

Fagprojekt

Adequate Semantics

Studies in Domain Theory

⋮
Bodil Biering
bodil@math.ku.dk

⋮
Rasmus Lerchedahl
rusmus@diku.dk

IT-University of Copenhagen (IT-C)

Spring 2003

Supervisor: Lars Birkedal

Abstract

This project is in two parts. The first part introduces a number of categorical constructions, some of which will be used in the second part. Many, however, will not and the treatment of categories of domains is in itself an aim of this project.

The second part introduces the programming language PCF and defines two semantics for call-by-value evaluation: An operational semantic, based on a calculus, and a denotational semantics, based on entities from the first part of the project. A notion of correspondence between the two semantics, called adequacy, is then defined and proved to hold. We also sketch two similar semantics for PCF with call-by-name evaluation.

The project should be readable by anyone knowing basic category theory though familiarity with semantics of programming languages may be helpful.

Contents

I	Categorical concepts	2
1	Preliminaries	3
1.1	Partial orders and monotone functions	3
1.2	Monads	6
2	Useful constructions	8
2.1	The lifting functor	8
2.2	The category of algebras for a monad	9
2.3	The Kleisli category of a monad	10
2.4	The CoKleisli Category of a Comonad	13
2.4.1	Motivation	13
2.4.2	Definitions	13
2.4.3	Construction	14
2.4.4	Verification	15
2.4.5	Application	19
3	Closure properties	21
3.1	SMCC	24
3.2	Partial exponentials	28
II	Adequacy	31
4	Call-by-value	32
4.1	The language PCF	32
4.2	Call-by-value operational semantics for PCF	34
4.3	Call-by-value denotational semantics	35
4.4	pCpo gives an adequate model for PCF-call-by-value	38
5	Call-by-name	44
5.1	Cppo gives an adequate model for PCF-call-by-name	45

Part I

Categorical concepts

Chapter 1

Preliminaries

This chapter contains some preliminary definitions and results.

1.1 Partial orders and monotone functions

Definition 1.1.1. A function $f : P \rightarrow Q$, where P and Q are partial orders is monotone iff for all $x, y \in P$

$$x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y).$$

Remark 1.1.2. Conventions: To express the fact that a partial function f is defined for a certain input x , say, we write $f(x) \downarrow$. For two mathematical expressions e and e' , possibly denoting elements of a partially ordered set, $e \sqsubseteq e'$ means that if $e \downarrow$ then also $e' \downarrow$ and $e \sqsubseteq e'$.

Definition 1.1.3. A partial function f is monotone iff for all x, y in P

$$x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y).$$

Lemma 1.1.4. Let $f : P \rightarrow Q$ be monotone. Then

$$e \sqsubseteq e' \Rightarrow f(e) \sqsubseteq f(e').$$

Proof: Suppose $e \sqsubseteq e'$. If $f(e) \downarrow$ then $e \downarrow$ and so by assumption $e' \downarrow$ since $e \sqsubseteq e'$. Because f is monotone, we then have $f(e) \sqsubseteq f(e')$. \square

Lemma 1.1.5. Composition of monotone partial functions is monotone.

Proof: Suppose $f : P \rightarrow Q$ and $g : Q \rightarrow R$ are monotone partial functions. Let $x, y \in P$, $x \sqsubseteq y$ then $f(x) \sqsubseteq f(y)$ as f is monotone, and by lemma 1.1.4 and g monotone $g(f(x)) \sqsubseteq g(f(y))$. \square

Definition 1.1.6. An ω -chain complete (or just complete for short) partial order (cpo) is a partial order, P , such that every increasing chain $x_0 \sqsubseteq x_1 \sqsubseteq \dots$ has a least upper bound $\bigsqcup_n x_n$.

Definition 1.1.7. A function $f : P \rightarrow Q$ of cpos is continuous iff it is monotone and for every ω -chain $(x_n)_{n \in \mathbb{N}}$, $f(\bigsqcup_n x_n) = \bigsqcup_n f(x_n)$.

To two (ore more) cpos P_1, P_2 we associate their product $P_1 \times P_2$, which is $\{\langle x_1, x_2 \rangle \mid x_i \in P_i\}$ with the componentwise ordering

$$\langle x_1, x_2 \rangle \sqsubseteq \langle y_1, y_2 \rangle \text{ iff } x_1 \sqsubseteq y_1 \text{ and } x_2 \sqsubseteq y_2.$$

This is a cpo with $\bigsqcup_n \langle x_1^{(n)}, x_2^{(n)} \rangle = \langle \bigsqcup_n x_1^{(n)}, \bigsqcup_n x_2^{(n)} \rangle$.

Lemma 1.1.8. *Let P be a cpo and $(x_{ij})_{i,j \in \mathbb{N}}$ be a doubly indexed set of elements of P such that $i \leq i' \wedge j \leq j' \Rightarrow x_{ij} \sqsubseteq x_{i'j'}$. Then*

$$\bigsqcup_i \bigsqcup_j x_{ij} = \bigsqcup_j \bigsqcup_i x_{ij} = \bigsqcup_i x_{ii}.$$

Proof: First, note that for all j , $(x_{ij})_i$ is a chain and for all i , $(x_{ij})_j$ is a chain. Let $y_i = \bigsqcup_j x_{ij}$. Now $(y_i)_i$ is also a chain because if $i \leq i'$ then, for all j ,

$$x_{ij} \sqsubseteq x_{i'j} \sqsubseteq \bigsqcup_j x_{i'j} = y_{i'}$$

so $y_{i'}$ is an upper bound of $(x_{ij})_j$ and since y_i is the least upper bound, we must have $y_i \sqsubseteq y_{i'}$. By a similar argument $z_j = \bigsqcup_i x_{ij}$ gives a chain $(z_j)_j$. Now

$$\begin{array}{lll} \forall i, j & x_{ij} \sqsubseteq \bigsqcup_j x_{ij} = y_i & \\ \forall j & z_j = \bigsqcup_i x_{ij} \sqsubseteq \bigsqcup_i y_i & \text{so } \bigsqcup_i y_i \text{ is an upper bound of } (z_j)_j \\ & \bigsqcup_j z_j = \bigsqcup_j \bigsqcup_i x_{ij} \sqsubseteq \bigsqcup_i y_i = \bigsqcup_i \bigsqcup_j x_{ij} & \text{because } \bigsqcup_j z_j \text{ is the least upper bound.} \end{array}$$

The inequality $\bigsqcup_i \bigsqcup_j x_{ij} \sqsubseteq \bigsqcup_j \bigsqcup_i x_{ij}$ is shown in the exact same way. To see that $\bigsqcup_i \bigsqcup_j x_{ij} = \bigsqcup_i x_{ii}$ we note that

$$\begin{array}{lll} \forall i, j & x_{ij} \sqsubseteq x_{kk} & \text{where } k = \max(i, j), \text{ so} \\ \forall i, j & x_{ij} \sqsubseteq \bigsqcup_k x_{kk} & \text{which implies} \\ & \bigsqcup_i \bigsqcup_j x_{ij} \sqsubseteq \bigsqcup_k x_{kk}. & \end{array}$$

On the other hand

$$\begin{array}{lll} \forall i & x_{ii} \sqsubseteq x_{ij} & \text{for } j \geq i, \text{ so} \\ \forall i & x_{ii} \sqsubseteq \bigsqcup_j x_{ij} & \text{and then} \\ & \bigsqcup_i x_{ii} \sqsubseteq \bigsqcup_i \bigsqcup_j x_{ij}. & \end{array}$$

□

Lemma 1.1.9. *A function $f : P_1 \times P_2 \rightarrow Q$ of cpos is continuous iff it is continuous in each argument separately.*

Proof: The monotonicity is clear. Assume f is continuous in each argument. Then

$$\begin{aligned} f(\bigsqcup_n \langle x_1^{(n)}, x_2^{(n)} \rangle) &\stackrel{def}{=} f(\langle \bigsqcup_n x_1^{(n)}, \bigsqcup_n x_2^{(n)} \rangle) \\ &= \bigsqcup_m f(\langle x_1^{(m)}, \bigsqcup_n x_2^{(n)} \rangle) \quad \text{as } f \text{ is cts. in its first argument} \\ &= \bigsqcup_m \bigsqcup_n f(\langle x_1^{(m)}, x_2^{(n)} \rangle) \quad \text{as } f \text{ is cts. in its second argument} \\ &= \bigsqcup_n f(\langle x_1^{(n)}, x_2^{(n)} \rangle) \quad \text{by lemma 1.1.8.} \end{aligned}$$

Now assume f is continuous. For any point $x_2 \in P_2$ we get

$$\begin{aligned} f(\langle \bigsqcup_n x_1^{(n)}, x_2 \rangle) &= f(\langle \bigsqcup x_1^{(n)}, \bigsqcup_n x_2 \rangle) \\ &= f(\bigsqcup_n \langle x_1^{(n)}, x_2 \rangle) && \text{by definition} \\ &= \bigsqcup_n f(\langle x_1^{(n)}, x_2 \rangle) && \text{because } f \text{ is cts.} \end{aligned}$$

Similarly for the second argument. □

Definition 1.1.10. A monotone partial function $f : P \rightarrow Q$ of cpos is continuous iff for any ω -chain $(x_n)_{n \in \mathbb{N}}$

$$f(\bigsqcup_n x_n) \downarrow \Leftrightarrow \exists m. f(x_m) \downarrow \wedge f(\bigsqcup_{n \geq m} x_n) = \bigsqcup_{n \geq m} f(x_n).$$

This can be expressed more elegantly:

Definition 1.1.11. A subset $A \subseteq P$ is upper closed iff, for all $x, y \in P$, if $x \in A$ and $x \sqsubseteq y$ then $y \in A$.

Definition 1.1.12. A subset $A \subseteq P$ is Scott-open if it is upper closed and, for any ω -chain $x_0 \sqsubseteq x_1 \sqsubseteq \dots$ in P , if $\bigsqcup_n x_n \in A$ then, for some n , $x_n \in A$.

Now we are able to give a more elegant definition of a partial continuous function:

Definition 1.1.13. A partial function $f : P \rightarrow Q$ is continuous iff it is monotone, $\text{dom}(f)$ is Scott-open and $f \upharpoonright \text{dom}(f)$ is (total) continuous.

Proposition 1.1.14. The Scott-open sets on a cpo P is a topology on P .

Proof:

- Clearly P and \emptyset are Scott-open.
- Let $\mathcal{O}_1, \mathcal{O}_2 \subseteq P$ be Scott-open. Clearly $\mathcal{O}_1 \cap \mathcal{O}_2$ is upper closed. Let $(x_i)_{i \in \mathbb{N}}$ be an ω -chain such that $\bigsqcup_i x_i \in \mathcal{O}_1 \cap \mathcal{O}_2$; then there must be i_1, i_2 such that $x_{i_1} \in \mathcal{O}_1$ and $x_{i_2} \in \mathcal{O}_2$. Let $k = \max(i_1, i_2)$. Then $x_k \in \mathcal{O}_1 \cap \mathcal{O}_2$ because both \mathcal{O}_1 and \mathcal{O}_2 are upper closed.
- Let $(\mathcal{O}_i)_i$ be a family of Scott-open sets in P . If $a \in \bigcup_i \mathcal{O}_i$ and $p \in P$ where $a \leq p$, then there must be some j such that $a \in \mathcal{O}_j$ and since \mathcal{O}_j is upper closed $p \in \mathcal{O}_j \subseteq \bigcup_i \mathcal{O}_i$. Assume $\bigsqcup_i x_i \in \bigcup_i \mathcal{O}_i$ then there must be a j such that $\bigsqcup_i x_i \in \mathcal{O}_j$. It follows that there must exist a k such that $x_k \in \mathcal{O}_j \subseteq \bigcup_i \mathcal{O}_i$.

□

Lemma 1.1.15. Let $\uparrow x$ be the set $\{p \in P \mid p \not\sqsubseteq x\}$. Then $\uparrow x$ is Scott-open for all $x \in P$.

Proof: Clearly $\uparrow x$ is upper closed. Suppose (x_i) is a chain with $\bigsqcup x_i \in \uparrow x$. If $x_i \notin \uparrow x$, for all i , we must have, by definition of $\uparrow x$, $\forall i. x_i \sqsubseteq x$. This means that x is an upper bound of (x_i) , but since $\bigsqcup x_i$ is the least upper bound, we have $\bigsqcup x_i \sqsubseteq x$, i.e., $\bigsqcup x_i \notin \uparrow x$. □

Proposition 1.1.16. Let $f : P \rightarrow Q$ be a partial function of cpos. Then f is continuous iff f is continuous with respect to the Scott-topology.

Proof: Suppose f is continuous and let $B \subseteq Q$ be a Scott-open set. If $x \in f^{-1}(B)$ and $x \sqsubseteq y$, then $f(x) \in B$ and $f(x) \sqsubseteq f(y)$ because f is monotone, so $f(y) \in B$ since B is upper closed; it follows that $y \in f^{-1}(B)$, so $f^{-1}(B)$ is upper closed. Now, let (x_i) be a chain in P with $\bigsqcup x_i \in f^{-1}(B)$, i.e., $f(\bigsqcup x_i) \in B$. Since f is continuous, there is an m such that $f(x_m) \downarrow$ and $f(\bigsqcup_{i \geq m} x_i) = \bigsqcup_{i \geq m} f(x_i)$. Because f is monotone, $f(x_m) \sqsubseteq f(x_{m+1}) \sqsubseteq \dots$ is a chain in Q and since $\bigsqcup_{i \geq m} f(x_i) \in B$, there is a k such that $f(x_k) \in B$, i.e., $x_k \in f^{-1}(B)$.

