LECTURE 5: RECURSIVELY DEFINED DOMAINS

In lecture 2 we showed how to give semantics to recursively defined values and programs:

Thm: every continuous \( f: A \to A \) on a cpo \( A \) has a least fixpoint, given by the lub of the \( w \)-chain

\[
\bot \leq f \bot \leq ff \bot \leq fff \bot \leq \ldots.
\]

In this lecture we generalise this theorem from values (elements of cpo's) to semantic domains (cpo's themselves):

Thm: every cocontinuous functor \( F: \text{CPO} \to \text{CPO} \) has a least fixpoint, given by the colimit of the \( w \)-chain

\[
\begin{array}{c}
1 \xrightarrow{\lambda x. \bot} F1 \xrightarrow{F(\lambda x. \bot)} FF1 \xrightarrow{FF(\lambda x. \bot)} \ldots
\end{array}
\]

We can then give semantics to recursively defined domains.
① Semantics of recursively defined types

example: the Haskell datatype definition

data List = Nil | Cons (Int, List)

can be given a semantics using a cpo \( L \) s.t.

\[
L = \text{Nil} + (\mathbb{Z} \times L).
\]

② Semantics of higher-order programming languages

example: the untyped \( \lambda \)-calculus with syntax

\[
\text{exp ::= var } \mid \text{ exp exp } \mid \lambda \text{var.exp}
\]

variables application abstraction

can be given a semantics using a cpo \( D \) s.t.

\[
D = D \rightarrow D.
\]
SEMANTICS OF RECURSIVE DOMAINS

Idea (Scott 1970): generalise the fixpoint approach (lecture 2) from values (elements of cpos) to domains (cpos).

**Step 1:** express recursive domains as fixpoints

example: \( L = \bot + (\mathbb{Z}_1 \times L) \)

\[ \Downarrow \]

\[ L = F(L) \]

where \( F \) is a (non-recursive) mapping on cpos,

\[ F(A) \overset{\text{def}}{=} \bot + (\mathbb{Z}_1 \times A). \]

**Step 2:** generalise the CPO fixpoint theorem

**Thm:** every continuous \( f : A \to A \) on a cpo \( A \) has a least fixpoint, given by the lob of the \( \omega \)-chain

\[ \bot \in f \bot \in f f \bot \in f f f \bot \ldots \]

The generalisation is "Scott's inverse limit construction."
GENERALISING FROM VALUES TO CPO'S

approximation ($x \leq y$)

A cpo $A$ "approximates" a cpo $B$ iff there is a continuous function $f : A \to B$.

Note: we might expect to require that $f$ be injective, but this is not technically necessary.

least element ($\bot \leq x$)

The one-point cpo $1$ = $\{\bot\}$, together with the function $\lambda x.1 : 1 \to A$ for each cpo $A$,

\[ 1 \xrightarrow{\lambda x.1} A. \]

Aside: the category CPO has no initial object; $1$ is terminal, but also serves as a "pseudo" initial object.

ω-chain

A family $\{A_i\}_{i \in \mathbb{N}_3}$ of cpos, together with a family $\{S_i : A_i \to A_{i+1}\}_{i \in \mathbb{N}_3}$ of continuous funs:

\[ A_0 \xrightarrow{S_0} A_1 \xrightarrow{S_1} A_2 \xrightarrow{S_2} A_3 \xrightarrow{S_2} \ldots. \]
**Upper bound** (of an ω-chain)

A cpo $A$ together with a family $\{g_i : A_i \rightarrow A \mid i \in \mathbb{N}\}$ of continuous $f$s, such that the following diagram commutes:

\[
\begin{array}{cccccc}
  & & & & & \\
  & & & & & \\
 & & & & & \\
A_0 \overset{g_0}{\rightarrow} A_1 \overset{g_1}{\rightarrow} A_2 \overset{g_2}{\rightarrow} A_3 \cdots \\
\end{array}
\]

i.e. $g_i = g_{i+1} \circ f_i$ for all $i \in \mathbb{N}$.

**Least upper bound**

The colimit (or inverse limit) of an ω-chain is an upper bound $(A, \{g_i\})$ s.t for any other upper bound $(B, \{h_i\})$ there is a unique continuous $k : A \rightarrow B$ s.t.

\[
\begin{array}{cccccc}
  & & & & & \\
  & & & & & \\
 & & & & & \\
A_0 \overset{g_0}{\rightarrow} A_1 \overset{g_1}{\rightarrow} A_2 \overset{g_2}{\rightarrow} A_3 \cdots \\
\end{array}
\]

i.e. $h_i = g_i \circ k$ for all $i \in \mathbb{N}$.
Note: if it exists, the colimit is "unique up to iso": if \((A', \xi_{g';3})\) is also a colimit of the chain, then \(A \cong A'\).

