Domain Theory

Part 5: Recursively Defined Domains

Dr. Liam O'Connor

based on material from

Graham Hutton, John Longley, Robert Muller, Dana Scott, Joseph E. Stoy, Carl Gunter, Glynn Winskel

March 25, 2024

1 Introduction

Suppose we have a language with recursive data types, such as this *List* type:

$$\textbf{data}\ \mathit{List} = \mathsf{Nil} \mid \mathsf{Cons}\ (\mathit{Int} \times \mathit{List})$$

As we saw previously with PCF, the denotation of a type τ is the domain which contains all the denotations of all closed expressions of type τ. What, then, is the domain that corresponds to *List* α? If we can find a cpo L such that[1]:

$$L \simeq \mathbf{1} + (\mathbb{Z}_\perp \times L)$$

Then this cpo L would serve adequately as a semantics. Similarly, in untyped lambda calculus, lambda values can only be functions on lambda values:

$$D \simeq D \to D$$

How can we find solutions to such recursive equations? How can we guarantee the existence of a (least) solution?

2 From Elements to Domains

We can start by generalising the fixed point approach we used for values (i.e. elements of cpos) to domains (i.e. cpos themselves).

Example

Our previous equations for lists:

$$L \simeq \mathbf{1} + (\mathbb{Z}_\perp \times L)$$

Can be expressed as the least fixed point of this mapping (specifically an endofunctor) $\mathcal{F}$ on cpos:

$$\mathcal{F}(A) \triangleq \mathbf{1} + (\mathbb{Z}_\perp \times A)$$

To ensure that least fixed points exist, and to give us a means of finding them, we must now generalise all of the concepts we used for least fixed points on values (information ordering, least upper bounds, continuity etc.) to domains themselves.

[1] where $\mathbf{1}$ is the CPO containing just one element $\perp$.

2.1 Information Ordering

> **Note**
>
> We will see later that this definition is insufficient when using function constructions, but it will suffice for our purposes for now.

Let us say (for now) that a cpo A *approximates* a cpo B (i.e. $A \sqsubseteq B$) iff there is a continuous function $f : A \to B$. Then, there is a least element for this ordering: the one-element cpo $\mathbf{1} = \{\bot\}$, as the continuous function $(\lambda x.\bot) : \mathbf{1} \to A$ exists for any cpo A.

> **Categorical aside**
>
> The category **Cpo** has no initial object. The cpo $\mathbf{1}$ is terminal; but also serves as a "pseudo" initial object due to the above.

2.2 Chains and Lubs

An ω-chain in this context consists of a family of cpos $\{A_i \mid i \in \mathbb{N}\}$, together with a family continuous functions $\{f_i : A_i \to A_{i+1} \mid i \in \mathbb{N}\}$, shown below[2]:

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

A cpo A is an upper bound of such an ω-chain if there is a family of continuous functions $\{g_i : A_i \to A \mid i \in \mathbb{N}\}$ such that the following diagram commutes (i.e. $g_i = g_{i+1} \circ f_i$ for all $i \in \mathbb{N}$):

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

A cpo A is the *least* upper bound of such an ω-chain if there further exists a *unique* k for any other upper bound B such that the following diagram commutes (i.e. $h_i = g_i \circ k$ for all $i \in \mathbb{N}$):

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

The least upper bound of such an ω-chain is also called its colimit.

[2] Once again, note this definition will be strengthened when we strengthen our information ordering later on.

Note

Uniqueness of lubs is up to isomorphism. That is, if A and B are both colimits of our ω-chain, then $A \simeq B$.

2.3 Monotonic and Continuous Functions

Endofunctors

An endofunctor on the category **Cpo** is a functor **Cpo** $\to$ **Cpo**, that is a mapping $\mathcal{F}$ on cpos together with a mapping $\mathcal{F}$ on continuous functions, such that:

1. If $f : A \to B$ then $\mathcal{F}(f) : \mathcal{F}(A) \to \mathcal{F}(B)$
2. $\mathcal{F}(\mathsf{id}_A : A \to A) = \mathsf{id}_{\mathcal{F}(A)} : \mathcal{F}(A) \to \mathcal{F}(A)$
3. $\mathcal{F}(f \circ g) = \mathcal{F}(f) \circ \mathcal{F}(g)$

Observe that, given our information ordering for cpos given earlier, the functor laws necessarily imply monotonicity for all functors $\mathcal{F}$.