Now suppose that f is continuous w.r.t. the Scott-topology. Firstly we show that $\text{dom}(f)$ is Scott-open: Q is open and $\text{dom}(f) = f^{-1}(Q)$. Secondly we must show that f is monotone: Suppose $x \sqsubseteq y \in \text{dom}(f)$ and $f(x) \not\sqsubseteq f(y)$. We have $f(x) \in \uparrow f(y)$ while $f(y) \notin \uparrow f(y)$, so x is in the open set $f^{-1}(\uparrow f(y))$ while y is not, but this contradicts $x \sqsubseteq y$.

Finally let (x_i) be a chain in $\text{dom}(f)$. Now f is monotone, so $(f(x_i))$ is a chain in Q . Since $x_i \sqsubseteq \bigsqcup x_i$, for all i , also $f(x_i) \sqsubseteq f(\bigsqcup x_i)$, i.e., $f(\bigsqcup x_i)$ is an upper bound, so

$$\bigsqcup f(x_i) \sqsubseteq f(\bigsqcup x_i).$$

If $f(\bigsqcup x_i) \not\sqsubseteq \bigsqcup f(x_i)$ then $f(\bigsqcup x_i) \in \uparrow \bigsqcup f(x_i)$, so $\bigsqcup x_i \in f^{-1}(\uparrow \bigsqcup f(x_i))$, and since this is an open set, there is a j such that $x_j \in f^{-1}(\uparrow \bigsqcup f(x_i))$, i.e., $f(x_j) \in \uparrow \bigsqcup f(x_i)$, which is also an open set so $\bigsqcup f(x_i) \in \uparrow \bigsqcup f(x_i)$ because $f(x_j) \sqsubseteq \bigsqcup f(x_i)$. But this is a contradiction, so we must have

$$f(\bigsqcup x_i) \sqsubseteq \bigsqcup f(x_i).$$

□

Corollary 1.1.17. *Composition of partial continuous functions is continuous.*

Proof: Composition of continuous functions on a topological space is continuous. □

1.2 Monads

Definition 1.2.1. *Let $T : \mathcal{D} \rightarrow \mathcal{D}$ be a functor, and $\eta : id_{\mathcal{D}} \Rightarrow T$ and $\mu : T \Rightarrow T$ natural transformations. Then the triple $(T, \mu, \eta) = \mathbb{T}$ is a monad iff the following diagrams commute*

$$\begin{array}{ccc} T & \xrightarrow{T\mu} & T \\ \mu T \downarrow & & \downarrow \mu \\ T & \xrightarrow{\mu} & T \end{array} \qquad \begin{array}{ccc} T & \xrightarrow{\eta T} & T & \xleftarrow{T\eta} & T \\ & \searrow id_T & \downarrow \mu & \swarrow id_T & \\ & & T & & \end{array}$$

Here $(T\mu)_D = T(\mu_D)$ and $(\mu T)_D = \mu_{TD}$.

Given an adjunction $(F, G, \varepsilon, \eta) : \mathcal{C} \xrightleftharpoons[F]{F} \mathcal{D}$, $F \dashv G$, we can construct the *canonical* monad (T, μ, η) as follows: $T = GF : \mathcal{D} \rightarrow \mathcal{D}$, η is the unit of the adjunction, and the components of μ are $\mu_D = G(\varepsilon_{FD}) : TD \rightarrow TD$, where $\varepsilon : FG \Rightarrow id_{\mathcal{C}}$ is the counit of the adjunction. To see that the canonical monad is in fact a monad, consider the following, for each object D :

$$\begin{aligned} \mu_D \circ (T\mu)_D &= \mu_D \circ (\mu T)_D \\ \Leftrightarrow G(\varepsilon_{FD}) \circ T(\mu_D) &= G(\varepsilon_{FD}) \circ \mu_{TD} \\ \Leftrightarrow G(\varepsilon_{FD}) \circ GFG(\varepsilon_{FD}) &= G(\varepsilon_{FD}) \circ G(\varepsilon_{FGFD}) \\ \Leftrightarrow G(\varepsilon_{FD} \circ FG(\varepsilon_{FD})) &= G(\varepsilon_{FD} \circ \varepsilon_{FGFD}). \end{aligned}$$

To see that $\varepsilon_{FD} \circ FG(\varepsilon_{FD}) = \varepsilon_{FD} \circ \varepsilon_{FGFD}$ consider the naturality square:

$$\begin{array}{ccc} FG(FGFD) & \xrightarrow{\varepsilon_{FGFD}} & FGFD \\ FG(\varepsilon_{FD}) \downarrow & & \downarrow \varepsilon_{FD} \\ FG(FD) & \xrightarrow{\varepsilon_{FD}} & FD \end{array} .$$

It commutes because ε is a natural transformation. Because F and G are adjoints, we have the following equality for each object C :

$$(G\varepsilon)_C \circ (\eta G)_C = id_{GC}. \quad (1.1)$$

So we get

$$\begin{aligned} \mu_D \circ (\eta T)_D &= id_{TD} \\ \Leftrightarrow G(\varepsilon_{FD}) \circ \eta_{GFD} &= id_{GFD} \end{aligned}$$

which holds by 1.1 with $C = FD$. Along the same lines we have

$$(\varepsilon F)_D \circ (F\eta)_D = id_{FD} \quad (1.2)$$

for all objects D of \mathcal{D} , so

$$\begin{aligned} \mu_D \circ (T\eta)_D &= id_{TD} \\ \Leftrightarrow G(\varepsilon_{FD}) \circ GF(\eta_D) &= id_{GFD} \\ \Leftrightarrow G(\varepsilon_{FD} \circ F(\eta_D)) &= G(id_{FD}), \end{aligned}$$

which holds by 1.2.

Dually we have the notion of a comonad, and for any adjunction $(F, G, \varepsilon, \eta) : \mathcal{C} \xrightleftharpoons[G]{F} \mathcal{D}$, $F \dashv G$, we can construct its *canonical* comonad on \mathcal{C} , as we shall see in detail in Section 2.4.

Chapter 2

Useful constructions

2.1 The lifting functor

In the following sections we will study the relations between and closure properties of four particular categories, which we define here.

Cpo (category of predomains) consists of complete partial orders and continuous functions.

Cppo (category of domains) consists of pointed complete partial orders (a pointed cpo is a cpo with a least element) and continuous functions.

Cppo_⊥ (category of domains and strict continuous functions) consists of pointed cpos and strict continuous functions, i.e. continuous functions that preserve the least element.

pCpo (category of predomains and partial continuous functions) consists of cpos and partial continuous functions.

These are well-defined categories as both continuous and partial continuous functions compose (and the identity function is the unit for composition in both cases).

Consider the categories Cpo of complete partial orders and continuous functions and Cppo_⊥ of pointed complete partial orders and strict continuous functions. Obviously, there is an inclusion functor

$$U : \text{Cppo}_{\perp} \rightarrow \text{Cpo}.$$

We claim that U has a left adjoint

$$F : \text{Cpo} \rightarrow \text{Cppo}_{\perp},$$

which, given a cpo X (i.e., an object of Cpo) adds to it a bottom element \perp_X . Given an arrow $f : X \rightarrow Y$ in Cpo

$$F(f) = \begin{cases} f & \text{on } X \\ \perp_Y & \text{on } \perp_X \end{cases}$$

is the arrow that sends the added bottom element of X to the added bottom of Y and is f on the rest. Thus $F(f)$ is a strict continuous function. The adjunction is given by the natural bijection:

$$\text{Cppo}_{\perp}(FX, Y) \xrightarrow{m_{X,Y}} \text{Cpo}(X, UY).$$

which is defined as follows: Given $\alpha : FX \rightarrow Y$

$$m_{X,Y}(\alpha) = \alpha \upharpoonright X.$$

Given $\beta : X \rightarrow UY$

$$m_{X,Y}^{-1}(\beta)$$

maps the added bottom \perp_X to the bottom element of Y , which exists because Y is in Cppo_\perp . Given this thorough description, the naturality should be easy to verify. The unit of the adjunction

$$\eta_X : X \rightarrow UF(X)$$

is just the inclusion. The counit

$$\varepsilon_Y : FU(Y) \rightarrow Y$$

is the identity on Y and maps \perp_Y to the least element of Y . The functor

$$T = UF : \text{Cpo} \rightarrow \text{Cpo},$$

which is actually just the functor F considered as a functor on Cpo instead of Cppo_\perp , is called the lifting functor, and since it arises from an adjunction, it extends to a monad as described in section 1.2.

Indeed, if we take η to be the unit of the adjunction, and the components of μ to be

$$T^2X \xrightarrow{U(\varepsilon_{FX})} TX,$$

then T^2X is X with two bottom elements added (the second one below the first one), and μ_X is then the identity on TX and it maps the second bottom \perp_{TX} to the first bottom \perp_X . Then $\mathbb{T} = (T, \mu, \eta)$ is the canonical monad of the adjunction $(F, U, \varepsilon, \eta)$.

2.2 The category of algebras for a monad

We show now that Cppo_\perp is the category of algebras for the monad \mathbb{T} . The category $\text{Cpo}^{\mathbb{T}}$ of \mathbb{T} -algebras for the monad \mathbb{T} on Cpo is defined as follows:

- *Objects* are pairs (X, h) where X is an object of Cpo and $h : TX \rightarrow X$ is a map in Cpo (i.e., a continuous function) such that

$$\begin{array}{ccc} T^2X & \xrightarrow{T(h)} & TX \\ \mu_X \downarrow & & \downarrow h \\ TX & \xrightarrow{h} & X \end{array} \quad \begin{array}{ccc} X & \xrightarrow{\eta_X} & TX \\ & \searrow id_X & \downarrow h \\ & & X \end{array}$$

commutes.

- *Morphisms* $(X, h) \rightarrow (Y, k)$ are morphisms $X \xrightarrow{f} Y$ in Cpo such that

$$\begin{array}{ccc} TX & \xrightarrow{T(f)} & TY \\ h \downarrow & & \downarrow k \\ X & \xrightarrow{f} & Y \end{array}$$

commutes.

Proposition 2.2.1. $\text{Cpo}^{\mathbb{T}} \cong \text{Cppo}_{\perp}$, meaning that the category of algebras for the monad \mathbb{T} is isomorphic to Cppo_{\perp} , considered as objects in the category of locally small categories.

Proof: For all objects (X, h) in $\text{Cpo}^{\mathbb{T}}$ we have

$$h\eta_X = id_X$$

so h is epi, i.e., surjective in Cpo . h is monotone, since h is in Cpo , and therefore $h(\perp_X)$ must be a least element of X . So X is an element of Cppo_{\perp} . Since η is the inclusion $X \rightarrow TX$ we must have

$$h \upharpoonright X = id_X.$$

This determines h uniquely. Actually, h does exactly the same as the counit ε except that they belong to different categories. We saw that X has a least element, so $X = UY$ for some object Y of Cppo_{\perp} . We realize that for the object (UY, h) we have

$$h = U(\varepsilon_Y) : UFUY \rightarrow UY.$$

For morphisms $(X, h) \rightarrow (Y, k)$ in $\text{Cpo}^{\mathbb{T}}$ we have the commuting square above. We have just seen that the least element of X is $h(\perp_X)$ and that of Y is $k(\perp_Y)$. By the definition of T we have $T(f)(\perp_X) = \perp_Y$, so

$$k(\perp_Y) = k(T(f)(\perp_X)) = f(h(\perp_X)),$$

that is, f preserves the least element, so f is in fact a morphism in Cppo_{\perp} . Define a functor

$$S : \text{Cpo}^{\mathbb{T}} \rightarrow \text{Cppo}_{\perp}$$

which is the identity on arrows, and sends an object (X, h) to X . The composition is clear, and also $id_{(X, h)} = id_X$, so S is a functor. S has an inverse functor S^{-1} with $S^{-1}(Y) = (UY, U(\varepsilon_Y))$, which is also the identity on arrows. This shows that $\text{Cpo}^{\mathbb{T}} \cong \text{Cppo}_{\perp}$, considered as objects in the category of locally small categories. \square

2.3 The Kleisli category of a monad

Given an adjunction $(F, G, \varepsilon, \eta) : \mathcal{C} \xrightleftharpoons[G]{F} \mathcal{D}$, $F \dashv G$ we have seen that we can construct a monad $(GF, G\varepsilon F, \eta)$ on \mathcal{D} which we call *the canonical monad*. Having seen that every adjunction gives rise to a monad by this construction, two natural question arises: given a monad (T, μ, η) , is there an underlying adjunction, i.e., an adjunction from which it could be constructed, and if there is, is it uniquely determined? The answer to the first question is yes, as we shall see in a moment, when we construct the Kleisli category. The answer to the second question is no, so as true categorists we form the category of answers to our question, that is, the category $\mathbb{T}\text{-Adj}$ of adjunctions that gives rise to the monad (T, μ, η) .

The category $\mathbb{T}\text{-Adj}$ is defined as follows:

Objects are triples (\mathcal{C}, F, G) , where \mathcal{C} is a category, and F and G are functors $\mathcal{C} \xrightleftharpoons[G]{F} \mathcal{D}$, part of an adjunction $(F, G, \varepsilon, \eta)$ ¹ such that $F \dashv G$, $GF = T$ and $G\varepsilon F = \mu$.

¹Note that the fourth component must be η .

Morphisms are functors. Given objects (\mathcal{C}, F, G) and (\mathcal{C}', F', G') a morphism K is a functor $\mathcal{C} \rightarrow \mathcal{C}'$ satisfying $KF = F', G'K = G$, i.e. the diagram

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{K} & \mathcal{C}' \\ \swarrow F & & \searrow G' \\ & \mathcal{D} & \\ \nwarrow G & & \nearrow F' \end{array}$$

almost commutes.

Composition, identities and associativity of the former are obvious.

Given a monad \mathbb{T} on a category \mathcal{D} , the *Kleisli category* $\mathcal{D}_{\mathbb{T}}$ of \mathbb{T} is defined [Oos], as the initial object in $\mathbb{T}\text{-Adj}$. The category $\mathbb{T}\text{-Alg}$ of \mathbb{T} -algebras is the terminal object. Dual to the Kleisli category we have the coKleisli category for the comonad. We will treat this dual case in detail in the next section, so for now we simply describe the structure of the Kleisli category without any proofs of its properties.