**Monotonic Function**

A functor (on CPO) is a mapping \(F\) on cpo's, together with a mapping \(F\) on continuous functions, s.t.

1. If \(f : A \to B\) then \(Ff : FA \to FB\), "preserves types"
2. \(F(id_A : A \to A) = id_{FA} : FA \to FA\), "preserves id"
3. \(F(f \circ g) = Ff \circ Fg\), "dist. over \(\circ\)"

**Continuous Function**

A functor \(F\) is ccontinuous iff it preserves colimits of \(\omega\)-chains, i.e. if \((A_i, \xi_{g;i})\) is a colimit of

\[
A_\omega \xrightarrow{\delta_0} A_2 \xrightarrow{\delta_2} A_3 \xrightarrow{\delta_3} A_4 \cdots
\]

then \((FA_i, \xi_{Fg;i})\) is a colimit of

\[
FA_\omega \xrightarrow{F\delta_0} FA_2 \xrightarrow{F\delta_2} FA_3 \xrightarrow{F\delta_3} FA_4 \cdots
\]
A fixpoint of a mapping \( F \) on cpo's is a cpo \( A \) such that \( FA \) is iso to \( A \), i.e. \( FA \cong A \).

**Note**: fixpoints as iso's is inherent in Scott's approach to recursive domain equations, but there are other approaches where fixpoints are equalities.

**Fixpoint theorem**

Theorem: every cocontinuous functor \( F \) has a least fixpoint, given by the colimit of the \( w \)-chain

\[
\begin{array}{c}
1 \xrightarrow{\lambda x.1} F1 \xrightarrow{F(\lambda x.1)} FF1 \xrightarrow{FF(\lambda x.1)} FFF1 \xrightarrow{FFF(\lambda x.1)} \ldots
\end{array}
\]

**Fact**: if \( F \) is a mapping on cpo's built up from basic cpo's using the ops \( \times, +, \to, \otimes, \oplus, \ominus, (-) \), from lecture 3, then \( F \) extends to a cocontinuous functor.
Example (binary numbers)

Consider the Haskell datatype definition

```
data Bin = Zero | One | Empty | One Bin.
```

e.g. `One (Zero (One Empty)) :: Bin`.

The corresponding recursive domain equation is

\[ B \equiv B + 1 + B. \]

Expressed as a fixpoint, this equation reads

\[ B \equiv F(B), \text{ where } F(X) \overset{\text{def}}{=} X + 1 + X. \]

The non-recursive mapping \( F \) on cpo's extends to a mapping on continuous functions:

\[ F(f) \overset{\text{def}}{=} f + \text{id}_Y + f \left[ \frac{f : X \to Y}{Ff : FX \to FY} \right] \]

and this defn. makes \( F \) a cocontinuous functor.
Hence the recursive type \( \text{Bin} \) has a semantics as a least fixpoint of \( F \), i.e. as the colimit of

\[
\Lambda \xRightarrow{\lambda x. \bot} F \Lambda \xrightarrow{F(\lambda x. \bot)} FFF \Lambda \xrightarrow{FF(\lambda x. \bot)} FFFF \Lambda \cdots
\]

What does the chain look like?

We write the sum injections \( m_i : A_i \to A_0 + A_1 + A_2 \), where \( i \in \{0, 1, 2\} \) as \( Z(\text{ero}) \), \( E(\text{mpty}) \), and \( O(\text{ne}) \) to make the link with the Haskell defn.

The \( n \)th cpo in the chain is the type of binary numbers with at most \( n \) defined digits. The colimit of the chain is the type of finite, partial and infinite binary numbers.
SERIOUS PROBLEM

The generalised fixpoint approach breaks down with recursive domains involving function-spaces.

Fact: mappings $F$ on cpos, built using $\to$ and $\rightleftharpoons$, do not in general extend to cocontinuous functors.

Example: $F(A) \overset{\text{def}}{=} A \to \mathbb{Z}_+$ extends to continuous functions by $F(f) \overset{\text{def}}{=} \lambda g. g \circ f$, which doesn't even have the right type to be a functor!

If $f : X \to Y$ then $Ff : (Y \to \mathbb{Z}_+) \to (X \to \mathbb{Z}_+)$, i.e. $Ff : FY \to FX$ “swapped”

Technically, the problem arises because $\to$ extends to a functor that's contravariant in its first arg:

$$f : A \to B \quad g : C \to D$$

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

“swapped”

$$f \to g \overset{\text{def}}{=} \lambda h. g \circ h \circ f.$$
SOLUTION (Scott 1972, Smyth & Plotkin 1982)

Solve recursive domain equations using functors on CPO* (cpo's and retraction pairs) rather than the simpler setting of CPO (cpo's and continuous fns).

