LECTURE 5: RECURSIVELY DEFINED DOMAINS

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

<u>Thm</u>: every <u>continuous</u> $f : A \to A$ on a cpo A has a least fixpoint, given by the <u>lub</u> of the ω-chain

$$\bot \sqsubseteq f\bot \sqsubseteq ff\bot \sqsubseteq fff\bot \sqsubseteq \cdots .$$

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

<u>Thm</u>: every <u>cocontinuous</u> <u>functor</u> $F : CPO \to CPO$ has a least fixpoint, given by the <u>colimit</u> of the ω-chain

$$\mathbb{1} \xrightarrow{\lambda x.\bot} F\mathbb{1} \xrightarrow{F(\lambda x.\bot)} FF\mathbb{1} \xrightarrow{FF(\lambda x.\bot)} FFF\mathbb{1} \cdots$$

We can then give semantics to recursively defined domains.

HOW DO RECURSIVELY DEFINED DOMAINS ARISE?

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.

$$L = \mathbb{1} + (\mathbb{Z}_\perp \times L).$$

② Semantics of higher-order programming languages

example: the untyped λ-calculus with syntax

$$\mathit{exp} ::= \underbrace{\mathit{var}}_{\text{variables}} \mid \underbrace{\mathit{exp}\ \mathit{exp}}_{\text{application}} \mid \underbrace{\lambda \mathit{var}.\mathit{exp}}_{\text{abstraction}}$$

can be given a semantics using a cpo D s.t.

$$D = D \to D.$$

SEMANTICS OF RECURSIVE DOMAINS

<u>Idea</u> (Scott 1970): generalise the fixpoint approach (lecture 2) from values (elements of cpos) to domains (cpo's).

<u>Step 1 : express recursive domains as fixpoints</u>

<u>example</u>: $L = \mathbb{1} + (\mathbb{Z}_\perp \times L)$

$$\Downarrow$$

$$L = F(L)$$

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

$$F(A) \stackrel{\text{def}}{=} \mathbb{1} + (\mathbb{Z}_\perp \times A).$$

<u>Step 2: generalise the CPO fixpoint theorem</u>

<u>Thm</u>: every continuous $f : A \to A$ on a cpo A has a least fixpoint, given by the lub of the ω-chain

$$\perp \sqsubseteq f\perp \sqsubseteq ff\perp \sqsubseteq fff\perp \cdots$$

The generalisation is "<u>Scott's inverse limit construction</u>".

GENERALISING FROM VALUES TO CPO'S

<u>approximation</u> ($x \sqsubseteq y$)

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

⌈ <u>Note</u>: we might expect to require that f be injective, but this is not technically necessary! ⌋

<u>least element</u> ($\perp \sqsubseteq x$)

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

$$\mathbb{1} \xrightarrow{\lambda x.\perp} A.$$

⌈ <u>Aside</u>: the category CPO has no initial object; $\mathbb{1}$ is terminal, but also serves as a "pseudo" initial object. ⌋

<u>ω-chain</u>

A family $\{A_i \mid i \in \mathbb{N}\}$ of cpos, together with a family $\{f_i : A_i \to A_{i+1} \mid i \in \mathbb{N}\}$ of continuous fns:

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} A_2 \xrightarrow{f_2} A_3 \to \cdots$$

upper bound (of an ω-chain)

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

$$
\begin{tikzcd}
 & & & A & \cdots \\
A_0 \arrow[r, "f_0"'] \arrow[urrr, "g_0"] & A_1 \arrow[r, "f_1"'] \arrow[urr, "g_1"] & A_2 \arrow[r, "f_2"'] \arrow[ur, "g_2"] & A_3 \arrow[u, "g_3"] & \cdots
\end{tikzcd}
$$

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 $k : A \to B$ s.t.