The Kleisli category $\mathcal{D}_{\mathbb{T}}$ has the same objects as \mathcal{D} . A map in $\mathcal{D}_{\mathbb{T}}$ from $X \rightarrow Y$ is an arrow $X \rightarrow T(Y)$ in \mathcal{D} . This makes composition a bit unusual: Given $X \xrightarrow{f} TY$ and $Y \xrightarrow{g} TZ$ in \mathcal{D} , f is considered an arrow from $X \rightarrow Y$ in $\mathcal{D}_{\mathbb{T}}$ and g goes from $Y \rightarrow Z$ in $\mathcal{D}_{\mathbb{T}}$, so in $\mathcal{D}_{\mathbb{T}}$ they are composable. The composition gf in $\mathcal{D}_{\mathbb{T}}$ is the composite

$$X \xrightarrow{f} TY \xrightarrow{T(g)} T^2Z \xrightarrow{\mu_Z} TZ$$

in \mathcal{D} . The identity is

$$X \xrightarrow{id_X} X \xrightarrow{\eta_X} TX.$$

For our monad \mathbb{T} on Cpo the Kleisli category $\text{Cpo}_{\mathbb{T}}$ has the same objects as Cpo and an arrow $X \rightarrow Y$ in $\text{Cpo}_{\mathbb{T}}$ is an arrow $X \rightarrow TY$ in Cpo , i.e., an arrow from the domain X to the codomain Y with an added bottom element. The adjunction $\text{Cpo}_{\mathbb{T}} \xrightleftharpoons[G_{\mathbb{T}}]{F_{\mathbb{T}}} \text{Cpo}$, $F_{\mathbb{T}} \dashv G_{\mathbb{T}}$

is defined as follows: $G_{\mathbb{T}}$ sends the object X to TX and arrows $X \xrightarrow{f} Y$ ($X \xrightarrow{f} TY$ in Cpo) to

$$TX \xrightarrow{Tf} T^2Y \xrightarrow{\mu_Y} TY$$

Functor $F_{\mathbb{T}}$ is the identity on objects and it sends $X \xrightarrow{f} Y$ to $X \xrightarrow{f} Y \xrightarrow{\eta_Y} TY$ considered as $X \xrightarrow{f} Y$ in $\text{Cpo}_{\mathbb{T}}$. This description of the Kleisli category with corresponding adjunction is found in [Oos, p. 60].

Proposition 2.3.1. $\text{Cpo}_{\mathbb{T}} \cong \text{pCpo}$

Proof: The idea is to let the added bottom \perp_X correspond to the undefined elements of a partial function. We define a functor

$$V : \text{Cpo}_{\mathbb{T}} \rightarrow \text{pCpo}$$

as follows: V is the identity on objects. Let $X \xrightarrow{f} Y$ be an arrow in $\text{Cpo}_{\mathbb{T}}$, i.e., f is an arrow from $X \rightarrow TY$ in Cpo , and let $\gamma_Y : TY \rightarrow Y$ be the partial function which is the identity on Y and undefined for \perp_Y . Then

$$V(f) = \gamma_Y \circ f.$$

$V(f)$ is indeed a partial continuous function, since both γ_Y and f are so. V is a functor: $V(id_X) = id_X$ since $id_X(a) = a \neq \perp_X$ because \perp_X is not an element of X . Recall that the composition of two arrows $X \xrightarrow{f} Y$ and $Y \xrightarrow{g} Z$ in $\text{Cpo}_{\mathbb{T}}$ is the composite

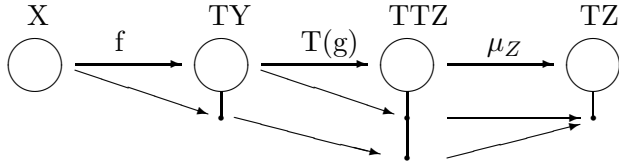
$$X \xrightarrow{f} TY \xrightarrow{T(g)} T^2Z \xrightarrow{\mu_Z} TZ.$$

To see that $V(gf) = V(g)V(f)$ we need to convince ourselves that the elements of X that are sent to \perp_Z by the composite gf , i.e. $(gf)^{-1}(\perp_Z)$ are exactly the elements of X where $V(g)V(f)$ is left undefined. The latter is seen to be

$$f^{-1}(\perp_Y) \cup f^{-1}(g^{-1}(\perp_Z))$$

(If either one of the partial functions $V(f)$ or $V(g)$ is undefined for an element a then their composite is undefined for this element.) Now, the composite gf is the arrow $\mu_Z T(g)f$. We calculate:

$$\begin{aligned} & (\mu_Z T(g)f)^{-1}(\perp_Z) \\ &= f^{-1}(T(g)^{-1}(\mu_Z^{-1}(\perp_Z))) \\ &= f^{-1}(T(g)^{-1}(\{\perp_Z, \perp_{TZ}\})) \\ &= f^{-1}(g^{-1}(\perp_Z) \cup \{\perp_Y\}) \end{aligned}$$

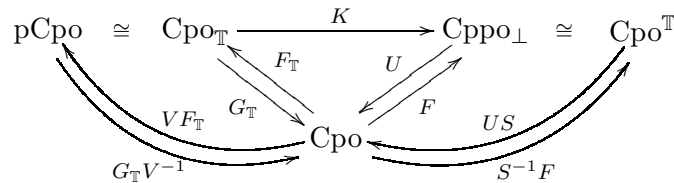


Functor V has an inverse $V^{-1} : \text{pCpo} \rightarrow \text{Cpo}_{\mathbb{T}}$. Definition on arrows $f : X \rightarrow Y$ is

$$V^{-1}(f) = \begin{cases} f & \text{on } \text{dom}(f) \\ \perp_Y & \text{on } X \setminus \text{dom}(f) \end{cases}$$

To see that V^{-1} is well-defined, i.e., $V^{-1}(f)$ is continuous, note that for a total, continuous function g and a chain (x_i) , $g(\bigsqcup_{i \geq k} x_i) = \bigsqcup_{i \geq k} g(x_i)$, for some k , implies $g(\bigsqcup_i x_i) = \bigsqcup_i g(x_i)$. \square

We now have a nice picture of how some of the categories are connected by adjunctions:



For some unique K Later we shall realize that there is an isomorphism $\mathbf{pCpo} \cong \mathbf{Cppo}_\perp$ which brings this figure to collapse; meaning that the category $T\text{-Adj} \simeq \mathbf{1}$. This tells us that the adjunction giving rise to the monad (\mathbb{T}, μ, η) on \mathbf{Cpo} is uniquely (up to isomorphism) determined.

2.4 The CoKleisli Category of a Comonad

It turns out to be handy to know the exact structure of the coKleisli category of a comonad. Finding this involves many categorical concepts and definitions, and is thus a good exercise and an instructive example at the same time. In the spirit of quest for experience and knowledge, this section contains a detailed treatment of the coKleisli category of a comonad.

The coKleisli category of a comonad is the dualization of the concept of a Kleisli category of a monad. The following treatment can equally be dualized.

2.4.1 Motivation

Given a category \mathcal{C} , there is a notion of a comonad on this category, being an endofunctor with certain properties. These comonads are apparently important. It turns out that whenever there is an adjunction between \mathcal{C} and some other category, this adjunction gives rise to a comonad on \mathcal{C} . Conversely every comonad comes from an adjunction, but perhaps from more than one. And this connection between comonads on \mathcal{C} and adjunctions from \mathcal{C} is the story of the coKleisli category. The coKleisli category of a comonad \mathbb{L} is the 'smallest' solution to the problem "what adjunctions give rise to \mathbb{L} ". Smallest in the sense that the adjunctions giving rise to \mathbb{L} form a category, and the coKleisli category of \mathbb{L} is the initial object in this category.

2.4.2 Definitions

Given a category \mathcal{C} , a comonad \mathbb{L} on \mathcal{C} is a triple (L, δ, ε) , where

- L is a functor $\mathcal{C} \rightarrow \mathcal{C}$
- δ is a natural transformation $L \Rightarrow L^2$
- ε is a natural transformation $L \Rightarrow id_{\mathcal{C}}$

such that

$$\begin{array}{ccc}
 L & \xrightarrow{\delta} & L^2 \\
 \delta \downarrow & & \downarrow L\delta \\
 L^2 & \xrightarrow{\delta L} & L^3
 \end{array}
 \quad \text{and} \quad
 \begin{array}{ccc}
 & L & \\
 id_L \swarrow & \downarrow \delta & \searrow id_L \\
 L & \xleftarrow{\varepsilon L} L^2 \xrightarrow{L\varepsilon} & L
 \end{array}$$

commute.

Proposition 2.4.1. *Given an adjunction $(F, G, \varepsilon, \eta)$:*

$$\mathcal{C} \xrightleftharpoons[G]{F} \mathcal{D} \quad F \dashv G,$$

the triple $(FG, \delta = F\eta G, \varepsilon)$ is a comonad on \mathcal{C} .

Proof: Clearly FG is a functor from \mathcal{C} to \mathcal{C} , and ϵ is known to be a natural transformation from FG to $id_{\mathcal{C}}$ ². Similarly η is a natural transformation from $id_{\mathcal{C}}$ to GF , $X \xrightarrow{\eta_X} GF(X)$. Thus $\eta_{GX}: GX \rightarrow GF(GX)$, and $F\eta_{GX}: FGX \rightarrow FGF(GX)$, is a natural transformation from L to L . That we actually obtain a natural transformation follows from an adequately rich functor theory³. \square

In light of this, the category $\mathbb{L}\text{-Adj}$ of adjunctions giving rise to \mathbb{L} are defined as follows:

Objects are triples (\mathcal{D}, F, G) , where \mathcal{D} is a category, and F and G are functors $\mathcal{C} \xrightleftharpoons[G]{F} \mathcal{D}$, part of an adjunction (F, G, ϵ, η) ⁴ so that $F \dashv G$, $FG = L$ and $F\eta G = \delta$.

Morphisms are functors. Given objects (\mathcal{D}, F, G) and (\mathcal{D}', F', G') a morphism K is a functor $\mathcal{D} \rightarrow \mathcal{D}'$ such that $F'K = F$ and $KG' = G$, i.e., the diagram

$$\begin{array}{ccc} \mathcal{D} & \xrightarrow{K} & \mathcal{D}' \\ \swarrow G & & \nearrow F' \\ & \mathcal{C} & \\ \searrow F & & \swarrow G' \end{array}$$

mostly commutes.

Composition, identities and associativity of the former are obvious.

Given a comonad \mathbb{L} on a category \mathcal{C} , the coKleisli category $\mathcal{C}_{\mathbb{L}}$ of \mathbb{L} is defined as the initial object of $\mathbb{L}\text{-Adj}$.

2.4.3 Construction

As an object of $\mathbb{L}\text{-Adj}$, the coKleisli category of \mathbb{L} is a triple $(\mathcal{C}_{\mathbb{L}}, F_{\mathbb{L}}, G_{\mathbb{L}})$. We will not, as the title adverts, construct this object as much as define it, later to prove the desired properties (including well-definedness).

Definition 2.4.2 ($\mathcal{C}_{\mathbb{L}}$). *The objects of $\mathcal{C}_{\mathbb{L}}$ are the objects of \mathcal{C} . A morphism $X \xrightarrow{\bar{f}} Y$ in $\mathcal{C}_{\mathbb{L}}$ is a morphism $LX \xrightarrow{f} Y$ in \mathcal{C} , i.e. $\mathcal{C}_{\mathbb{L}}(X, Y) \stackrel{def}{=} \mathcal{C}(LX, Y)$. Given morphisms $X \xrightarrow{\bar{f}} Y$ and $Y \xrightarrow{\bar{g}} Z$, the composite $X \xrightarrow{\bar{g} \circ \bar{f}} Z$ is found as $\overline{g \circ Lf \circ \delta_X}$. The following diagrams might help to illustrate:*

$$\begin{array}{ccc} \text{in } \mathcal{C}_{\mathbb{L}} & & \text{in } \mathcal{C} \\ X \xrightarrow{f} Y \xrightarrow{g} Z & & \begin{array}{ccc} LX & \xrightarrow{f} & Y \\ \delta_X \downarrow & & \\ L^2X & \xrightarrow{L_f} & LY \xrightarrow{g} Z \end{array} \end{array}$$

Definition 2.4.3 ($F_{\mathbb{L}}$). *The functor $F_{\mathbb{L}}: \mathcal{C}_{\mathbb{L}} \rightarrow \mathcal{C}$ is defined as follows:*

²See [Oos, p. 45].

³One which has the stated claim as a theorem.

⁴Note that the third component must be ϵ .

Objects $F_{\mathbb{L}}(X) = LX$

Morphisms $F_{\mathbb{L}}(X \xrightarrow{\bar{f}} Y) = Lf \circ \delta_X$

$$LX \xrightarrow{f} Y \quad \mapsto \quad LX \xrightarrow{\delta_X} L^2X \xrightarrow{Lf} LY$$

Definition 2.4.4 ($G_{\mathbb{L}}$). *The functor $G_{\mathbb{L}}: \mathcal{C} \rightarrow \mathcal{C}_{\mathbb{L}}$ is defined as follows:*

Objects $G_{\mathbb{L}}(X) = X$

Morphisms $G_{\mathbb{L}}(X \xrightarrow{f} Y) = \overline{f \circ \varepsilon_X}$

$$LX \xrightarrow{\varepsilon_X} X \xrightarrow{f} Y.$$

2.4.4 Verification

Several aspects must be verified. Firstly, we must check that $\mathcal{C}_{\mathbb{L}}$ is a category – that composition is associative, and identities exists. Then we must show that $(\mathcal{C}_{\mathbb{L}}, F_{\mathbb{L}}, G_{\mathbb{L}})$ is an object of $\mathbb{L}\text{-Adj}$ – that $F_{\mathbb{L}} \dashv G_{\mathbb{L}}$ with suitable components, and that $F_{\mathbb{L}}G_{\mathbb{L}} = L$. Finally we must show that $(\mathcal{C}_{\mathbb{L}}, F_{\mathbb{L}}, G_{\mathbb{L}})$ is initial in $\mathbb{L}\text{-Adj}$.

