correspondence.)

Fact: cpo’s form a commutative monoid “up to iso”
under $\times$ and $1$, i.e. for all cpo’s $A, B, C$,

$$A \times 1 \cong A,$$
$$A \times B \cong B \times A,$$
$$A \times (B \times C) \cong (A \times B) \times C.$$

Fact: pairs of continuous functions $A \to B$ and $A \to C$ are in 1-1 correspondence with continuous functions $A \to B \times C$,

$$[A \to B] \times [A \to C] \cong [A \to (B \times C)].$$

Proof: take $f = (-,-)$ and $g = \lambda h. (\pi_0 \circ h, \pi_2 \circ h)$ then the iso follows immediately from the UEP rule for $(-,-)$.

Categorically: the existence of an operator $\times$ on cpo’s satisfying ($\ast$) expresses that CPO has binary products.
FUNCTIONS

Theorem: if $A$ and $B$ are cpo's then

$$A ightarrow B = \{ f : A \rightarrow B \mid f \text{ is continuous} \}$$

is a cpo under the ordering

$$f \leq g \text{ iff } \forall a \in A, fa \leq ga.$$  

Intuition: "the information content of a function is increased by increasing the information content of the result value for any (or many) argument values."

Proof: We must show two things.

1. $A \rightarrow B$ has a least element

   $\bot_{A \rightarrow B}$ is the "constant bottom" function $\lambda a. \bot.$

2. $\bigcup X$ exists for all directed $X \subseteq A \rightarrow B$

   We will show that $\bigcup X = \lambda a. \bigcup \{ fa \mid f \in X \}$...
We must show four things now:

2.1 \[ \bigcup \exists f \alpha \mid \exists f \in X^3 \text{ exists in } B \]

Let \( g a, h a \in \exists f \alpha \mid f \in X^3 \). Since \( X \) is directed, there exists a function \( k e X \) s.t \( g \leq k \) and \( h \leq k \). Now (using the defn. of \( \leq \) on functions), \( k a \in \exists f \alpha \mid f \in X^3 \) is such that \( g a \leq k a \) and \( h a \leq k a \). Hence \( \exists f \alpha \mid f \in X^3 \) is directed and its lub exists since \( B \) is a cpo.

\[ \begin{align*}
\varphi = \lambda a. \bigcup \exists f \alpha \mid f \in X^3 \text{ is continuous}.
\end{align*} \]

Let \( Y \subseteq A \) be directed. Then

\[ \begin{align*}
\varphi(\downarrow Y) &= \bigcup \exists f(\downarrow Y) \mid f \in X^3 \quad [\text{defn } \varphi] \\
&= \bigcup \exists \exists f \mu \mid \mu \in Y \mid f \in X^3 \quad [\varphi \text{ is continuous}] \\
&= \bigcup \exists \exists f \mu \mid f \in X^3 \mid \mu \in Y \quad [\text{lubs}] \\
&= \bigcup \exists \varphi(\mu) \mid \mu \in Y \quad [\text{defn } \varphi]
\end{align*} \]

(Note: monotonicity of \( \varphi \) is easy too.)
2.3 \( \Phi \) is an upper bound for \( X \subseteq A \rightarrow B \).

Let \( g \in X \). Then for any \( a \in A \) we have
\[ g \epsilon \in \exists f \in X \} \].
Now (by 2.1 and lubs) we have
\[ g \epsilon \in \bigcup \{ f \epsilon \in X \} \],
which using the defn. for \( \Phi \) reads \( g \epsilon \in \Phi a \). Hence
(using \( \subseteq \) on functions) we have \( g \subseteq \Phi \). \( \Box \)

2.4 \( \Phi \) is the least upper bound for \( X \)

Let \( g : A \rightarrow B \) be an upper bound for \( X \subseteq A \rightarrow B \).
Then (using \( \subseteq \) on functions) \( g \epsilon \) is an upper bound
for \( \exists f \epsilon \in X \} \), for all \( a \epsilon \in A \). Now (using 2.1
and lubs) we have \( \bigcup \{ f \epsilon \in X \} \subseteq g \epsilon a \), which
using the defn. for \( \Phi \) reads \( \Phi a \subseteq g a \). Hence
(using \( \subseteq \) on functions) we have \( \Phi \subseteq g \). \( \Box \)

\[ \ast \]
EXAMPLE \((B_1 \to \mathbb{B}_1)\)