Defn: a retraction pair from a cpo \( A \) to a cpo \( B \) is a pair \((f : A \to B, g : B \to A)\) of continuous fns

\[
\begin{array}{c}
A \\
\downarrow \quad g \\
\downarrow \quad f \\
B
\end{array}
\]

such that

1. \( g \circ f = \text{id}_A \) \[\text{i.e. } \forall x \in A. \ g(f(x)) = x.\]
2. \( f \circ g = \text{id}_B \) \[\text{i.e. } \forall y \in B. \ f(g(y)) = y.\]

\( f \) is called an embedding; and \( g \) a projection.
Note: "retraction pair" is a weakening of "iso": going $A \rightarrow B \rightarrow A$ info is preserved $(g \circ f = \text{id}_A)$, but going $B \rightarrow A \rightarrow B$ info can be lost $(f \circ g \neq \text{id}_B)$.

Examples (of retraction pairs)

(i.e. there can be many retraction pairs $A \xrightarrow{s} B$.)

Examples (of non-retraction pairs)

$b \rightarrow x \rightarrow a$, but $a \neq b$. $y \rightarrow b \rightarrow z$, but $z \neq y$. 

S.N.
Facts: if $A \xrightarrow{g} B$ is a retraction pair, then

* $f$ and $g$ are **strict**: $f1 = 1$ and $g1 = 1$.

* retraction pairs compose:

\[
\begin{align*}
A & \xrightarrow{g} B \xrightarrow{i} C \\
\text{retraction pairs} & \\
A & \xrightarrow{i \circ g} C
\end{align*}
\]

* $g$ is uniquely determined by $f$, and vica-versa: e.g. if $A \xrightarrow{g'} B$ is another retraction pair then $g = g'$.

* $A$ is isomorphic to the range $\{f \times 1 \mid x \in A\} \subseteq B$ of $f$.

The last fact above motivates a new (stronger and more intuitive) notion of approximation for cpo's:

A "approx" $B$ iff there is a retraction pair $A \xleftarrow{i} B$. 
Fact: the other notions (least element, \( w \)-chain, upper bound, colimit, cocontinuous functor) naturally generalise from continuous \( \mathbb{F} \)s to retraction pairs.

The fixpoint results generalise too...

Theorem: every cocontinuous functor \( F \) on \( \text{CPO}^\mathbb{F} \) has a least fixpoint, given by the colimit of the \( w \)-chain

\[
\begin{align*}
1 & \xrightarrow{g_0} F1 \xrightarrow{g_1} FF1 \xrightarrow{g_2} FFF1 = \cdots \\
& \quad \text{where the retraction pairs are defined by}
\end{align*}
\]

\[
\begin{align*}
(5_0, g_0) & \overset{\text{def}}{=} (\lambda x.1, \lambda x.1), \\
(5_{i+1}, g_{i+1}) & \overset{\text{def}}{=} F(5_i, g_i).
\end{align*}
\]

Fact: the operators \( \times, +, \to, \& \), \( \oplus, \otimes \), \( \odot \), \( (-)_\downarrow \) on \( \text{CPO}\)'s all extend to **covariant** functors on \( \text{CPO}^\mathbb{F} \).
Example (products)

\[(f, g) \times (h, i) \overset{\text{def}}{=} (f \times h, g \times i)\]

on retraction pairs \quad on continuous funs (lecture 3)

\text{types:} \quad \begin{array}{ccc}
A & \xrightarrow{g} & B \\
\downarrow{f} & & \downarrow{h} \\
(A \times C) & \overset{g \times h}{\xrightarrow{f \times i}} & (B \times D)
\end{array}

Example (function spaces)

\[(f, g) \to (h, i) \overset{\text{def}}{=} (g \to h, f \to i)\]

\text{types:} \quad \begin{array}{ccc}
A & \xrightarrow{g} & B \\
\downarrow{f} & & \downarrow{h} \\
(A \to C) & \overset{g \circ h}{\xrightarrow{f \circ i}} & (B \to D)
\end{array}

no swapping!