Proposition 2.4.5. *$\mathcal{C}_{\mathbb{L}}$ is a category*

Proof: We must that for any three morphisms $X \xrightarrow{\bar{f}} Y \xrightarrow{\bar{g}} Z \xrightarrow{\bar{h}} W$, $(\bar{h} \circ \bar{g}) \circ \bar{f} = \bar{h} \circ (\bar{g} \circ \bar{f})$. We simply calculate both sides:

$$\begin{aligned} (\bar{h} \circ \bar{g}) \circ \bar{f} &= \overline{\bar{h} \circ Lg \circ \delta_Y \circ \bar{f}} \\ &= \overline{\bar{h} \circ Lg \circ \delta_Y \circ Lf \circ \delta_X} \end{aligned}$$

$$\begin{aligned} \bar{h} \circ (\bar{g} \circ \bar{f}) &= \overline{\bar{h} \circ \overline{g \circ Lf \circ \delta_X}} \\ &= \overline{\bar{h} \circ L(g \circ Lf \circ \delta_X) \circ \delta_X} \\ &= \overline{\bar{h} \circ Lg \circ L^2f \circ L\delta_X \circ \delta_X} \end{aligned}$$

The naturality square for δ

$$\begin{array}{ccc} LX & & L^2X \xrightarrow{\delta_{LX}} L^3X \\ f \downarrow & & \downarrow L^2f \\ Y & & LY \xrightarrow{\delta_Y} L^2Y \end{array}$$

gives $\delta_Y \circ Lf = L^2f \circ \delta_{LX}$, making the left hand side almost equal to the right hand side. If only $\delta_{LX} = L\delta_X \dots$. The first defining diagram for δ

$$\begin{array}{ccc} LX & \xrightarrow{\delta_X} & L^2X \\ \delta_X \downarrow & & \downarrow L\delta_X \\ L^X & \xrightarrow{\delta_{LX}} & L^3X \end{array}$$

gives us not exactly that, but something good enough in our case.

We must also find, for each object X , an identity \bar{i} on X , being a morphism $LX \xrightarrow{i} X$ in \mathcal{C} with \bar{i} neutral to composition in \mathcal{C}_L . The obvious choice is to put $i = \varepsilon_X$. So let us investigate whether $id_X = \overline{\varepsilon_X}$ works:

For $X \xrightarrow{\bar{f}} Y$, we get

$$\bar{f} \circ \overline{\varepsilon_X} = \overline{f \circ L\varepsilon_X \circ \delta_X}$$

The (second half of the) second defining diagram for ε and δ gives

$$\varepsilon_{LX} \circ \delta_X = id_{LX} \qquad \begin{array}{ccc} & LX & \\ id_{LX} \swarrow & \downarrow \delta_X & \searrow id_{LX} \\ LX & \xleftarrow{\varepsilon_{LX}} L^2X \xrightarrow{L\varepsilon_X} & LX \end{array} \qquad L\varepsilon_X \circ \delta_X = id_{LX}$$

making everything work out nicely.

For $W \xrightarrow{\bar{g}} X$, we get

$$\overline{\varepsilon_X} \circ \bar{g} = \overline{\varepsilon_X \circ Lg \circ \delta_W}$$

The naturality square for ε gives

$$\begin{array}{ccc} LW & \xleftarrow{\varepsilon_{LW}} & L^2W \\ g \downarrow & & \downarrow Lg \\ X & \xleftarrow{\varepsilon_X} & LX \end{array} \qquad g \circ \varepsilon_{LW} = \varepsilon_X \circ Lg$$

Combining this with the first half of the second defining diagram, we get

$$\begin{aligned} \overline{\varepsilon_X} \circ \bar{g} &= \overline{\varepsilon_X \circ Lg \circ \delta_W} \\ &= \overline{g \circ \varepsilon_{LW} \circ \delta_W} \\ &= \overline{g \circ id_{LW}} \\ &= \bar{g} \end{aligned}$$

□

Proposition 2.4.6. $F_L \dashv G_L$

Proof: We should as the first thing show that F_L and G_L are indeed functors. Given the calculations above, it is easy to see that

$$F_L(id_Y) = F_L(\overline{\varepsilon_Y}) = L\varepsilon_Y \circ \delta_Y = id_{LY} = id_{F_L Y}$$

and

$$G_L(id_X) = \overline{id_X \circ \varepsilon_X} = \overline{\varepsilon_X} = id_{G_L X}.$$

Preservation of compositions is almost as easy. Let $X \xrightarrow{\bar{f}} Y$ and $Y \xrightarrow{\bar{g}} Z$ be given. As δ is a natural transformation from L to L^2 , we have that $\delta_Y \circ Lf = L^2f \circ \delta_X$, and thus

$$\begin{aligned} F_L(\bar{g} \circ \bar{f}) &= F_L(\overline{g \circ Lf \circ \delta_X}) \\ &= L(g \circ Lf \circ \delta_X) \circ \delta_X \\ &= Lg \circ (L^2f \circ L\delta_X) \circ \delta_X \\ &= Lg \circ \delta_Y \circ Lf \circ \delta_X \\ &= F_L(\bar{g}) \circ F_L(\bar{f}). \end{aligned}$$

Similarly, given $X \xrightarrow{f} Y$ and $Y \xrightarrow{g} Z$ we find, remembering that $\varepsilon_Y \circ Lf = f \circ \varepsilon_X$,

$$\begin{aligned} G_{\mathbb{L}}(g) \circ G_{\mathbb{L}}(f) &= \overline{g \circ \varepsilon_Y \circ f \circ \varepsilon_X} \\ &= \overline{g \circ \varepsilon_Y \circ L(f \circ \varepsilon_X) \circ \delta_X} \\ &= \overline{g \circ (\varepsilon_Y \circ Lf) \circ L\varepsilon_X \circ \delta_X} \\ &= \overline{g \circ f \circ \varepsilon_X \circ L\varepsilon_X \circ \delta_X} \\ &= \overline{g \circ f \circ \varepsilon_X} \\ &= G_{\mathbb{L}}(g \circ f). \end{aligned}$$

So they are functors alright; now for the adjunction. . .

For $C \in \mathcal{C}_0$ and $D \in \mathcal{C}_{\mathbb{L}0}$, we must find a natural bijection between the Hom sets $\mathcal{C}(F_{\mathbb{L}}D, C)$ and $\mathcal{C}_{\mathbb{L}}(D, G_{\mathbb{L}}C)$. These can be rewritten as $\mathcal{C}(LD, C)$ and $\mathcal{C}_{\mathbb{L}}(D, C)$, and the natural guess becomes $f: LD \rightarrow C \mapsto \bar{f}: D \rightarrow C$. This is a bijection, but is it natural?

The naturality requirement states that for $D \xrightarrow{\bar{u}} D'$ and $C' \xrightarrow{v} C$ and any $F_{\mathbb{L}}D' \xrightarrow{\alpha} C'$, we must have

$$\overline{v \circ \alpha \circ F_{\mathbb{L}}\bar{u}} = G_{\mathbb{L}}v \circ \bar{\alpha} \circ \bar{u},$$

which translates to

$$\overline{v \circ \alpha \circ Lu \circ \delta_D} = \overline{v \circ \varepsilon_{C'} \circ \bar{\alpha} \circ \bar{u}} = \overline{v \circ \varepsilon_{C'} \circ L\alpha \circ \delta_{D'} \circ Lu \circ \delta_D}.$$

Now, if only $\alpha = \varepsilon_{C'} \circ L\alpha \circ \delta_{D'}$, we would be done. . . But that follows from the commutativity of

$$\begin{array}{ccc} & LD' & \\ id_{LD'} \swarrow & \downarrow \delta_{D'} & \\ LD' & \xleftarrow{\varepsilon_{LD}} & L^2D \\ \alpha \downarrow & & \downarrow L\alpha \\ C' & \xleftarrow{\varepsilon_{C'}} & LC' \end{array}$$

The top triangle is the first half of the second defining diagram, and the bottom square is a naturality square for ε . \square

Proposition 2.4.7. *The components of $F_{\mathbb{L}} \dashv G_{\mathbb{L}}$ matches L : The counit is ε and the unit η satisfies $F\eta G = \delta$.*

Proof: The components of the adjunction are given through the bijection $\overline{(\cdot)}$'s action on identities. We would like

$$\overline{\varepsilon_C} = id_{G_{\mathbb{L}}C}$$

and

$$F_{\mathbb{L}}(\overline{id_{F_{\mathbb{L}}G_{\mathbb{L}}D}}) = \delta_D$$

for all C and D in \mathcal{C} .

The first equation is easily verified, as

$$id_{G_{\mathbb{L}}C} = id_C = \overline{\varepsilon_C},$$

and the second unravels just as beautifully:

$$F_{\mathbb{L}}(\overline{id_{F_{\mathbb{L}}G_{\mathbb{L}}D}}) = F_{\mathbb{L}}(\overline{id_{LD}}) = L(id_{LD}) \circ \delta_D = \delta_D$$

□

Proposition 2.4.8. $F_{\mathbb{L}}G_{\mathbb{L}} = L$

Proof: On objects we have $F_{\mathbb{L}}G_{\mathbb{L}}X = F_{\mathbb{L}}X = LX$. On morphisms we get

$$F_{\mathbb{L}}G_{\mathbb{L}}f = F_{\mathbb{L}}\overline{f \circ \varepsilon_X} = L(f \circ \varepsilon_X) \circ \delta_X = Lf \circ L\varepsilon_X \circ \delta_X = Lf \circ id_{LX} = Lf.$$

□

Proposition 2.4.9. $(\mathcal{C}_{\mathbb{L}}, F_{\mathbb{L}}, G_{\mathbb{L}})$ is initial in $\mathbb{L}\text{-Adj}$.

Proof: For $(\mathcal{C}_{\mathbb{L}}, F_{\mathbb{L}}, G_{\mathbb{L}})$ to be an initial object, we must find a unique morphism into any other object of $\mathbb{L}\text{-Adj}$. So given $(\mathcal{D}, F, G) \in \mathbb{L}\text{-Adj}_0$, we must find a unique functor $K_{\mathcal{D}}: \mathcal{C}_{\mathbb{L}} \rightarrow \mathcal{D}$ so that $FK_{\mathcal{D}} = F_{\mathbb{L}}$ and $K_{\mathcal{D}}G_{\mathbb{L}} = G$.

First we construct a $K_{\mathcal{D}}$ that works:

$$\begin{array}{ccc} \mathcal{C}_{\mathbb{L}} & \xrightarrow{K} & \mathcal{D} \\ & \swarrow G_{\mathbb{L}} & \searrow F \\ & \mathcal{C} & \swarrow G \\ & \nwarrow F_{\mathbb{L}} & \end{array}$$

Objects $K_{\mathcal{D}}X = GX$

Morphisms $K_{\mathcal{D}}\overline{f} = mf$

where mf denotes the transpose of f under the adjunction $F \dashv G$:

$$FGX = LX \xrightarrow{f} Y \quad \mapsto \quad GX \xrightarrow{mf} GY.$$

This satisfies the equations on objects:

$$FK_{\mathcal{D}}X = FGX = LX = F_{\mathbb{L}}X \quad \text{and} \quad K_{\mathcal{D}}G_{\mathbb{L}}X = K_{\mathcal{D}}X = GX.$$

On morphisms the equations turn out a little trickier: For a morphism $X \xrightarrow{\overline{f}} Y$ in $\mathcal{C}_{\mathbb{L}}$, $FK_{\mathcal{D}} = F_{\mathbb{L}}$ becomes

$$F(mf) = Lf \circ \delta_X.$$

To show this, we pull, out of the big black magicians hat, the following marvelous string of equations:

$$F(mf) = F(Gf \circ m(id_{LX})) = FGf \circ F(m(id_{FGX})) = Lf \circ F\eta_{GX} = Lf \circ \delta_X.$$

This amazing calculation is explained as follows: The first equality stems from the fact that the transpose is determined on its value on identities. The second is just F being a functor, and the third and fourth are a matter of definitions.

For $X \xrightarrow{f} Y$ the other equation, $K_{\mathcal{D}}G_{\mathbb{L}} = G$, becomes

$$m(f \circ \varepsilon_X) = Gf.$$

Thus we calculate

$$m^{-1}(Gf) = \varepsilon_Y \circ FGf = \varepsilon_Y \circ Lf = f \circ \varepsilon_X,$$

the last equality by the naturality of ε .

We can now show that $K_{\mathcal{D}}$ is a functor: For identities we get

$$K_{\mathcal{D}}(id_X) = K_{\mathcal{D}}(\overline{\varepsilon_X}) = m(\varepsilon_X) = id_{GX},$$

and for compositions we have by the naturality of $(\bar{\cdot})^{-1}$ and m :

$$K_{\mathcal{D}}(\overline{g \circ f}) = m(\bar{\cdot})^{-1}(\overline{g \circ f}) = m(g \circ F_{\mathbb{L}}(\overline{f})) = m(g \circ FK_{\mathcal{D}}(\overline{f})) = m(g) \circ K_{\mathcal{D}}(\overline{f}) = K_{\mathcal{D}}(\overline{g}) \circ K_{\mathcal{D}}(\overline{f}).$$

So $K_{\mathcal{D}}$ works, and its values on objects are fully determined by $K_{\mathcal{D}}X = K_{\mathcal{D}}G_{\mathbb{L}}X = GX$.