Note: at each step up in this cpo we move one arrow-tip one step up in the cpo \(\mathbb{B}_1\).
PRIMITIVE "FUNCTIONS ON FUNCTIONS"

1) "Destructing" functions

apply : \((A \to B) \times A \to B\)

apply \((f, a)\) = \(f(a)\)

Fact: apply is continuous

2) "Constructing" functions

If \(f : A \times B \to C\) then

\(\text{curry}(f) : A \to (B \to C)\)

is defined by

\(\text{curry}(f) a = \lambda b. f(a, b)\)

Fact: if \(f\) is continuous, so is \(\text{curry}(f)\).

Intuition: \(\text{curry}(f)\) has the same behaviour as \(f\) except it takes its arguments "one at a time" rather than together.
FACTS ABOUT apply and curry

* They can be used to define \(\to\) on functions:

If \(f : A \to B\) and \(g : C \to D\) then \(f \to g : (B \to C) \to (A \to D)\)

is defined by \(f \to g = \text{curry } (g \circ \text{apply } \circ \text{id } \times f)\).

(operationally: \(f \to g = \lambda h. g \circ h \circ f\).)

* curry satisfies a universal property:

\[
\begin{array}{ccc}
A \times B & \overset{f}{\longrightarrow} & C \\
\downarrow \text{curry } (f) \times \text{id } B & \nearrow \text{apply} \\
(B \to C) \times B & & \\
\end{array}
\]

* the UEP can be used to prove properties of \(\to\), e.g.

\[
id_A \to \ id_B = \ id_{A \to B}
\]

\[
(f \circ g) \to (h \circ i) = (g \to h) \circ (h \to i)
\]

* the UEP establishes an important iso:

\[
[A \times B \to C] \cong [A \to (B \to C)]
\]

Categorically: \(\to\) is a functor \(CPC^{op} \times CPC \to CPC\),

CPC has exponentials.
**Sums**

**Theorem:** if $A$ and $B$ are cpo's then

$$A + B = \{(0,a) \mid a \in A\} \cup \{(1,b) \mid b \in B\} \cup \bot_{A+B}$$

is a cpo under the ordering

$$x \leq y \text{ iff } \begin{cases} 
\text{true} & \text{if } x = \bot_{A+B}, \\
\text{a} \leq \text{a'} & \text{if } x = (0,a) \text{ and } y = (0,a'), \\
\text{b} \leq \text{b'} & \text{if } x = (1,b) \text{ and } y = (1,b'), \\
\text{false} & \text{otherwise.}
\end{cases}$$

**Intuition:** the ordering on $A+B$ is just as in $A$ and $B$ separately, except that $\bot_{A+B}$ is less than anything.

**Note:** the cpo $A+B$ can be pictured as

![Diagram of $A+B$]
FUNCTIONS ON SUMS

1. Constructing sums

\[ \text{inl : } A \to A + B \quad \text{inl} \ a = (0, a) \quad \text{"inject left"} \]
\[ \text{inr : } B \to A + B \quad \text{inr} \ b = (1, b) \quad \text{"inject right"} \]

2. Destructing sums

If \( f : A \to C \) and \( g : B \to C \) then

\[ [f, g] : A + B \to C \quad \text{"case"} \]

is defined by

\[ [f, g] \ x = \begin{cases} \text{fa} & \text{if } x = (0, a), \\ \text{gb} & \text{if } x = (1, b), \\ \bot_c & \text{if } x = \bot_{A+B}. \end{cases} \]

3. Defining \( + \) on functions

If \( f : A \to B \) and \( g : C \to D \) then

\( f + g : A + C \to B + D \)

is defined by

\( f + g = [\text{inl} \circ f, \text{inr} \circ g]. \)
FACTS ABOUT SUMS

* \([f,g] \circ \text{inl} = f \) and \([f,g] \circ \text{inr} = g\), i.e.

\[
\begin{array}{c}
\text{C} \\
\downarrow^f \\
\text{A} \\
\downarrow^\text{inl} \quad [f,g] \\n\downarrow \quad \uparrow \text{inr} \\
\text{A+B} \quad \text{B}
\end{array}
\]

* **BUT** in general, \([f,g]\) is only weakly universal:

\((h \circ \text{inl} = f \) and \(h \circ \text{inr} = g\) iff \([f,g] \in h\).