$$
\begin{tikzcd}
 & & & B & \cdots \\
 & & & A \arrow[u, dashed, "k"'] & \cdots \\
A_0 \arrow[r, "f_0"'] \arrow[urrr, "g_0"] \arrow[uurrr, bend left, "h_0"] & A_1 \arrow[r, "f_1"'] \arrow[urr, "g_1"] \arrow[uurr, bend left, "h_1"] & A_2 \arrow[r, "f_2"'] \arrow[ur, "g_2"] \arrow[uur, bend left, "h_2"] & A_3 \arrow[u, "g_3"'] \arrow[uu, bend left, "h_3"] & \cdots
\end{tikzcd}
$$

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 $(A', \{g_i'\})$ 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 cocontinuous iff it preserves colimits of ω-chains, i.e. if $(A_., \{g_i\})$ is a colimit of

$$A_{.0} \xrightarrow{f_0} A_{.1} \xrightarrow{f_1} A_{.2} \xrightarrow{f_2} A_{.3} \cdots$$

then $(FA_., \{Fg_i\})$ is a colimit of

$$FA_{.0} \xrightarrow{Ff_0} FA_{.1} \xrightarrow{Ff_1} FA_2 \xrightarrow{Ff_2} FA_3 \cdots$$

fixpoint

A <u>fixpoint</u> of a mapping F on cpo's is a cpo A such that FA is iso to A, i.e. $FA \cong A$.

<u>Note</u>: 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

<u>Theorem</u>: every cocontinuous functor F has a least fixpoint, given by the colimit of the ω-chain

$$\mathbb{1} \xrightarrow{\lambda x.\bot} F\mathbb{1} \xrightarrow{F(\lambda x.\bot)} FF\mathbb{1} \xrightarrow{FF(\lambda x.\bot)} FFF\mathbb{1} - \cdots .$$

what kind of functors are cocontinuous?

<u>Fact</u>: if F is a mapping on cpo's built up from basic cpo's using the ops $\times, +, \rightarrow, \otimes, \oplus, \multimap, (-)_\bot$ from lecture 3, then F extends to a cocontinuous functor.

— * —

Example (binary numbers)

Consider the Haskell datatype definition

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

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

The corresponding recursive domain equation is

$$\mathcal{B} \cong \mathcal{B} + \mathbb{1} + \mathcal{B}.$$

Expressed as a fixpoint, this equation reads

$$\mathcal{B} \cong F(\mathcal{B}), \text{ where } F(X) \stackrel{\text{def}}{=} X + \mathbb{1} + X.$$

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

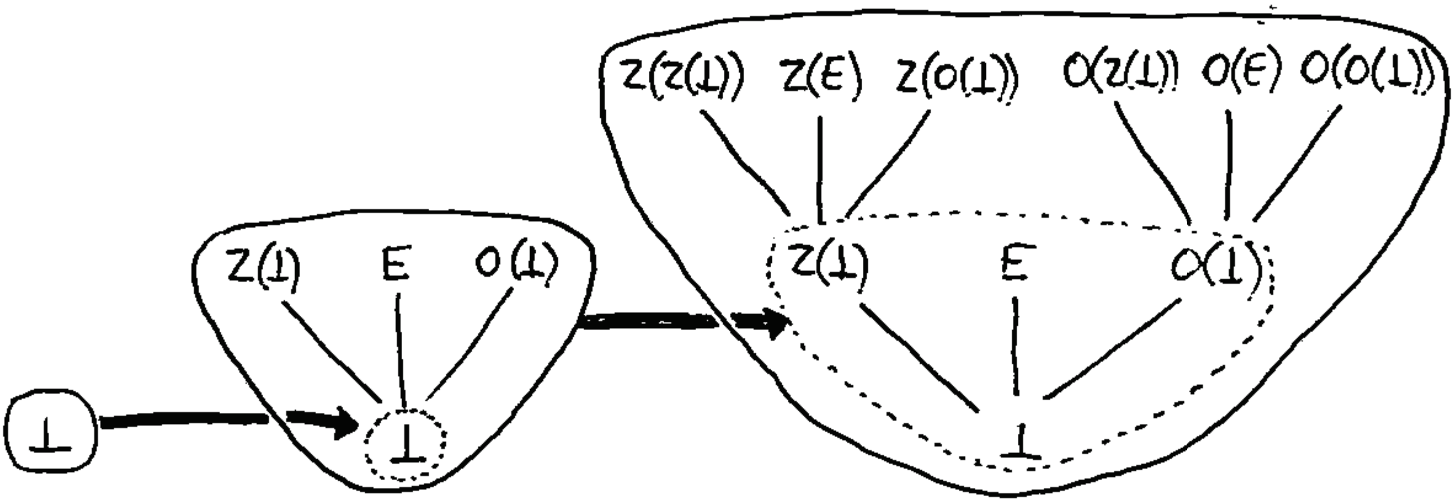
$$F(f) \stackrel{\text{def}}{=} f + id_{\mathbb{1}} + f \qquad \left[\frac{f : X \to Y}{Ff : FX \to FY}\right]$$