Because our generalisation of lubs is a *co*limit, our notion of continuity for a functor $\mathcal{F}$ is called *co*continuity. An functor $\mathcal{F}$ is cocontinuous iff it preserves colimits of ω-chains. That is, given this chain where A is a colimit:

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

Then $\mathcal{F}(A)$ is a colimit in the following chain:

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

2.4 Fixed Points

A fixed point of endofunctor on cpos $\mathcal{F}$ is a cpo $\mathcal{F}$ such that $\mathcal{F}(A) \simeq A$.

Note

In Scott's approach, which we follow here, our fixed points are up to isomorphism (suitable for languages with isorecursive types), but there are other approaches where they are equalities (suitable for languages with equirecursive types).

Our fixed point theorem generalises much as one might expect:

Theorem: Every cocontinuous endofunctor $\mathcal{F}$ on cpos has a least fixed point, given by the colimit of the ω-chain:

$$\mathbf{1} \xrightarrow{\lambda x.\bot} \mathcal{F}(\mathbf{1}) \xrightarrow{\mathcal{F}(\lambda x.\bot)} \mathcal{F}(\mathcal{F}(\mathbf{1})) \xrightarrow{\mathcal{F}(\mathcal{F}(\lambda x.\bot))} \mathcal{F}(\mathcal{F}(\mathcal{F}(\mathbf{1}))) \cdots\cdots\cdots$$

Example

Consider the Haskell-style data type:

$$\textbf{data}\ Bin = \mathsf{Zero}\ Bin \mid \mathsf{Empty} \mid \mathsf{One}\ Bin$$

So, e.g. $\mathsf{One}\ (\mathsf{Zero}\ (\mathsf{One}\ \mathsf{Empty})) : Bin$. The recursive domain equation is, expressed as a fixed point:

$$B \simeq \mathcal{F}(B) \quad \text{where } \mathcal{F}(X) = X + \mathbf{1} + X$$

We wish to show that $\mathcal{F}$ is a cocontinuous endofunctor. We have a mapping $\mathcal{F}$ on cpos (objects of the category **Cpo**), but for it to be a functor we additionally need a mapping $\mathcal{F}$ on continuous functions (morphisms of the category **Cpo**).

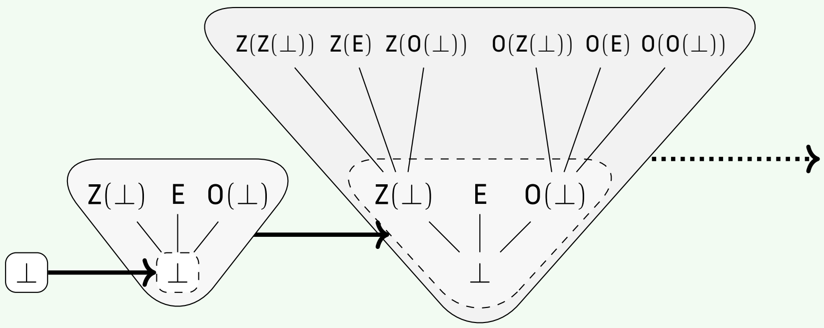
Recalling our *sum* construction from previous lectures, we may remember that sums are already a bifunctor $\mathbf{Cpo} \times \mathbf{Cpo} \to \mathbf{Cpo}$. Thus, our mapping on continuous functions $\mathcal{F}$ can be *derived* from our mapping on cpos $\mathcal{F}$ by using the morphism mapping from the sum construction. Given a continuous function $f : A \rightarrow B$, our morphism mapping is:

$$\begin{aligned}\mathcal{F}(f) &: \mathcal{F}(A) \rightarrow \mathcal{F}(B)\\ \mathcal{F}(f) &\triangleq f + \mathsf{id}_{\mathbf{1}} + f\end{aligned}$$