To realize the uniqueness on morphisms, we again look to naturality:

$$m(f) = m(\bar{\cdot})^{-1}(\overline{f}) = m(\bar{\cdot})^{-1}(id \circ \overline{f}) = m(\varepsilon \circ F_{\mathbb{L}}(\overline{f})) = m(\varepsilon \circ FK_{\mathcal{D}}(\overline{f})) = m(\varepsilon) \circ K_{\mathcal{D}}(\overline{f}) = id \circ K_{\mathcal{D}}(\overline{f}) = K_{\mathcal{D}}(\overline{f}),$$

where all we have used is that $FK_{\mathcal{D}} = F_{\mathbb{L}}$ and that the two adjunctions share counits. \square

2.4.5 Application

We now use our knowledge of the structure of the coKleisli category of a comonad to show one such isomorphic to a much simpler category. The comonad in question is the canonical comonad of the adjunction in Section 2.1:

$$\mathbb{L} = (L = FU, \delta = F\eta U, \varepsilon)$$

where $L: \text{Cppo}_{\perp} \rightarrow \text{Cppo}_{\perp}$ becomes a lifting functor and η and ε are the unit and counit of $F \dashv U$ respectively.

To find $\text{Cppo}_{\perp\mathbb{L}}$ we take the objects of Cppo_{\perp} and all morphisms from Cppo_{\perp} of the form $LX \xrightarrow{f} Y$. A morphism of this form is then considered a morphism from X to Y in $\text{Cppo}_{\perp\mathbb{L}}$. Such a morphism must map the added bottom element of X to the bottom element of Y since it is strict, but on all the original elements of X , it is constrained only by continuity. We now have a quite obvious bijection between the homsets $\text{Cppo}_{\perp\mathbb{L}}(X, Y)$ and $\text{Cppo}(X, Y)$. In fact we claim that $\text{Cppo}_{\perp\mathbb{L}}$ and Cppo are isomorphic:

Theorem 2.4.10. $\text{Cppo}_{\perp\mathbb{L}} \cong \text{Cppo}$

Proof: We define the functor $H: \text{Cppo}_{\perp\mathbb{L}} \rightarrow \text{Cppo}$ on objects as the identity, and on morphisms by restriction. This ensures that, for any morphism $\bar{f}: X \rightarrow Y$, $H(\bar{f}): H(X) \rightarrow H(Y)$.

Obviously, identities are mapped to identities. Composition works as follows: For $X \xrightarrow{\bar{f}} Y$ and $Y \xrightarrow{\bar{g}} Z$,

$$\overline{g \circ f} = \overline{g \circ Lf \circ \delta_X},$$

as seen earlier, but what exactly is δ_X in our case?

Recall that $\eta_X: X \rightarrow UF(X)$ is just the inclusion, and so $\delta_X = F\eta_{U(X)}: F(UX) \rightarrow FUF(UX)$ is just the inclusion of $L(X)$ into $L(X)$.

Thus $Lf \circ \delta_X$ is really f followed by an inclusion into LY , i.e., the following diagram commutes:

$$\begin{array}{ccc} LX & \xrightarrow{f} & Y \\ \delta_X \downarrow & & \downarrow i_Y \\ LX & \xrightarrow{Lf} & LY \end{array}$$

Now, $g \circ i_Y$ is the same as taking g restricted to Y , and so the restriction of $\overline{g \circ i \circ f}$ is the same as the composition of the restrictions of f and g .

Thus H is a functor, and it is clearly both full, faithful and more than essentially surjective on objects, so we have an equivalence. \square

Now we are able to compress our world into the following diagram:

$$\begin{array}{ccccc} & & \text{pCpo} & \cong & \text{Cppo}_{\perp} \\ & & \swarrow G_L & & \nwarrow F \\ & & & & \\ \text{Cppo} & \cong & \text{Cppo}_{\perp\mathbb{L}} & \xrightarrow{K} & \text{Cpo} \\ & & \swarrow F_L & & \nwarrow U \end{array}$$

For some unique K .

Chapter 3

Closure properties

Theorem 3.0.11. *Cpo is ccc and has finite coproducts (i.e. Cpo is biccc).*

Proof: The proof is so close to the case of Set, that we will provide only a sketch; we simply take the structures from Set and claim, that the required objects are cpos, and morphisms continuous. See [Oos] for some details.

The product of (A, \leq_A) with (B, \leq_B) is given as $(A \times B, \leq_{A \times B})$, where $\leq_{A \times B}$ is defined by

$$(a, b) \leq_{A \times B} (a', b') \Leftrightarrow a \leq_A a' \wedge b \leq_B b'$$

The projections are as in Set, and are easily seen to be continuous. Indeed

$$\bigsqcup_i (a_i, b_i) = \left(\bigsqcup_i a_i, \bigsqcup_i b_i \right)$$

The exponent $(B, \leq_B)^{(A, \leq_A)}$ is (B^A, \leq_{B^A}) , where \leq_{B^A} is given by

$$f \leq_{B^A} g \Leftrightarrow \forall a \in A. f(a) \leq_B g(a)$$

Here we have $\bigsqcup_i f_i$ defined by

$$\left(\bigsqcup_i f_i \right)(x) = \bigsqcup_i f_i(x)$$

That this is an adjoint to the product goes largely as in Set, too.

The coproduct of (A, \leq_A) with (B, \leq_B) is given as $(A + B, \leq_{A+B})$, where $A + B$ is the disjoint union (as in Set) and \leq_{A+B} is defined by

$$x \leq_{A+B} y \Leftrightarrow x \leq_A y \vee x \leq_B y$$

The injections are as in Set, and are easily seen to be continuous. If (x_i) is a chain in $A + B$ then it is wholly contained in either A or B . The lub is then inherited. \square

We now show that Cppo inherits the ccc property of Cpo but not the coproducts.

Theorem 3.0.12. *Cppo is a full sub-ccc of Cpo.*

Proof: Clearly all pointed cpos are in particular cpos, so it is a sub category. And clearly all continuous functions are continuous functions, so it is full. Not quite that clearly, it is also ccc. It suffices to check, that the constructed objects are pointed, though. That is easy. \square

Theorem 3.0.13.

1. Every endo function $f : X \rightarrow X$ in Cppo has a least fixed point

$$Y(f) = \bigsqcup_{n \in \omega} f^n(\perp).$$

2. Y determines a map (a continuous function) from $X^X \rightarrow X$ in Cppo.
3. (Uniformity) For every $h : X \rightarrow Y$ in Cppo $_{\perp}$, and f, g in Cppo, if the diagram

$$\begin{array}{ccc} X & \xrightarrow{h} & Y \\ f \downarrow & & \downarrow g \\ X & \xrightarrow{h} & Y \end{array}$$

commutes in Cppo, then $h(Y(f)) = Y(g)$.

Proof:

1. Consider the chain $\perp \leq f(\perp) \leq f^2(\perp) \leq \dots$, this is indeed a chain since $\perp \leq f(\perp)$ and f is monotone. Let $a = \bigsqcup_n f^n(\perp)$. Since f is continuous we have

$$f(a) = f\left(\bigsqcup_n f^n(\perp)\right) = \bigsqcup_n f^{n+1}(\perp) = a,$$

so a is a fixed point of f .

Let b be some other fixed point of f . We know that $\perp \leq b = f(b)$ and so, since f is monotone, $f^n(\perp) \leq f^n(b) = b$, for any n , which means that

$$a = \bigsqcup_n f^n(\perp) \leq \bigsqcup_n f^n(b) = \bigsqcup_n b = b.$$

2. The object X^X exists in Cppo since Cppo is a full sub-ccc of Cpo by Theorem 3.0.12. (The order in X^X is the pointwise order induced by the order in X .) Y is monotone: if $f \leq g$ in X^X , by definition $f(\perp) \leq g(\perp)$ and so because f is monotone, $f^{i+1}(\perp) \leq f(g^i(\perp)) \leq g^{i+1}(\perp)$ whenever $f^i(\perp) \leq g^i(\perp)$. By induction we then get $f^n(\perp) \leq g^n(\perp)$, so

$$Y(f) = \bigsqcup_n f^n(\perp) \leq \bigsqcup_n g^n(\perp) = Y(g).$$

Let $f_1 \leq f_2 \leq f_3 \leq \dots$ be a chain in X^X and let $f = \bigsqcup_n f_n(\perp)$. Then by the same induction argument as above, for any i and any n , $f^i(\perp) \geq f_n^i(\perp)$ and since all f_n 's and f are monotone, $f^i(\perp)$ and $f_n^i(\perp)$ are chains (in i); therefore $\bigsqcup_i f^i(\perp) \geq \bigsqcup_i f_n^i(\perp)$ so $\bigsqcup_i f^i(\perp)$ is an upper bound, for all n , by which we get $\bigsqcup_i f^i(\perp) \geq \bigsqcup_n \bigsqcup_i f_n^i(\perp)$. Now consider

$$\begin{aligned} Y(\bigsqcup_n f_n) &= \bigsqcup_i (\bigsqcup_n f_n)^i(\perp) \\ &= \bigsqcup_i f^i(\perp) \\ &\geq \bigsqcup_n \bigsqcup_i f_n^i(\perp) && \text{by the above calculation} \\ &= \bigsqcup_n Y(f_n). \end{aligned}$$

The other inequality follows from:

$$\begin{aligned}
 f_n &\leq \bigsqcup_n f_n && \text{for all } n \\
 Y(f_n) &\leq Y(\bigsqcup_n f_n) && \text{for all } n, \text{ by monotonicity of } Y \\
 \bigsqcup_n Y(f_n) &\leq \bigsqcup_n Y(\bigsqcup_n f_n) \\
 &= Y(\bigsqcup_n f_n) && \text{since } Y(\bigsqcup_n f_n) \text{ is a constant chain.}
 \end{aligned}$$

3. We reason as follows:

$$\begin{aligned}
 h(Y(f)) &= h(\bigsqcup_n f^n(\perp)) \\
 &= \bigsqcup_n h f^n(\perp) && \text{h is continuous} \\
 &= \bigsqcup_n g h f^{n-1}(\perp) && \text{by the assumption} \\
 &= \bigsqcup_n g h f^{n-2}(\perp) \\
 &\vdots \\
 &= \bigsqcup_n g^n h(\perp) \\
 &= \bigsqcup_n g^n(\perp) && \text{h is strict} \\
 &= Y(g).
 \end{aligned}$$

□

Proposition 3.0.14. *Cppo has no coproducts.*

Proof: As Cppo has fixed points and is ccc, [HP] is applicable, and tells us, that not all coproducts can exist (in [HP] it is shown, that $1 + 1$ does not exist). In fact no coproducts exists, as we can show directly:

Given pointed cpos A and B , let $A + B$ be an alleged coproduct object, with in_A and in_B the proposed injections:

$$\forall C, f, g \exists! u. \quad \begin{array}{ccc} & & C \\ & f \nearrow & \nwarrow g \\ A & \xrightarrow{in_A} & A + B \xleftarrow{in_B} B \end{array}$$

We choose C to be $\{1 < 2\}$. Now $A + B$ has a bottom element, say \perp . We have two cases:

If \perp is in the range of neither in_A nor in_B , we choose f and g both to be constant 2. Then u is not unique, as it has a free choice for \perp .

If one of the injections hit \perp , say in_A , then we define f to be constant 2, and g to be constant 1. The diagram can then never be brought to commute as B is nonempty. □

Proposition 3.0.15. *Cppo $_{\perp}$ has coequalizers*

Proof: Let A and B be pointed cpos, and f and g strict continuous function between them:

$$A \begin{array}{c} \xrightarrow{f} \\ \xrightarrow{g} \end{array} B$$

We now form the set $R = \{(fa, ga) | a \in A\} \subseteq B \times B$. This can be seen as a relation on B , and we take as \bar{R} its reflexive, symmetric and transitive closure. We then have an equivalence relation, enabling us to consider $C = B/\bar{R}$, which we promptly equip with the ordering \leq_C :

$$c_1 \leq_C c_2 \Leftrightarrow \exists b_1 \in c_1, b_2 \in c_2. b_1 <_B b_2$$

where $<_B$ denotes the order on B . On this fine set we define yet another equivalence relation $=_C$:

$$c_1 =_C c_2 \Leftrightarrow c_1 \leq_C c_2 \wedge c_2 \leq_C c_1$$

finally arriving at our coequalizer object $C/_=C$. This is a pointed cpo with the order induced from C . The coequalizer morphism is the canonical surjection s :

$$b \mapsto [b]_{=C}.$$

To see that we have indeed constructed a coequalizer, consider an object D and a morphism $B \xrightarrow{h} D$ such that $h \circ f = h \circ g$. The map $[b]_{=C} \mapsto h(b)$ then factors h through s . Of course we have to show this map well-defined i.e. $[b_1]_{=C} = [b_2]_{=C} \Rightarrow h(b_1) = h(b_2)$:

First we consider

$$(b_1, b_2) \in R \Rightarrow \exists a \in A. b_1 = f(a) \wedge b_2 = g(a) \Rightarrow h(b_1) = h(f(a)) = h(g(a)) = h(b_2)$$

thus we also have that $[b_1]_{\overline{R}} = [b_2]_{\overline{R}} \Rightarrow h(b_1) = h(b_2)$. Now

$$\begin{aligned} [b_1]_{=C} = [b_2]_{=C} & \Rightarrow \\ [b_1]_{\overline{R}} \leq_C [b_2]_{\overline{R}} \wedge [b_2]_{\overline{R}} \leq_C [b_1]_{\overline{R}} & \Rightarrow \\ \exists b'_1, b''_1 \in [b_1]_{\overline{R}}. \exists b'_2, b''_2 \in [b_2]_{\overline{R}}. b'_1 <_B b'_2 \wedge b''_1 <_B b''_2 & \Rightarrow \\ \exists b'_1, b''_1, b'_2, b''_2 \in B. h(b'_1) = h(b''_1) = h(b_1) \wedge h(b'_2) = h(b''_2) = h(b_2) \wedge b'_1 <_B b'_2 \wedge b''_1 <_B b''_2 & \Rightarrow \\ h(b_1) <_D h(b_2) \wedge h(b_2) <_D h(b_1) & \Rightarrow \\ h(b_1) = h(b_2) & \Rightarrow \end{aligned}$$

so our map is well-defined, and clearly factors h through s uniquely. \square

3.1 SMCC

Proposition 3.1.1. *Cppo $_{\perp}$ has finite products, inherited from Cpo.*

Proof: By Proposition 2.2.1 and by the fact, according to [Oos, p. 59], that $\text{Cpo}^{\mathbb{T}}$ is at least as complete as Cpo. The latter meaning that whenever a limit of a diagram exists in Cpo the diagram also has a limit in Cppo_{\perp} . \square

Exponentials are not inherited though. However w.r.t. a weaker notion of product, a tensor product, Cppo_{\perp} does have a closed structure as we shall now show.