and this defn. makes F a cocontinuous functor.

Hence the recursive type Bin has a semantics as a least fixpoint of F, i.e. as the colimit of

$$\mathbb{1} \xrightarrow{\lambda x.\perp} F\mathbb{1} \xrightarrow{F(\lambda x.\perp)} FF\mathbb{1} \xrightarrow{FF(\lambda x.\perp)} FFF\mathbb{1} - \cdots$$

What does the chain look like?



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

The nth 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.

<u>Fact</u>: mappings **F** on cpo's, built using $\rightarrow$ and $\multimap$, <u>do</u> <u>not</u> in general extend to cocontinuous functors.

<u>example</u>: $F(A) \stackrel{\text{def}}{=} A \rightarrow \mathbb{Z}_\perp$ extends to continuous functions by $F(f) \stackrel{\text{def}}{=} \lambda g.\, g \circ f$, which doesn't even have the right type to be a functor!

$$\text{if } f : X \rightarrow Y \text{ then } Ff : (Y \rightarrow \mathbb{Z}_\perp) \rightarrow (X \rightarrow \mathbb{Z}_\perp), \text{ i.e. } Ff : FY \rightarrow FX$$

"swapped"

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

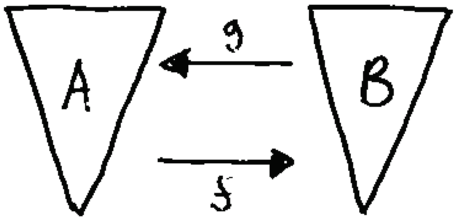
$$\frac{f : A \rightarrow B \quad g : C \rightarrow D}{f \rightarrow g : (B \rightarrow C) \rightarrow (A \rightarrow D)} \qquad f \rightarrow g \stackrel{\text{def}}{=} \lambda h.\, g \circ h \circ f$$

"swapped"

SOLUTION (Scott 1972, Smyth & Plotkin 1982)

Solve recursive domain equations using functors on $\mathbf{CPO}^R$ (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



such that

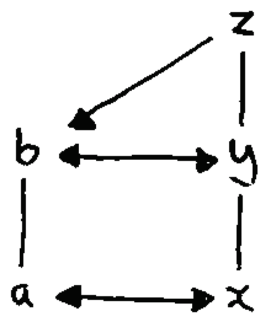
(1) $g \circ f = id_A$ $\quad$ [i.e. $\forall x \in A.\ g(f x) = x$.]

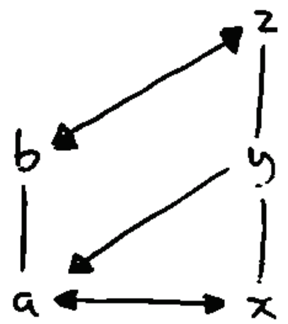
(2) $f \circ g \sqsubseteq id_B$ $\quad$ [i.e. $\forall y \in B.\ f(g y) \sqsubseteq y$.]

f is called an embedding ; and g a projection.