Where $(+) : (A \rightarrow B) \times (C \rightarrow D) \to (A \times C) \rightarrow (B \times D)$ is the function defined in previous lectures. Proof that this endofunctor is cocontinuous is left as an exercise. Hence the semantics of the type *Bin*, i.e. $[\![Bin]\!] : \mathbf{Cpo}$ is the least fixed point of $\mathcal{F}$, i.e. the colimit of the ω-chain:

$$\mathbf{1} \xrightarrow{\lambda x.\bot} \mathcal{F}(\mathbf{1}) \xrightarrow{\mathcal{F}(\lambda x.\bot)} \mathcal{F}(\mathcal{F}(\mathbf{1})) \xrightarrow{\mathcal{F}(\mathcal{F}(\lambda x.\bot))} \mathcal{F}(\mathcal{F}(\mathcal{F}(\mathbf{1}))) \cdots\cdots\cdots$$

Let us visualise this chain[a]:



Thus, the nth cpo in the chain contains binary numbers with at most n defined digits. The colimit of the chain contains all binary numbers (finite, partial, and infinite!).

[a] Here the sum injections have been written as Z, E, O rather than (combinations of) inl and inr to keep the connection with the Haskell data type clear.

3 From Cpo to $\mathbf{Cpo}^R$

Given a mapping on cpos $\mathcal{F} : \mathbf{Cpo} \to \mathbf{Cpo}$ that is made up of the primitives $\times$, $+$, $\otimes$, $\oplus$, $\mathbf{1}$ and $(\cdot)_\perp$, we can *extend* it to an endofunctor by using the underlying functor structure of these constructions, generating a morphism mapping $\mathcal{F} : (A \rightarrow B) \to (\mathcal{F}(A) \rightarrow \mathcal{F}(B))$.

Examples

$$\begin{array}{lcll} \mathcal{F}(X) = X + \mathbf{1} & \rightsquigarrow & \mathcal{F}(f) = f + \mathsf{id}_\mathbf{1} & \text{((co)-natural numbers)} \\ \mathcal{F}(X) = \mathbf{1} + (\mathbb{Z}_\perp \times X) & \rightsquigarrow & \mathcal{F}(f) = \mathsf{id}_\mathbf{1} + (\mathsf{id}_{\mathbb{Z}_\perp} \times f) & \text{((co)-lists of integers)} \end{array}$$

A Serious Problem

This approach breaks down for $\multimap$ and $\rightarrow$, as they are contravariant in their first argument. This means the functor that extend to is not $\mathbf{Cpo} \to \mathbf{Cpo}$ but $\mathbf{Cpo}^{op} \to \mathbf{Cpo}$. The morphisms end up the wrong way around.

As an example, consider $\mathcal{F}(X) = X \rightarrow \mathbb{Z}_\perp$. Then, recalling the morphism mapping of the $\rightarrow$ functor:

$$\frac{f : A \rightarrow B \qquad g : C \rightarrow D}{f{\rightarrow}g : (B \rightarrow C) \rightarrow (A \rightarrow D)}$$

We can generate a morphism mapping for $\mathcal{F}$:

$$\mathcal{F}(f) \;=\; f{\rightarrow}\mathsf{id}_{\mathbb{Z}_\perp} \;=\; (\lambda h.\, \mathsf{id}_{\mathbb{Z}_\perp} \circ h \circ f) \;=\; (\lambda h.\, h \circ f)$$

However, this mapping has the wrong type. For $f : A \rightarrow B$, then $\mathcal{F}(f) : (B \rightarrow \mathbb{Z}_\perp) \rightarrow (A \rightarrow \mathbb{Z}_\perp)$, which is $\mathcal{F}(B) \rightarrow \mathcal{F}(A)$, not the required $\mathcal{F}(A) \rightarrow \mathcal{F}(B)$. This is because the generated functor is contravariant, not covariant.

3.1 Retraction Pairs

Rather than solve our recursive equations using fixed points of endofunctors on **Cpo** (the category of cpos and continuous functions), we will use endofunctors on $\mathbf{Cpo}^R$, the category of cpos and retraction pairs.