Definition 3.1.2. *A category \mathcal{C} is said to be monoidal if there is a functor $\otimes: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$, an object $e \in \text{Obj}(\mathcal{C})$, and natural isomorphisms*

$$\alpha_{a,b,c}: a \otimes (b \otimes c) \rightarrow (a \otimes b) \otimes c, \quad \lambda_a: e \otimes a \rightarrow a, \quad \rho_a: a \otimes e \rightarrow a,$$

so that

$$\begin{array}{ccc} a \otimes (b \otimes (c \otimes d)) & \xrightarrow{\alpha} & (a \otimes b) \otimes (c \otimes d) \\ \downarrow 1 \otimes \alpha & & \downarrow \alpha \\ a \otimes ((b \otimes c) \otimes d) & & ((a \otimes b) \otimes c) \otimes d \\ \searrow \alpha & \nearrow \alpha \otimes 1 & \\ & (a \otimes (b \otimes c)) \otimes d & \end{array} \quad \begin{array}{ccc} a \otimes (e \otimes b) & & \\ \downarrow \alpha & \searrow 1 \otimes \lambda & \\ (a \otimes e) \otimes b & \nearrow \rho \otimes 1 & a \otimes b \end{array}$$

commute. $(\mathcal{C}, \otimes, e, \alpha, \lambda, \rho)$ is then a monoidal category.

Definition 3.1.3. A monoidal category $(\mathcal{C}, \otimes, e, \alpha, \lambda, \rho)$ is said to be symmetric if there is a natural isomorphism

$$\gamma_{a,b}: a \otimes b \rightarrow b \otimes a$$

so that

$$\begin{array}{ccc} a \otimes b & \xrightarrow{id} & a \otimes b \\ & \searrow \gamma & \nearrow \gamma \\ & b \otimes a & \end{array}, \quad \begin{array}{ccc} a \otimes e & \xrightarrow{\rho} & a \\ & \searrow \gamma & \nearrow \lambda \\ & e \otimes a & \end{array}$$

and

$$\begin{array}{ccc} a \otimes (b \otimes c) & \xrightarrow{\alpha} & (a \otimes b) \otimes c \\ \downarrow 1 \otimes \gamma & & \searrow \gamma \\ a \otimes (c \otimes b) & & c \otimes (a \otimes b) \\ & \searrow \alpha & \downarrow \alpha \\ & (a \otimes c) \otimes b & \xrightarrow{\gamma \otimes 1} & (c \otimes a) \otimes b \end{array}$$

commute. $(\mathcal{C}, \otimes, e, \alpha, \lambda, \rho, \gamma)$ is then a symmetric monoidal category.

Definition 3.1.4. A symmetric monoidal category $(\mathcal{C}, \otimes, e, \alpha, \lambda, \rho, \gamma)$ is called closed if the functor

$$- \otimes b: \mathcal{C} \rightarrow \mathcal{C}$$

has a right adjoint $b \multimap -$, for all $b \in \text{Obj}(\mathcal{C})$. $(\mathcal{C}, \otimes, e, \alpha, \lambda, \rho, \gamma, (b \multimap -)_{b \in \text{Obj}(\mathcal{C})})$ is then a symmetric monoidal closed category.

We now turn to the category Cppo_\perp , which we wish to show symmetric monoidal closed. To that end we define the smash product by means of the ordinary product \times on pCpo , and the functors $(-)_\downarrow$ and $(-)_\perp$. Thus, we first define these:

Definition 3.1.5. $(-)_\downarrow: \text{Cppo}_\perp \rightarrow \text{pCpo}$

For $A \in \text{Obj}(\text{Cppo}_\perp)$, $A_\downarrow \in \text{Obj}(\text{pCpo})$ is just A with the bottom element removed. For $A, B \in \text{Obj}(\text{Cppo}_\perp)$ and $f \in \text{Cppo}_\perp(A, B)$, $f_\downarrow \in \text{pCpo}(A_\downarrow, B_\downarrow)$ is just f restricted to $f^{-1}(B_\downarrow)$, regarded as a partial function on A_\downarrow .

Note that this is not just a composite of previously defined functors.

Definition 3.1.6. The functor $(-)_\perp: \text{pCpo} \rightarrow \text{Cppo}_\perp$ works just like the functor $F: \text{Cpo} \rightarrow \text{Cppo}_\perp$ from Section 2.1. Objects are loaded with an extra bottom element, and for $f \in \text{pCpo}(A, B)$, f_\perp becomes everywhere defined, by sending all points in A_\perp on which f is not defined (including the added bottom element) to the added bottom element of B_\perp .

Proposition 3.1.7. $(-)_\downarrow: \text{Cppo}_\perp \rightarrow \text{pCpo}$ and $(-)_\perp: \text{pCpo} \rightarrow \text{Cppo}_\perp$ are functors.

As the functors $(-)_\downarrow$ and $(-)_\perp$ are inverses, we have the following isomorphism:

Proposition 3.1.8. $\text{Cppo}_\perp \cong \text{pCpo}$.

And so we are ready to define the smash product on Cppo_\perp :

Definition 3.1.9. *The functor $\otimes : \text{Cppo}_\perp \times \text{Cppo}_\perp \rightarrow \text{Cppo}_\perp$ is defined by:*

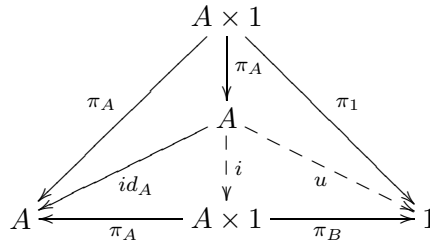
$$A \otimes B \stackrel{\text{def}}{=} (A_\downarrow \times B_\downarrow)_\perp.$$

$$f \otimes g \stackrel{\text{def}}{=} (f_\downarrow \times g_\downarrow)_\perp.$$

At this point we need the following nice basic lemmas for Cartesian categories. We will use the notation that $\pi_A^{A \times B}$ and $\pi_B^{A \times B}$ in general denotes the projections from $A \times B$ to A and B respectively. As labels on arrows in diagrams, and in other contexts where the domain is obvious, we will leave out the superscript.

Lemma 3.1.10. *The projections $\pi_A^{A \times 1}$ and $\pi_1^{1 \times A}$ are isomorphisms, for all objects A .*

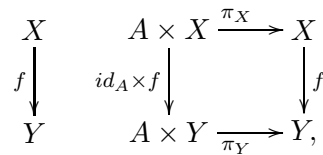
Proof: Consider the following commuting diagram:



where i is the product induced unique morphism that actually make it commute, and u is the unique morphism into the terminal object. We see that $\pi_A \circ i$ is the identity. As projections are monos¹ and $\pi_A \circ (i \circ \pi_A) = \pi_A \circ id$, $i \circ \pi_A$ is also the identity. Similarly, π_B is also invertible. □

Lemma 3.1.11. *Projections are natural in their codomains.*

Proof: We claim that for all objects A and B , $\pi_-^{A \times -}$ is a natural transformation from $A \times -$ to id and $\pi_-^{- \times B}$ from $- \times B$ to id . The commuting square for $\pi_X^{A \times X}$ looks like this:



which commutes by definition. The same goes for $\pi_-^{- \times B}$. □

Lemma 3.1.12. *Products of isomorphisms are isomorphisms.*

Proof: This is obvious, as $id_A \times id_B = id_{A \times B}$, for all A and B . □

Lemma 3.1.13. *Products of natural transformations are natural transformations.*

¹This is easily seen in the product diagram, since f is $\langle \pi_A \circ f, \pi_B \circ f \rangle$ and unique.

Proof: We claim that for natural transformations $\alpha: F \Rightarrow G$ and $\beta: F' \Rightarrow G'$, $\alpha \times \beta: F \times G \Rightarrow F' \times G'$ is a natural transformation. The naturality square looks like this:

$$\begin{array}{ccccc}
 X \times X' & & FX \times GX' & \xrightarrow{\alpha_X \times \beta_{X'}} & F'X \times G'X \\
 \downarrow f \times g & & \downarrow Ff \times Gg & & \downarrow F'f \times G'g \\
 Y \times Y' & & FY \times GY & \xrightarrow{\alpha_Y \times \beta_{Y'}} & F'Y \times G'Y'
 \end{array}$$

which commutes as

$$(\alpha_Y \times \beta_{Y'}) \circ (Ff \times Gg) = (\alpha_Y \circ Ff) \times (\beta_{Y'} \circ Gg) = (F'f \circ \alpha_X) \times (G'g \circ \beta_{X'}) = (F'f \times G'g) \circ (\alpha_X \times \beta_{X'}).$$

□

Theorem 3.1.14. $(\text{Cppo}_\perp, \otimes, 1_\perp, \alpha, \lambda, \rho, \gamma)$, is a symmetric monoidal category, where \otimes is smash product, 1 denotes the terminal object of pCpo , and α, λ, ρ and γ are given by:

$$\begin{aligned}
 \alpha_{A,B,C} &= ((\pi_{A_\downarrow}^{(A_\downarrow \times (B_\downarrow \times C_\downarrow))}) \times \pi_{B_\downarrow}^{(A_\downarrow \times (B_\downarrow \times C_\downarrow))}) \times \pi_{C_\downarrow}^{(A_\downarrow \times (B_\downarrow \times C_\downarrow))})_\perp \\
 \lambda_A &= (\pi_{A_\downarrow}^{A_\downarrow \times 1})_\perp \\
 \rho_A &= (\pi_{A_\downarrow}^{1 \times A_\downarrow})_\perp \\
 \gamma_{A,B} &= (\pi_{A_\downarrow}^{A_\downarrow \times B_\downarrow} \times \pi_{B_\downarrow}^{A_\downarrow \times B_\downarrow})_\perp
 \end{aligned}$$

Here $\pi_{B_\downarrow}^{(A_\downarrow \times (B_\downarrow \times C_\downarrow))}$ is to be understood as $\pi_{B_\downarrow}^{B_\downarrow \times C_\downarrow} \circ \pi_{B_\downarrow \times C_\downarrow}^{(A_\downarrow \times (B_\downarrow \times C_\downarrow))}$.

Proof: We are to show that the above mentioned morphisms are natural bijections. This is largely done by the previous lemmas, as naturality is preserved by composition and functors preserve commuting diagrams.

That the defining diagrams commute is an easy albeit tedious case of trial and success. □

Proposition 3.1.15. For each object B in Cppo_\perp , the functor $-\otimes B: \text{Cppo}_\perp \rightarrow \text{Cppo}_\perp$ has a right adjoint $B \multimap -$, where $B \multimap C$ is the set of strict, continuous functions from B to C .

Proof: Clearly $B \multimap C$ is an object of Cppo_\perp ; ordering is pointwise, i.e., $f \sqsubseteq g \Leftrightarrow \forall x. f(x) \sqsubseteq g(x)$. The bottom element $\perp_{B \multimap C}$ is $\lambda b. \perp_C$. The functor $B \multimap -$ sends an arrow $f: C \rightarrow C'$ to $\hat{f}: B \multimap C \rightarrow B \multimap C'$, by $\hat{f}(\alpha) = f \circ \alpha$. We define a natural isomorphism

$$\text{Cppo}_\perp(A \otimes B, C) \xrightarrow{m_{A,C}} \text{Cppo}_\perp(A, B \multimap C)$$

by mapping $f: A \otimes B \rightarrow C$ to $m_{A,C}(f) = \text{exp}_f: A \rightarrow B \multimap C$ where $\text{exp}_f(a) = \lambda b. \hat{f}(a, b)$, where \hat{f} is the strict extension of f to $A \times B$, i.e.

$$\hat{f}(a, b) = \begin{cases} f(a, b) & \text{if } (a, b) \in A_\downarrow \times B_\downarrow \\ \perp_C & \text{otherwise.} \end{cases}$$

Clearly \hat{f} is strict and continuous. $exp_f(\perp_A) = \lambda b. \hat{f}(\perp_A, b) = \lambda b. \perp_C$, so exp_f is strict; it is also monotone: for $a_1 \sqsubseteq a_2$ we have $exp_f(a_1) = \lambda b. f(a_1, b) \sqsubseteq \lambda b. f(a_2, b) = exp_f(a_2)$ since f is monotone. To see that exp_f preserves lubs, let (a_i) be a chain. $exp_f(\bigsqcup a_i) = \lambda b. \hat{f}(\bigsqcup a_i, b) = \bigsqcup_i \lambda b. \hat{f}(a_i, b)$ (by lemma 1.1.9 since \hat{f} is continuous.) $= \bigsqcup_i exp_f(a_i)$, so exp_f is indeed a strict, cts. function. Furthermore, for any a , $exp_f(a)$ is an object of $B \multimap C$, because $exp_f(a)$ is a strict, cts. function by lemma 1.1.9 and \hat{f} being strict and continuous. For $g : A \rightarrow B \multimap C$ strict and continuous, we define $m_{A,C}^{-1}(g) = \tilde{g} : A \otimes B \rightarrow C$ by

$$\begin{aligned} \tilde{g}(a, b) &= g(a)(b) \quad \text{and} \\ \tilde{g}(\perp_{A \otimes B}) &= \perp_C \end{aligned}$$