<u>Note</u>: "retraction pair" is a weakening of "iso": going $A \to B \to A$ info is <u>preserved</u> ($g \circ f = id_A$), but going $B \to A \to B$ info <u>can be lost</u> ($f \circ g \leq id_B$).

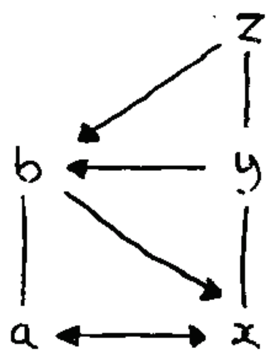
<u>Examples</u> (of retraction pairs)

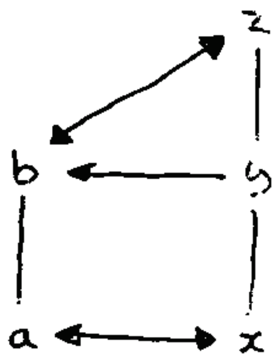




(i.e. there can be <u>many</u> retraction pairs $A \underset{f}{\overset{g}{\leftrightarrows}} B$.)

<u>Examples</u> (of <u>non</u>-rectraction pairs)





$b \mapsto x \mapsto a$, but $a \neq b$. $y \mapsto b \mapsto z$, but $z \nleq y$.

<u>Facts</u>: if $A \underset{f}{\overset{g}{\rightleftarrows}} B$ is a retraction pair, then

* f and g are <u>strict</u>: $f\bot = \bot$ and $g\bot = \bot$.

* retraction pairs <u>compose</u>:

$$\underbrace{A \underset{f}{\overset{g}{\rightleftarrows}} B \underset{h}{\overset{i}{\rightleftarrows}} C}_{\text{retraction pairs}} \qquad\qquad \underbrace{A \underset{h \circ f}{\overset{g \circ i}{\rightleftarrows}} C}_{\text{their composition}}$$

* g is <u>uniquely determined</u> by f, and vica-versa: e.g. if $A \underset{f}{\overset{g'}{\rightleftarrows}} B$ is another retraction pair then $g = g'$.

* A is <u>isomorphic</u> to the range $\{f\,x \mid x \in A\} \subseteq B$ 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 $A \rightleftarrows B$.

Fact: the other notions (least element, ω-chain, upper bound, colimit, ωcontinuous functor) naturally generalise from continuous fns to retraction pairs.

The fixpoint results generalise too...

Theorem: every ωcontinuous functor F on CPO^R has a least fixpoint, given by the colimit of the ω-chain

$$\mathbb{1} \underset{f_0}{\overset{g_0}{\rightleftarrows}} F\mathbb{1} \underset{f_1}{\overset{g_1}{\rightleftarrows}} FF\mathbb{1} \underset{f_2}{\overset{g_2}{\rightleftarrows}} FFF\mathbb{1} \rightleftarrows \cdots$$

where the retraction pairs are defined by

$$(f_0, g_0) \stackrel{\text{def}}{=} (\lambda x.\bot, \lambda x.\bot),$$

$$(f_{i+1}, g_{i+1}) \stackrel{\text{def}}{=} F(f_i, g_i).$$

Fact: the operators $\times, +, \rightarrow, \otimes, \oplus, \multimap, (-)_\bot$ on cpo's all extend to <u>covariant</u> functors on CPO^R.