Definition

A retraction pair (f, g) on cpos A to B consists of continuous functions $A \underset{g}{\overset{f}{\rightleftarrows}} B$ s.t.:

1. $g \circ f = \mathsf{id}_A$ (i.e. $\forall x \in A.\ g(f(x)) = x$)
2. $f \circ g \sqsubseteq \mathsf{id}_B$ (i.e. $\forall y \in B.\ f(g(y)) \sqsubseteq y$)

In this retraction pair, f is called a embedding and g is called a projection.

Retraction pairs are weakenings of isomorphisms. Going $A \rightarrow B \rightarrow A$, all information is *preserved* due to requirement 1, but going $B \rightarrow A \rightarrow B$ *may lose* some information (hence the use of $\sqsubseteq$ in requirement 2). Using retraction pairs instead of mere continuous functions as our morphisms will enable us to assign a semantics to recursive function types, but first, let us familiarise ourselves with them.

Examples and Counterexamples

Some examples of retraction pairs:

$$
\begin{tikzcd}
& z \arrow[dl] \arrow[d, dash] \\
b \arrow[r, leftrightarrow] \arrow[d, dash] & y \arrow[d, dash] \\
a \arrow[r, leftrightarrow] & x
\end{tikzcd}
\qquad
\begin{tikzcd}
& z \arrow[d, dash] \\
b \arrow[ur, leftrightarrow] \arrow[d, dash] & y \arrow[d, dash] \arrow[dl] \\
a \arrow[r, leftrightarrow] & x
\end{tikzcd}
$$

(this demonstrates that there can be *many* retraction pairs $A \underset{g}{\overset{f}{\rightleftarrows}} B$)

The following, however, are *not* retraction pairs:

$$
\begin{tikzcd}
& z \arrow[d, dash] \\
b \arrow[ur, leftrightarrow] \arrow[d, dash] & y \arrow[l] \arrow[d, dash] \\
a \arrow[r, leftrightarrow] & x
\end{tikzcd}
\qquad
\begin{tikzcd}
& z \arrow[dl] \arrow[d, dash] \\
b \arrow[dr] \arrow[d, dash] & y \arrow[l] \arrow[d, dash] \\
a \arrow[r, leftrightarrow] & x
\end{tikzcd}
$$

($y \mapsto b \mapsto z$ but $z \not\sqsubseteq y$) ($b \mapsto x \mapsto a$ but $a \neq b$)

We can compose retraction pairs by composing their embeddings and projections:

$$A \underset{g}{\overset{f}{\rightleftarrows}} B \underset{i}{\overset{h}{\rightleftarrows}} C \qquad \rightsquigarrow \qquad A \underset{g \circ i}{\overset{h \circ f}{\rightleftarrows}} C$$

It follows from the definition of retraction pairs that, for a pair $A \underset{g}{\overset{f}{\rightleftarrows}} B$:

1. f and g are strict, i.e. $f(\bot) = \bot$ and $g(\bot) = \bot$.
2. g is *uniquely determined* by f and vice-versa, so if another retraction pair $A \underset{g'}{\overset{f}{\rightleftarrows}} B$ exists, then $g = g'$. To see why, remember that f and g must be continuous and therefore monotonic.
3. A is isomorphic to the range of f, i.e. $\{f(x) \mid x \in A\}$.

This last fact allows us to define a more intuitive notion of approximation, or information ordering, for cpos. Rather than say that for cpos A and B, $A \sqsubseteq B$ iff there exists a continuous function $A \twoheadrightarrow B$, we now say that:

$$A \sqsubseteq B \text{ iff there exists a retraction pair } A \underset{g}{\overset{f}{\rightleftarrows}} B \ .$$

3.2 Generalising Fixed Points

Fact

All of our other notions for **Cpo** naturally generalise to a setting with retraction pairs $\mathbf{Cpo}^R$: least elements, ω-chains, upper bounds, colimits, cocontinuous endofunctors...

Our fixed point theorem is exactly the same as before, except we now will use retraction pairs instead of continuous functions.