\tilde{g} is strict, to see that it is monotone consider $(a_1, b_1) \sqsubseteq (a_2, b_2)$, $\tilde{g}(a_1, b_1) = g(a_1)(b_1) \sqsubseteq g(a_2)(b_1)$ (because g is monotone) $\sqsubseteq g(a_2)(b_2) = \tilde{g}(a_2, b_2)$ (because $g(a_2)$ is monotone). Note that a chain in $A \otimes B$ is either the constant chain $\perp_{A \otimes B}$ or it is a chain in A or B possibly starting with $\perp_{A \otimes B}$. Therefore we only need to consider the restriction of \tilde{g} to $A_\downarrow \times B_\downarrow$ in order to show that \tilde{g} is continuous. By lemma 1.1.9 $\tilde{g}|_{A_\downarrow \times B_\downarrow}$ is continuous iff it is continuous in both arguments. Let (a_i) be a chain in A_\downarrow , then $\tilde{g}(\bigsqcup a_i, b) = g(\bigsqcup a_i)(b) = (\bigsqcup g(a_i))(b)$ (since g is cts.) $= \bigsqcup \tilde{g}(a_i, b)$. Let (b_i) be a chain in B , then $\tilde{g}(a, \bigsqcup b_i) = g(a)(\bigsqcup b_i) = \bigsqcup_i g(a)(b_i)$ because $g(a)$ is continuous. It is straight forward to verify that m and m^{-1} are inverses. For naturality, consider $f : A \rightarrow A'$ and $g : C' \rightarrow C$, both strict and cts. To verify that the square

$$\begin{array}{ccc} \text{Cppo}_\perp(A \otimes B, C) & \xrightarrow{m_{A,C}} & \text{Cppo}_\perp(A, B \multimap C) \\ \text{Cppo}_\perp(f \otimes Id_B, g) \uparrow & & \uparrow \text{Cppo}_\perp(f, \bar{g}) \\ \text{Cppo}_\perp(A' \otimes B, C') & \xrightarrow{m_{A',C'}} & \text{Cppo}_\perp(A', B \multimap C') \end{array}$$

commutes, take $\alpha \in \text{Cppo}_\perp(A' \otimes B, C')$ and let's chase it through the diagram. On one side we get $m_{A,C}(g \circ \alpha \circ f \otimes id_B)$ and on the other side we get $\bar{g} \circ m_{A',C'}(\alpha) \circ f$. Let $g \circ \alpha \circ f \otimes id_B = \gamma$ then $m_{A,C}(\gamma) = exp_\gamma$ so for $a \in A$ $exp_\gamma(a) = \lambda b. \hat{\gamma}(a, b) = \lambda b. g \circ \hat{\alpha} \circ f \otimes id_B(a, b)$ (because f and g are strict) $= \lambda b. g \circ \hat{\alpha}(f(a), b)$.

$\bar{g} \circ m_{A',C'}(\alpha) \circ f(a) = \bar{g} \circ exp_\alpha \circ f(a) = \bar{g}(\lambda b. \hat{\alpha}(f(a), b)) = \lambda b. g \circ \hat{\alpha}(f(a), b)$. By extensionality we get $m_{A,C}(g \circ \alpha \circ f \otimes id_B) = \bar{g} \circ m_{A',C'}(\alpha) \circ f$. \square

Corollary 3.1.16. $(\text{Cppo}_\perp, \otimes, 1_\perp, \alpha, \lambda, \rho, \gamma)$, is a symmetric monoidal closed category.

3.2 Partial exponentials

Proposition 3.2.1. pCpo is partial Cartesian closed.

Proof: The statement means that the subcategory of total maps, Cpo is ccc. This happens to be the case according to 3.0.11. \square

The product functor on Cpo extends to a functor $\times' : \text{pCpo} \times \text{pCpo} \rightarrow \text{pcpo}$. \times' equals \times on objects, and for partial functions f, g

$$f \times' g(x, y) = \begin{cases} f \times g(x, y) & \text{if both } fx \text{ and } gy \text{ are defined} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Proposition 3.2.2. pCpo has partial exponentials.

Proof: That pCpo has partial exponentials means that, for every X in Cpo the functor $(- \times' X) : \text{Cpo} \rightarrow \text{pCpo}$ has a right adjoint $X \multimap - : \text{pCpo} \rightarrow \text{Cpo}$. When this holds one also says that the Kleisli category (pCpo) has Kleisli exponentials. We are already familiar with one adjunction between Cpo and pCpo (defined in section 2.3), namely

$$\text{Cpo} \begin{array}{c} \xleftarrow{G_{\mathbb{T}}V^{-1}} \\ \xrightarrow{VF_{\mathbb{T}}} \end{array} \text{pCpo}, \quad VF_{\mathbb{T}} \dashv G_{\mathbb{T}}V^{-1}$$

composing this with the adjunction

$$\text{Cpo} \begin{array}{c} \xleftarrow{(-)^X} \\ \xrightarrow{- \times X} \end{array} \text{Cpo}, \quad - \times X \dashv (-)^X$$

we get an adjunction

$$\text{Cpo} \begin{array}{c} \xleftarrow{(-)^X} \\ \xrightarrow{- \times X} \end{array} \text{Cpo} \begin{array}{c} \xleftarrow{G_{\mathbb{T}}V^{-1}} \\ \xrightarrow{VF_{\mathbb{T}}} \end{array} \text{pCpo}$$

that just might be the one we are searching for. We only need to show that for every X , $- \times' X$ equals $VF_{\mathbb{T}}(- \times X)$. For objects this is clear since $VF_{\mathbb{T}}$ is the identity on objects. On arrows $f : Y \rightarrow Y'$ in Cpo , we have

$$f \xrightarrow{- \times X} f \times id_X \xrightarrow{F_{\mathbb{T}}} \eta_{Y \times X} \circ f \times id_X$$

recall that $\eta_{Y \times X} : Y \times X \rightarrow T(Y \times X)$ is the inclusion and $\eta_{Y \times X} \circ f \times id_X : X \rightarrow T(Y \times X)$ is considered as an arrow from $X \rightarrow Y \times X$ in $\text{Cpo}_{\mathbb{T}}$. Now

$$V(\eta_{Y \times X} \circ f \times id_X) = \gamma_{Y \times X} \circ \eta_{Y \times X} \circ f \times id_X = f \times id_X$$

since $(\eta_{Y \times X} \circ f \times id_X)^{-1}(\perp_{Y \times X}) = \emptyset$.

So $X \multimap - \stackrel{\text{def}}{=} (G_{\mathbb{T}}V^{-1}(-))^X$ and $X \multimap Y$ is the cppo of continuous functions from X to TY , which is isomorphic (by V) to the cppo of partial continuous functions from X to Y . \square

Remark 3.2.3. From [Jac] we get the following results which we mention without any further details.

- Cpo is smc (in fact ccc) and \mathbb{T} is a monad on Cpo which we claim is commutative (we will not define nor proof this property). By [Jac, Theorem 4.3] this implies that the Kleisli category $\text{Cpo}_{\mathbb{T}} = \text{pCpo}$ also has an smc structure and the free functor $\text{Cpo} \xrightarrow{F_{\mathbb{T}}} \text{Cpo}_{\mathbb{T}}$ preserves the smc structure.

We have shown above that $(- \times' -) = VF_{\mathbb{T}}(- \times -)$, i.e., the product on pCpo is the preserved product. This means that the product \times' on pCpo is in fact an smc tensor product and the right adjoint for \times' that we found by guessing is the preserved closed structure.

- Cpo is smc and \mathbb{T} is a commutative monad and by Proposition 3.0.15 $\text{Cpo}^{\mathbb{T}} = \text{Cppo}_{\perp}$ has coequalizers (in fact only coequalizers of reflexive pairs is required). By [Jac, Lemma 5.2] $\text{Cpo}^{\mathbb{T}}$ is smc and $\text{Cpo} \xrightarrow{F} \text{Cppo}_{\perp}$ preserves smc structure.

The preserved product matches our smash product, since for any $A, B \in \text{Obj}(\text{Cpo})$ we have $F(A \times B) = FA \otimes FB$. (Similarly for arrows).

In this chapter we have been looking at quite a few closure properties for our four categories of interest. We feel that now is an appropriate time to move on, but if we were to study this further, we would consider the following question: For which diagrams is \mathbf{Cpo} or \mathbf{Cppo}_\perp complete or cocomplete?

Part II

Adequacy

Chapter 4

Call-by-value

In this chapter we present two semantics for the meta programming language PCF, and then we show a form of equivalence between them called adequacy.

4.1 The language PCF

PCF is a simply typed language with two ground types: ι and o , and for any types σ, τ PCF also has the type $\sigma \rightarrow \tau$.

Terms We have denumerably many variables: $x, y, z, \dots \in \text{Var}$. We have the following constants:

- $\text{pred} : \iota \rightarrow \iota$,
- $\text{succ} : \iota \rightarrow \iota$,
- $\text{zero?} : \iota \rightarrow o$,
- $\text{cond}^\beta : o \rightarrow (\beta \rightarrow (\beta \rightarrow \text{beta}))$, for β ground type.
- $\text{tt} : o$,
- $\text{ff} : o$,
- $\bar{n} : \iota$, one for each $n \in \mathbb{N}$.

The terms of PCF are then defined by the following grammar:

$$t ::= x \mid k \mid (t_1 t_2) \mid \lambda x : \sigma. t \mid \mu x. t$$

where k ranges over the above constants.

Definition 4.1.1. *Let $\text{FV}(t)$ be the set of free variables of the term t , we say that a term t is closed iff $\text{FV}(t) = \emptyset$.*

Typing rules Wellformed terms are terms that are typeable by the following typing rules. Since a variable can be used as a placeholder for a term of any type, we need a context that provides information about the types of the free variables in a term. A context Γ is a finite mapping (i.e., a partial function with a finite domain) from the set Var of variables to the collection of types.

$$\frac{\Gamma(x) = \sigma}{\Gamma \vdash x : \sigma},$$

$$\Gamma \vdash \bar{n} : \iota, \quad \Gamma \vdash \text{tt} : \iota, \quad \Gamma \vdash \text{ff} : \iota,$$

$$\Gamma \vdash \text{pred} : \iota \rightarrow \iota, \quad \Gamma \vdash \text{succ} : \iota \rightarrow \iota, \quad \Gamma \vdash \text{zero?} : \iota \rightarrow o,$$

$$\Gamma \vdash \text{cond}^\beta : o \rightarrow (\beta \rightarrow (\beta \rightarrow \beta)),$$

$$\frac{\Gamma \vdash t_1 : \sigma \rightarrow \sigma' \quad \Gamma \vdash t_2 : \sigma}{\Gamma \vdash (t_1 t_2) : \sigma'},$$

$$\frac{\Gamma, x : \sigma \vdash t : \sigma'}{\Gamma \vdash \lambda x : \sigma. t : \sigma \rightarrow \sigma'}, \quad \frac{\Gamma, f : \sigma \vdash t : \sigma}{\Gamma \vdash \mu f. t : \sigma}.$$

The type system is deterministic. Note that a term t cannot be typeable unless $\text{FV}(t) \subseteq \text{dom}(\Gamma)$

Canonical forms Canonical forms are closed terms of the form

$$\bar{n} \mid \lambda x : \sigma. t \mid \text{tt} \mid \text{ff}$$

Canonical forms are sometimes called syntactic values.

PCF Programs PCF Programs are closed terms of ground type.

Example 4.1.2. *The term $\text{fac} \equiv \mu f. \lambda x : \iota. \text{cond}^\iota(\text{zero? } x) \bar{1}(x \times f(\text{pred } x))$ is a closed term of type $\iota \rightarrow \iota$. Thus, for all n , $\text{fac } \bar{n}$ is a PCF program.*

We have split the proof tree into four parts in order to make it fit on the page. The first trees prove the assumptions of the later ones.

$$\frac{f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^\iota : o \rightarrow (\iota \rightarrow (\iota \rightarrow \iota)) \quad \frac{f : \iota \rightarrow \iota, x : \iota \vdash \text{zero?} : \iota \rightarrow o \quad f : \iota \rightarrow \iota, x : \iota \vdash x : \iota}{f : \iota \rightarrow \iota, x : \iota \vdash (\text{zero? } x) : o}}{f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^\iota(\text{zero? } x) : \iota \rightarrow (\iota \rightarrow \iota)}$$

enables

$$\frac{f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^\iota(\text{zero? } x) : \iota \rightarrow (\iota \rightarrow \iota) \quad f : \iota \rightarrow \iota, x : \iota \vdash \bar{1} : \iota}{f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^\iota(\text{zero? } x) \bar{1} : \iota \rightarrow \iota}$$

too, with

$$\frac{f : \iota \rightarrow \iota, x : \iota \vdash x : \iota \quad \frac{f : \iota \rightarrow \iota, x : \iota \vdash f : \iota \rightarrow \iota \quad \frac{f : \iota \rightarrow \iota, x : \iota \vdash \text{pred} : \iota \rightarrow \iota \quad f : \iota \rightarrow \iota, x : \iota \vdash x : \iota}{f : \iota \rightarrow \iota, x : \iota \vdash \text{pred } x : \iota}}{f : \iota \rightarrow \iota, x : \iota \vdash f(\text{pred } x) : \iota}}{f : \iota \rightarrow \iota, x : \iota \vdash x \times f(\text{pred } x) : \iota}$$

gives

$$\frac{\frac{f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^t(\text{zero? } x) \bar{1} : \iota \rightarrow \iota \quad f : \iota \rightarrow \iota, x : \iota \vdash x \times f(\text{pred } x) : \iota}{f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)) : \iota \rightarrow \iota}}{f : \iota \rightarrow \iota \vdash \lambda x. \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)) : \iota \rightarrow \iota} \vdash \mu f. \lambda x. \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)) : \iota \rightarrow \iota$$

Here we have used the macro $a \times b \equiv (\mathbf{tms } a)b$ where

$$\mathbf{tms} \equiv \mu f. \lambda a : \iota. \lambda b : \iota. \text{cond}^t(\text{zero? } a) \bar{0}(f(\text{pred } a)b + b)$$

which again uses the macro $a + b \equiv (\mathbf{plus } a)b$ where

$$\mathbf{plus} \equiv \mu f. \lambda a : \iota. \lambda b : \iota. \text{cond}^t(\text{zero? } a)b(\text{succ}(f(\text{pred } a)b)).$$

Similar derivations to the above shows \mathbf{tms} and \mathbf{plus} closed terms of type $\iota \rightarrow \iota$.