Example (products)

$$(f,g) \times (h,i) \stackrel{\text{def}}{=} (f \times h,\ g \times i)$$

on retraction pairs

on continuous fns (lecture 3)

types:
$$\frac{A \underset{f}{\overset{g}{\leftrightarrows}} B \qquad C \underset{h}{\overset{i}{\leftrightarrows}} D}{(A \times C) \underset{f \times h}{\overset{g \times i}{\leftrightarrows}} (B \times D)}$$

Example (function spaces)

$$(f,g) \to (h,i) \stackrel{\text{def}}{=} (g \to h,\ f \to i)$$

types:
$$\frac{A \underset{f}{\overset{g}{\leftrightarrows}} B \qquad C \underset{h}{\overset{i}{\leftrightarrows}} D}{(A \to C) \underset{g \to h}{\overset{f \to i}{\leftrightarrows}} (B \to D)}$$

no swapping!

Fact: if F is a mapping on cpo's, built up from basic cpo's using the ops $\times, +, \to, \otimes, \oplus, \multimap, (-)_\perp$, then F extends to a cocontinuous functor on CPO^R.

WHAT DOES THE COLIMIT LOOK LIKE?

Defn (Scott): the "inverse limit" of an ω-chain

$$D_0 \underset{f_0}{\overset{g_0}{\rightleftarrows}} D_1 \underset{f_1}{\overset{g_1}{\rightleftarrows}} D_2 \underset{f_2}{\overset{g_2}{\rightleftarrows}} D_3 \rightleftarrows \cdots$$

of retraction pairs is the set of ω-tuples

$$D_\infty \stackrel{\text{def}}{=} \{(x_0, x_1, \dots) \mid x_i \in D_i \text{ and } x_i = g_i(x_{i+1})\}.$$

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_∞ is a cpo. More generally, if each cpo D_i is a Scott domain (lecture 4) then so is D_∞.

Fact: D_∞ is an upper bound of the ω-chain.

proof (sketch): we must construct a family

$$\{ D_i \underset{\theta_{i,\infty}}{\overset{\theta_{\infty,i}}{\rightleftarrows}} D_\infty \mid i \in \mathbb{N} \}$$

of retraction pairs, s.t. the following diagram commutes:

$$
\begin{tikzcd}
 & & D_\infty \arrow[dll, bend right] \arrow[dl, shift right] \arrow[d, shift right] & \cdots \\
D_0 \arrow[r, shift right, "f_0"'] \arrow[urr, bend left, shift left] & D_1 \arrow[l, shift right, "g_0"'] \arrow[r, shift right, "f_1"'] \arrow[ur, shift left] & D_2 \arrow[l, shift right, "g_1"'] \arrow[u, shift right] & \cdots
\end{tikzcd}
$$

<u>part 1 : $\Theta_{\infty,i} : D_\infty \to D_i$</u>

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

<u>part 2 : $\Theta_{i,\infty} : D_i \to D_\infty$</u>

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

where the auxilliary fns $\Theta_{i,j} : D_i \to D_j$ (for $i,j \in \mathbb{N}$) are defined by composing sequences of embeddings (fs) or projections (gs), depending on whether $i<j$ or $j<i$.

<u>example</u>:

$$
\begin{aligned}
\Theta_{2,\infty}\, x &= (\Theta_{2,0}\, x, \Theta_{2,1}\, x, \Theta_{2,2}\, x, \Theta_{2,3}\, x, \Theta_{2,4}\, x, \dots) \\
&= (\underbrace{g_0(g_1\, x), g_1\, x}_{\text{"approximations" to } x \text{ (since } g\text{s are projections)}}, \underbrace{x}_{x}, \underbrace{f_2\, x, f_3(f_2\, x), \dots}_{\text{"equivalent" to } x \text{ (since } f\text{s are embeddings)}})
\end{aligned}
$$

□

<u>Fact</u>: D_∞ is a colimit of the ω-chain.