Fact: If \( F \) is a mapping on cpos, built up from basic cpos using the ops \( \times, +, -, \circ, \otimes, \Theta, \cdot, (\cdot)_a \), then \( F \) extends to a cocontinuous functor on \( \text{CPO}^R \).
**WHAT DOES THE COLIMIT LOOK LIKE?**

**Defn (Scott):** the "inverse limit" of an \( w \)-chain

\[
D_0 \leftrightarrow^{g_0} D_1 \leftrightarrow^{g_1} D_2 \leftrightarrow^{g_2} D_3 = \cdots
\]

of retraction pairs is the set of \( w \)-tuples

\[
D_\infty \overset{\text{def}}{=} \{ (x_0, x_1, \ldots) \mid x_i \in D_i \text{ and } x_i = g_i(x_{i+1}) \}\.
\]

**Intuition:** read "\( x_i = g_i(x_{i+1}) \)" as "each element in the tuple is consistent with earlier elements".

**Fact:** under the evident pointwise ordering, \( D_\infty \) is a cpo. More generally, if each cpo \( D_i \) is a Scott domain (lecture 4) then so is \( D_\infty \).

**Fact:** \( D_\infty \) is an upper bound of the \( w \)-chain.

**proof (sketch):** we must construct a family

\[
\{ D_i \overset{\Theta_{i, \infty}}{\leftrightarrow} D_\infty \mid i \in \mathbb{N}\}
\]

of retraction pairs, s.t. the following diagram commutes.
part 1: $\Theta_{\infty, i} : D_\infty \rightarrow D_i$

$\Theta_{\infty, i} (x_0, x_1, ...) \overset{\text{def}}{=} x_i$

part 2: $\Theta_{i, \infty} : D_i \rightarrow D_\infty$

$\Theta_{i, \infty} x \overset{\text{def}}{=} (\Theta_{i, 0} x, \Theta_{i, 1} x, \Theta_{i, 2} x, ...)$

where the auxiliary functions $\Theta_{i, j} : D_i \rightarrow D_j$ (for $i, j \in \mathbb{N}$) are defined by composing sequences of embeddings ($s$s) or projections ($g$s), depending on whether $i < j$ or $j < i$.

Example:

$\Theta_{2, \infty} x = (\Theta_{2, 0} x, \Theta_{2, 1} x, \Theta_{2, 2} x, \Theta_{2, 3} x, \Theta_{2, 4} x, ...)$

$= (g_0(g_1 x), g_1 x, x, s_2 x, s_3 (s_2 x), ...)$

"approximations" to $x$ "equivalent" to $x$

(since $g$s are projections) (since $s$s are embeddings)
Fact: $D_\omega$ is a colimit of the $w$-chain.

- *

Example: the untyped $\lambda$-calculus (slide 5.1) can be given a semantics using a cpo $D$ such that

$$D \cong D \to D.$$  

The least such cpo, $D_\omega$, can be constructed as the colimit of the $w$-chain

$$1 \rhd \to F 1 \rhd \to FF 1 \rhd \to \cdots$$

where $F(X) \overset{\text{def}}{=} X \to X$.

But... the least solution is trivial ($D_\omega = 1$) since $1 \cong 1 \to 1$. A non-trivial solution is obtained by starting at $2 \overset{!}{=} \cdot$, rather than $1$:

$$2 \rhd \to F 2 \rhd \to FF 2 \rhd \to \cdots.$$
In this case the $\omega$-chain has the form

Finding a non-trivial model $\mathcal{D} \models \lambda \rightarrow \mathcal{D}$ of the untyped $\lambda$-calculus was Scott's original motivation for developing domain theory. The construction of such a model in 1972 is one of the most significant results in the history of theoretical computer science.
EXERCISES

1. Show that $F(X) \overset{\text{def}}{=} 1 + X$ and $F(S) \overset{\text{def}}{=} \text{id}_1 + f$
defines a functor $F$, i.e. verify the three properties
required of a functor (slide 5.5).

2. The "lazy natural numbers" can be defined as the
least solution to $X \equiv 1 + X$, i.e. as the least
fixpoint of the functor $F$ in exercise 1. Draw
the first four approximations to the least fixpoint,
writing Zero and Succ for the sum injections.

3. Repeat exercise 2 assuming
   a) Zero is strict (i.e. $\text{Zero} \ 1 = 1$)
   b) Succ is strict (i.e. $\text{Succ} \ 1 = 1$)

   What familiar types arise (up to iso) as the least
   fixpoints when these strictness conditions are added?

4. Show that $F(X) \overset{\text{def}}{=} Z_\bot \rightarrow X$ can be extended to
continuous funs such that if $f : A \rightarrow B$ then $F(f) : FA \rightarrow FB$.
[[Contrast with $F(X) \overset{\text{def}}{=} X \rightarrow Z_\bot$ from slide 5.9.]]