Theorem: Every cocontinuous endofunctor $\mathcal{F}$ on $\mathbf{Cpo}^{\mathsf{R}}$ has a least fixed point, given by the colimit of the ascending ω-chain:

$$\mathbf{1} \underset{g_0}{\overset{f_0}{\rightleftarrows}} \mathcal{F}(\mathbf{1}) \underset{g_1}{\overset{f_1}{\rightleftarrows}} \mathcal{F}(\mathcal{F}(\mathbf{1})) \underset{g_2}{\overset{f_2}{\rightleftarrows}} \mathcal{F}(\mathcal{F}(\mathcal{F}(\mathbf{1}))) \cdots$$

Where the retraction pairs (f_i, g_i) are defined by:

$$\begin{array}{lcl} (f_0, g_0) & \triangleq & (\lambda x.\bot, \lambda x.\bot) \\ (f_{i+1}, g_{i+1}) & \triangleq & \mathcal{F}(f_i, g_i) \end{array}$$

3.3 Extending Cpo Mappings to Endofunctors on $\mathbf{Cpo}^{\mathsf{R}}$

Fact

If $\mathcal{F}$ is a mapping on cpos built up from basic cpos using the constructions $\times$, $\otimes$, $+$, $\oplus$, $(\cdot)_\bot$, *as well as* $\to$ and $\circ\!\!\to$, then $\mathcal{F}$ extends to a (covariant!) cocontinuous functor on $\mathbf{Cpo}^{\mathsf{R}}$.

3.3.1 Products

Given two retraction pairs (f, g) and (h, i), our retraction pair for their product is just the product of their embeddings and projections, i.e. $(f \times h, g \times i)$, using the product operation on continuous functions from previous lectures:

$$\frac{A \underset{g}{\overset{f}{\rightleftarrows}} B \qquad C \underset{i}{\overset{h}{\rightleftarrows}} D}{A \times C \underset{g \times i}{\overset{f \times h}{\rightleftarrows}} B \times D}$$

3.3.2 Functions

We are given two retraction pairs $A \underset{g}{\overset{f}{\rightleftarrows}} B$ and $C \underset{i}{\overset{h}{\rightleftarrows}} D$.
To define a suitable covariant functor, we define our morphism mapping in terms of the $\to$ operation on continuous functions defined in earlier lectures:

$$(f, g) \to (h, i) \triangleq (g \to h, f \to i)$$

Note that the positions of f and g are swapped in the above definition.

$$\frac{A \underset{g}{\overset{f}{\rightleftarrows}} B \qquad C \underset{i}{\overset{h}{\rightleftarrows}} D}{A \to C \underset{f \to i}{\overset{g \to h}{\rightleftarrows}} B \to D}$$

Unlike previously with the category $\mathbf{Cpo}$, this functor is covariant: Note the positions of A and B are not swapped!

3.4 Details of the Colimit

Definition

Given an ω-chain of retraction pairs:

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

The colimit (or inverse limit) is the set of ω-tuples:

$$D_\infty \triangleq \{(x_0, x_1, \ldots) \mid x_i \in D_i \wedge x_i = g_i(x_{i+1})\}$$

That is, it is countable sequence of elements, one for each domain D_i, where each element is consistent with earlier elements.

Ordering: The ordering is the pointwise ordering of products, naturally generalised to ω-tuples. Under this ordering, D_∞ is a cpo. In fact if each D_i is a Scott domain, so is D_∞.

Let us prove that D_∞ is an upper bound of the ω-chain. We must construct a family of retraction pairs:

$$\left\{ D_i \underset{\theta_{\infty,i}}{\overset{\theta_{i,\infty}}{\rightleftarrows}} D_\infty \;\middle|\; i \in \mathbb{N} \right\}$$

such that the following diagram commutes:

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

Defining the projections of the retraction pairs $D_i \underset{\theta_{\infty,i}}{\overset{\theta_{\infty,i}}{\rightleftarrows}} D_\infty$ is straightforward:

$$\theta_{\infty,i}(x_0, x_1, \ldots) \triangleq x_i$$

However, to define the embeddings $\theta_{i,\infty}$ requires us to, for a given value $x \in D_i$, produce an ω-tuple of values consistent with x in every domain D_k. We do this by defining $\theta_{i,\infty}$ in terms of helper functions $\theta_{i,j}$:

$$\theta_{i,\infty}(x) \triangleq (\theta_{i,0}(x), \theta_{i,1}(x), \theta_{i,2}(x), \ldots)$$