—— * ——

<u>Example</u>: the untyped λ-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_∞, can be constructed as the colimit of the ω-chain

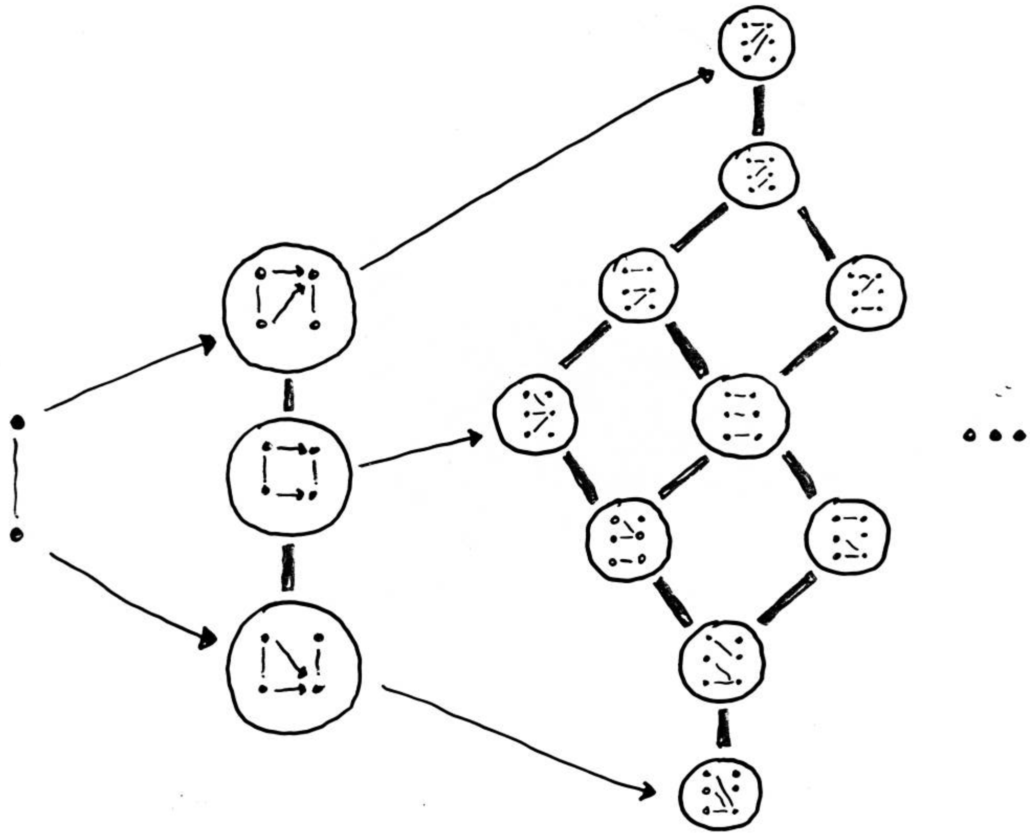
$$\mathbb{1} \rightleftarrows F\mathbb{1} \rightleftarrows FF\mathbb{1} \rightleftarrows \cdots$$

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

<u>But</u>... the least solution is <u>trivial</u> ($D_\infty = \mathbb{1}$) since $\mathbb{1} \cong \mathbb{1} \to \mathbb{1}$. A non-trivial solution is obtained by starting at $\mathbb{2} = \vdots$, rather than $\mathbb{1}$:

$$\mathbb{2} \rightleftarrows F\mathbb{2} \rightleftarrows FF\mathbb{2} \rightleftarrows \cdots.$$

In this case the ω-chain has the form



Finding a non-trivial model $D \cong D \to D$ of the untyped λ-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 $F(X) \stackrel{\text{def}}{=} \mathbb{1} + X$ and $F(f) \stackrel{\text{def}}{=} id_{\mathbb{1}} + f$ 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 $X \cong \mathbb{1} + X$, 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. Zero $\perp = \perp$)

b) Succ is strict (i.e. Succ $\perp = \perp$)

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

④ Show that $F(X) \stackrel{\text{def}}{=} \mathbb{Z}_\perp \to X$ can be extended to continuous fns such that if $f : A \to B$ then $Ff : FA \to FB$. [[Contrast with $F(X) \stackrel{\text{def}}{=} X \to \mathbb{Z}_\perp$ from slide 5.9.]]