For example,

\[
\begin{array}{c}
\text{2} \\
\downarrow^{\lambda x.T} \\
1 \\
\downarrow^\text{inl} \\
1 + 1 \\
\downarrow^\text{inr} \\
1
\end{array}
\]

\[ h = \lambda x. T \text{ makes this commute, but } h \notin [\lambda x. T, \lambda x. T] \]

* **... hence** many “expected” iso's do not hold; e.g.

\[ A + (B+C) \neq (A+B) + C \]

\[ [A \to C] \times [B \to C] \neq [A+B \to C] \]

**Categorically:** + is a bifunctor on CPO, but CPO only has local binary sums.
When giving semantics to programs in a call-by-name (or lazy) language, $\rightarrow$, $\times$ and $+$ are just what we want:

* functions can be non-strict, i.e. $\bot \rightarrow \bot$ is OK;
* pairing is non-strict, e.g. $(\text{false}, \bot) \in \text{IB}_L \times \text{IB}_L \neq \bot$;
* constructors are non-strict, e.g. $(0, \bot) \in \text{IB}_L + \text{IB}_L \neq \bot$.

To properly capture strictness properties (e.g. in a call-by-value language) we also need strict versions of $\rightarrow$, $\times$ and $+$.

**Theorem:** if $A$ and $B$ are cpos then so are

$$A \rightarrow B = \exists f : A \rightarrow B \mid f(\bot) = \bot,$$

$$A \otimes B = \exists (a, b) : A \times B \mid a \neq \bot \text{ and } b \neq \bot \mid \exists \perp A \otimes B,$$

$$A + B = \exists (t, x) : A + B \mid x \neq \perp \mid \exists \perp A \otimes B.$$

(the orderings are the evident ones.)
Note: $\oplus$ and $\otimes$ are often called "smash" sum and product.

(two $\bot$'s are smashed together to give a single $\bot$)

Fact: the strict cpo constructions satisfy

$$\left[ A \rightarrow B \right] \times \left[ A \rightarrow C \right] \cong \left[ A \rightarrow B \otimes C \right],$$

$$\left[ A \otimes B \rightarrow C \right] \cong \left[ A \rightarrow (B \rightarrow C) \right],$$

$$\left[ A \rightarrow C \right] \times \left[ B \rightarrow C \right] \cong \left[ A \oplus B \rightarrow C \right].$$

[[Categorically: the category CPO$_\bot$ of cpos and strict continuous functions has products, exponentials, and sums.]]

Fact: the lazy/strict constructions are related in useful ways,

$$A \rightarrow B \cong A \bot \rightarrow B$$

$$A + B \cong A \bot \oplus B \bot \quad \left( \Uparrow \Downarrow \cong \Uparrow \Uparrow \Downarrow \right)$$

$$(A \times B)_\bot \cong A \bot \otimes B \bot$$

where "lifting" $(-)_\bot$ adds a new $\bot$ to a cpo, i.e. $\triangle A \rightarrow \triangle A$
EXERCISES

1. Show using the UEP rule that \( (f, g) \circ h = (f \circ h, g \circ h) \).

2. Draw the cpo \( 3 \to 3 \).
   (see exercise 1.2 and slide 3.11.)

3. Express \( \lambda (a, (b, c)) . ((a, b), c) \) using combiners.

4. What is the common name for apply \( \circ \) fixid?

5. Show that apply \( \circ \) is continuous.

(You may omit the proof of monotonicity, and make use of the following useful result:

**Fact:** a function \( f : A \times B \to C \) is continuous iff it is continuous in each argument separately, i.e.

\[
\forall a \in A. f(a, -) : B \to C \text{ is continuous; and } \\
\forall b \in B. f(-, b) : A \to C \text{ is continuous.}
\] )