The helper functions $\theta_{i,j}$ are defined by composing sequences of embeddings (fs) or projections (gs), depending on whether $i < j$ or $j < i$. For example, when $i = 2$:

$$\begin{aligned}
\theta_{2,\infty}(x) &= (\theta_{i,0}(x), \theta_{i,1}(x), \theta_{i,2}(x), \theta_{i,3}(x), \theta_{i,4}(x), \ldots) \\
&= (\underbrace{g_0(g_1(x)), g_1(x)}_{\textit{approximations to } x}, x, \underbrace{f_2(x), f_3(f_2(x)), \ldots}_{\textit{equivalent to } x})
\end{aligned}$$

Fact

D_∞ is the colimit of the ω-chain:

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

4 Untyped λ calculus

Recall the untyped λ calculus:

$$e ::= x \mid e_1\, e_2 \mid \lambda x.\, e$$

We mentioned before that the semantic domain of this language must be a cpo D such that

$$D \triangleq D \to D$$

In other words, solutions to this equation are fixed points of $\mathcal{F}(X) = X \to X$. The least such fixed point can expressed, by the theorem above, as the colimit of the ω-chain:

$$\mathbf{1} \underset{g_0}{\overset{f_0}{\rightleftarrows}} \mathcal{F}(\mathbf{1}) \underset{g_1}{\overset{f_1}{\rightleftarrows}} \mathcal{F}(\mathcal{F}(\mathbf{1})) \underset{g_2}{\overset{f_2}{\rightleftarrows}} \mathcal{F}(\mathcal{F}(\mathcal{F}(\mathbf{1}))) \cdots$$

But, the least solution to this equation is *trivial*: $D_\infty \simeq \mathbf{1}$, because $\mathbf{1} \simeq \mathbf{1} \to \mathbf{1}$. A non-trivial solution is obtained by starting the chain at $\mathbf{2}$, the Scott domain containing just $\{\top, \bot\}$, rather than $\mathbf{1}$.

$$\mathbf{1} \underset{g_0}{\overset{f_0}{\rightleftarrows}} \mathcal{F}(\mathbf{2}) \underset{g_1}{\overset{f_1}{\rightleftarrows}} \mathcal{F}(\mathcal{F}(\mathbf{2})) \underset{g_2}{\overset{f_2}{\rightleftarrows}} \mathcal{F}(\mathcal{F}(\mathcal{F}(\mathbf{2}))) \cdots$$

We will now sketch a proof of the retraction pair between D_∞ and $D_\infty \to D_\infty$, by providing two functions, $\mathbf{up} : D_\infty \to (D_\infty \to D_\infty)$ and its inverse $\mathbf{down} : (D_\infty \to D_\infty) \to D_\infty$.

Observe that each element of D_i (for $i > 0$) is a *function* whose domain is D_{i-1}. Therefore, an element of D_∞ is an ω-tuple of such functions. To define $\mathbf{up}(d)(x)$, we must essentially "apply" each function in the element $d \in D_\infty$ (viewed as an ω-tuple of functions) to the element $x \in D_\infty$ (viewed as an ω-tuple of value). More specifically, we say that $\mathbf{up}(d)(x)$ is an ω-tuple $(y_m \mid y \in \mathbb{N})$, as follows:

$$y_m = \bigsqcup_{k \in \mathbb{N}} \theta_{m+k,m}(d_{m+k+1}(x_{m+k}))$$

To define $\mathbf{down}(f)$, we are given a (continuous) function $f : D_\infty \to D_\infty$ and must construct the ω-tuple of approximations at every D_n. We do this by projecting the action of f down to D_n. We say that $\mathbf{down}(f)$ is an ω-tuple $(v_n \mid n \in \mathbb{N})$ where:

$$\begin{array}{lcl} v_0 & \triangleq & \theta_{\infty,0}(f(\theta_{0,\infty}(\bot_{D_0}))) \\ v_{n+1} & \triangleq & \theta_{\infty,n} \circ f \circ \theta_{n,\infty} \end{array}$$

4.1 Semantics for untyped λ-calculus

Using the new functions **up** and **down**, it is now straightforward to define a semantics for untyped λ-calculus, where σ is an environment $\mathsf{Var} \to D_\infty$:

$$\begin{array}{lcl} [\![\cdot]\!] : (\mathsf{Var} \to D_\infty) \to D_\infty \\ [\![x]\!]\sigma & = & \sigma(x) \\ [\![e_0\ e_1]\!]\sigma & = & \mathbf{up}([\![e_0]\!]\sigma)([\![e_1]\!]\sigma) \\ [\![\lambda x.\, e]\!]\sigma & = & \mathbf{down}(\Lambda v \in D_\infty.\, [\![e]\!]\ \sigma(x \mapsto v)) \end{array}$$

This semantics does not distinguish non-terminating computations from terminating ones. To model call-by-value more faithfully, we could use the equation $D \simeq D \to D_\bot$ instead, and for call-by-name we could use $D \simeq D_\bot \to D_\bot$. The constructions are very similar.

Exercises

1. Show that $\mathcal{F}(X) \triangleq \mathbf{1} + X$ and $\mathcal{F}(f) \triangleq \mathsf{id}_1 + f$ define a functor $\mathcal{F}$, i.e. that they satisfy the functor laws.

2. The lazy natural numbers, also called conats, are defined as the least solution to the equation $X \simeq \mathbf{1} + X$, i.e. as the fixed point of $\mathcal{F}$ in the previous question. Draw the first four approximations to the least fixed point. Call the two sum injections zero and succ.

3. Repeat exercise 2 assuming:
   a) zero is strict (i.e. zero $\bot = \bot$)
   b) succ is strict (i.e. succ $\bot = \bot$)

   What types (up to iso) result from these least fixed points?

4. Show that, in contrast to $\mathcal{F}(X) = X \rightharpoonup \mathbb{Z}_\bot$ from earlier, the mapping $\mathcal{G}(X) = \mathbb{Z}_\bot \rightharpoonup X$ *can* be extended to continuous functions giving a covariant endofunctor in **Cpo**.

Glossary

cocontinuous A functor $\mathcal{F}$ is *cocontinuous* iff it preserves colimits of ω-chains . 3, 4, 6, 7

colimit Colimits are a categorical generalisation of the least upper bound. 2–4, 6–10

contravariant A contravariant functor $\mathcal{F}$ is a functor that, instead of associating a morphism $X \xrightarrow{m} Y$ with a morphism $\mathcal{F}(X) \xrightarrow{\mathcal{F}(m)} \mathcal{F}(Y)$, it instead gives a morphism $\mathcal{F}(Y) \xrightarrow{\mathcal{F}(m)} \mathcal{F}(X)$. 5, 10

covariant A covariant functor is a functor that is not contravariant. 5, 7, 10

embedding In a retraction pair from A to B, the *embedding* is the function $f : A \to B$ that retains all information. 5–8, 10

endofunctor An *endofunctor* on a category **C** is a functor from **C** to **C**. Usually covariant unless otherwise specified. 1, 3–7, 10

equirecursive A type system in which a recursive type $\mu x.\tau$ is considered definitionally equal to $\tau[x := \mu x.\tau]$. 3

isorecursive A type system in which a recursive type $\mu x.\tau$ is not definitionally equal to $\tau[x := \mu x.\tau]$, but there is isomorphism consisting of an embedding (usually called unroll) and a projection (usually called roll) to convert between them . 3

projection In a retraction pair from A to B, the *projection* is the function $g : B \to A$ that may lose some information. 5–8, 10

retraction pair Also called an embedding-projection pair, consists of two continuous functions $f : A \rightharpoonup B$ and $g : B \rightharpoonup A$ such that:

1. $g \circ f = \mathsf{id}_A$ (i.e. $\forall x \in A.\ g(f(x)) = x$)
2. $f \circ g \sqsubseteq \mathsf{id}_B$ (i.e. $\forall y \in B.\ f(g(y)) \sqsubseteq y$)

. 5–10