4.2 Call-by-value operational semantics for PCF

An operational semantics is a set of evaluation rules that defines a mapping (evaluation) from terms to canonical forms. The rules are:

$$\begin{aligned} & c \Rightarrow c, \text{ for canonical forms } c, \\ & \frac{t \Rightarrow \overline{n+1}}{\text{pred } t \Rightarrow \bar{n}}, \quad \frac{t \Rightarrow \bar{0}}{\text{pred } t \Rightarrow \bar{0}}, \quad \frac{t \Rightarrow \bar{n}}{\text{succ } t \Rightarrow \overline{n+1}}, \quad \frac{t \Rightarrow \overline{n+1}}{\text{zero? } t \Rightarrow \text{ff}}, \\ & \frac{t \Rightarrow \bar{0}}{\text{zero? } t \Rightarrow \text{tt}}, \quad \frac{t_0 \Rightarrow \text{tt} \quad t_1 \Rightarrow c}{\text{cond}^\beta t_0 t_1 t_2 \Rightarrow c}, \quad \frac{t_0 \Rightarrow \text{ff} \quad t_2 \Rightarrow c}{\text{cond}^\beta t_0 t_1 t_2 \Rightarrow c}, \\ & \frac{t_1 \Rightarrow \lambda x. t'_1 \quad t_2 \Rightarrow c_2 \quad t'_1[c_2/x] \Rightarrow c}{(t_1 t_2) \Rightarrow c}, \quad \frac{t[\mu f. t/f] \Rightarrow c}{\mu f. t \Rightarrow c}. \end{aligned}$$

Proposition 4.2.1. *Evaluation is deterministic and respects types.*

Proof: That evaluation is deterministic is obvious. That types are respected, is shown by structural induction on the rules. We will treat only the two last cases:

$$\frac{t_1 \Rightarrow \lambda x. t'_1 \quad t_2 \Rightarrow c_2 \quad t'_1[c_2/x] \Rightarrow c}{(t_1 t_2) \Rightarrow c} : \text{Assume } c : \alpha \text{ and } c_2 : \beta. \text{ By the induction hypothesis, we have a } \Gamma \text{ such that } \Gamma \vdash t'_1[c_2/x] : \alpha. \text{ Thus } \Gamma, x : \beta \vdash t'_1 : \alpha \text{ yielding } \Gamma \vdash \lambda x : \beta. t'_1 : \beta \rightarrow \alpha. \text{ Thus knowing } t_1 : \beta \rightarrow \alpha \text{ and } t_2 : \beta, \text{ all by the induction hypothesis, we conclude } (t_1 t_2) : \alpha \text{ as required.}$$

$$\frac{t[\mu f. t/f] \Rightarrow c}{\mu f. t \Rightarrow c} : \text{Assume } c : \alpha. \text{ If } f \text{ is not free in } t, \text{ we have } t : \alpha, \text{ quickly leading to } \mu f. t : \alpha. \text{ Otherwise, assume } \mu f. t : \beta. \text{ We have, as before, a } \Gamma \text{ such that } \Gamma, f : \beta \vdash t : \alpha. \text{ However, as the typing system is deterministic, we infer from the typing of } \mu f. t \text{ a } \Gamma' \text{ so that } \Gamma', f : \alpha \vdash t : \alpha. \text{ These two typings must then coincide, and } \alpha = \beta.$$

The remaining cases are easy. □

4.3 Call-by-value denotational semantics

We have now reached the point where we connect Part 1 with Part 2. A denotational semantics is a mapping from “terms in context” to morphisms between mathematical domains like the ones we have been studying in Part 1. We say that we interpret the terms when we define this mapping, and together with the collection of domains this is called a model (of PCF). Our model is the category pCpo , and we interpret types as objects of the category and terms as morphism of the category. This will probably get more clear as we carry on.

Interpretation of contexts and types. Types are interpreted as follows:

- $\llbracket \iota \rrbracket = \mathbb{N}$,
- $\llbracket o \rrbracket = \mathbb{T}$,
- $\llbracket \sigma_1 \rightarrow \sigma_2 \rrbracket = \llbracket \sigma_1 \rrbracket \rightarrow \llbracket \sigma_2 \rrbracket$

Here \mathbb{N} is the discrete cpo of natural numbers, i.e., no elements are comparable. \mathbb{T} is the discrete cpo $\{\text{true}, \text{false}\}$. And $\llbracket \sigma_1 \rrbracket \rightarrow \llbracket \sigma_2 \rrbracket$ is the cppo of all partial continuous functions from the cpo $\llbracket \sigma_1 \rrbracket$ to $\llbracket \sigma_2 \rrbracket$. (This is actually the Kleisli exponential).

Note that $\llbracket \sigma \rrbracket$ has a bottom element if and only if σ is a function type.

Recall that $\Gamma : \text{Var} \rightarrow \text{Types}$.

Definition 4.3.1. An environment ρ for Γ is a partial function from Var to $\bigcup_{x \in \text{dom}(\Gamma)} \llbracket \Gamma(x) \rrbracket$ which is typerespecting, i.e., $\rho(x) \in \llbracket \Gamma(x) \rrbracket$.

We also have an updating function:

$$\rho[v/x](y) \stackrel{\text{def}}{=} \begin{cases} v & \text{if } y = x \\ \rho(y) & \text{else} \end{cases}$$

Now we can define how contexts are interpreted:

$$\llbracket \Gamma \rrbracket = \{\rho \mid \forall x \in \text{dom}(\Gamma). \rho(x) \downarrow \Rightarrow \rho(x) \in \llbracket \Gamma(x) \rrbracket\}.$$

Lemma 4.3.2. $\llbracket \Gamma \rrbracket$ is a cpo ordered by \sqsubseteq .

Denotational semantics

Definition 4.3.3. Given a partial continuous function $\phi: \llbracket \Gamma, f : \sigma \rrbracket \rightarrow \llbracket \sigma \rrbracket$, $\widehat{\phi}: \llbracket \Gamma \rrbracket \rightarrow (\llbracket \sigma \rrbracket \rightarrow \llbracket \sigma \rrbracket)$ is given by

$$\widehat{\phi}(\rho) = v \mapsto \phi(\rho[v/f]).$$

The interpretation of a term $t : \sigma$ in context Γ should be a partial continuous function from environments to interpretation of σ , i.e.,

$$\llbracket \Gamma \vdash t : \sigma \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket.$$

- $\llbracket \Gamma \vdash \bar{n} : \iota \rrbracket \rho = n$,
- $\llbracket \Gamma \vdash \text{tt} : o \rrbracket \rho = \text{true}$,

- $\llbracket \Gamma \vdash \text{ff} : o \rrbracket \rho = \text{false}$,
- $\llbracket \Gamma \vdash \text{succ} : \iota \rightarrow \iota \rrbracket \rho = x \mapsto x + 1$,
- $\llbracket \Gamma \vdash \text{pred} : \iota \rightarrow \iota \rrbracket \rho = x \mapsto x - 1$,
- $\llbracket \Gamma \vdash \text{zero?} : \iota \rightarrow \iota \rrbracket \rho = x \mapsto \text{if } x = 0 \text{ then } \text{true} \text{ else } \text{false}$,
- $\llbracket \Gamma \vdash \text{cond}^\beta : o \rightarrow (\beta \rightarrow (\beta \rightarrow \beta)) \rrbracket \rho = x_1 \mapsto (x_2 \mapsto (x_3 \mapsto \text{if } x_1 \text{ then } x_2 \text{ else } x_3))$,
- $\llbracket \Gamma \vdash x : \sigma \rrbracket \rho = \rho(x)$,
- $\llbracket \Gamma \vdash (t_1 t_2) : \sigma' \rrbracket \rho = \llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho \llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho$,
- $\llbracket \Gamma \vdash \lambda x.t : \sigma \rightarrow \sigma' \rrbracket \rho = v \mapsto \llbracket \Gamma, x : \sigma \vdash t : \sigma' \rrbracket \rho[v/x]$,
- $\llbracket \Gamma \vdash \mu f.t : \sigma \rrbracket \rho = \bigsqcup_{n \in \omega} g(n)\rho$, where
 - $g(n) : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$
 - $g(0)\rho \uparrow$
 - $g(n+1)\rho = \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho[g(n)\rho/f]$

We are now going to show that this interpretation is well-defined, i.e., that for all typeable terms $t : \sigma$, $\llbracket \Gamma \vdash t : \sigma \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$ is a partial continuous function of cpos. Before we can show this we need some lemmas:

Lemma 4.3.4. *For $v \in \llbracket \sigma \rrbracket$ and $f \notin \text{dom}(\Gamma)$, $\rho \in \llbracket \Gamma \rrbracket \Rightarrow \rho[v/f] \in \llbracket \Gamma, f : \sigma \rrbracket$.*

Proof: First note that

$$f \notin \text{dom}(\Gamma) \Rightarrow \forall x \in \text{dom}(\Gamma). \Gamma(x) = (\Gamma, f : \sigma)(x) \wedge \rho[v/f](x) = \rho(x)$$

Now

$$\begin{aligned} \rho \in \llbracket \Gamma \rrbracket &\Rightarrow \forall x \in \text{dom}(\Gamma). \rho(x) \in \llbracket \Gamma(x) \rrbracket \\ &\Rightarrow \forall x \in \text{dom}(\Gamma). \rho[v/f](x) \in \llbracket (\Gamma, f : \sigma)(x) \rrbracket \wedge \\ &\quad \rho[v/f](f) = v \in \llbracket \sigma \rrbracket = \llbracket (\Gamma, f : \sigma)(f) \rrbracket \\ &\Rightarrow \rho[v/f] \in \llbracket (\Gamma, f : \sigma)(f) \rrbracket \end{aligned}$$

□

Lemma 4.3.5. *Given $\rho \in \llbracket \Gamma \rrbracket$ and a variable f with $\Gamma(f) = \sigma$, the function $v \mapsto \rho[v/f] : \llbracket \sigma \rrbracket \rightarrow \llbracket \Gamma \rrbracket$ is continuous.*

Proof: Easy from the definitions. □

Proposition 4.3.6. *Given a valid typing $\Gamma \vdash t : \sigma$, the interpretation is well-defined and $\llbracket \Gamma \vdash t : \sigma \rrbracket$ is a partial continuous function $\llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$.*

Proof: The proof is by structural induction on the derivation of $\Gamma \vdash t : \sigma$. The cases of the constants are all welldefined and trivially continuous as the cpos interpreting the ground types are discrete. We will treat the remaining cases:

$\Gamma \vdash x : \sigma$ The interpretation is $\rho \mapsto \rho(x)$ which is both welldefined and continuous.

$\Gamma \vdash (t_1 t_2) : \sigma$ The interpretation is $app \circ (\llbracket \Gamma \vdash t_1 : \sigma' \rightarrow \sigma \rrbracket, \llbracket \Gamma \vdash t_2 : \sigma' \rrbracket)$, where app denotes application¹. As typing is deterministic, the induction hypothesis apply, and we find ourselves with a composition of welldefined continuous functions.

$\Gamma \vdash \lambda x : \sigma'. t : \sigma' \rightarrow \sigma''$ The interpretation takes ρ to the function

$$v \mapsto \llbracket \Gamma, x : \sigma' \vdash t : \sigma'' \rrbracket \rho[v/x].$$

By the induction hypothesis and lemma 4.3.5, this function is continuous.

$\Gamma \vdash \mu f. t : \sigma$ First we note, that $g(0) : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$, and that if $g(n) : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$ then $g(n+1) = \rho \mapsto \llbracket \Gamma, f : \widehat{\sigma} \vdash t : \sigma \rrbracket \rho(g(n)\rho)$ is also a partial function from $\llbracket \Gamma \rrbracket$ to $\llbracket \sigma \rrbracket$.

Now, as in the previous case, the function $g' = \llbracket \Gamma, f : \widehat{\sigma} \vdash t : \sigma \rrbracket$ is continuous. We then note that for all n

$$g(n+1)\rho = g'(\rho)(g(n)\rho).$$

So, as g' is total, $g(n)\rho \downarrow \Rightarrow g(n+1)\rho \downarrow$. Now, since $\forall \rho. g(0)\rho \uparrow$, $g(0) \leq g(1)$. Thus g' being continuous ensures $g(1) \leq g(2)$ yielding $g(2) \leq g(3)$ and so on. Consequently $\bigsqcup_{n \in \omega} g(n)$ is a welldefined continuous function $\llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$.

□

Lemma 4.3.7 (Substitution Lemma). *Let $s : \sigma$ be a closed term, then*

$$\llbracket \Gamma, x : \sigma \vdash t : \sigma' \rrbracket \rho[\llbracket \Gamma \vdash s : \sigma \rrbracket \rho/x] = \llbracket \Gamma \vdash t[s/x] : \sigma' \rrbracket \rho$$

Proof: A tedious structural induction. □

Example 4.3.8. *We now show that $\llbracket \emptyset \vdash fac : \iota \rightarrow \iota \rrbracket$ is in fact the faculty function, assuming that the interpretation of $a \times b$ is the interpretation of a times the interpretation of b . Recall the definition*

$$\begin{aligned} fac &\equiv \mu f. \lambda x : \iota. \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)), \\ \llbracket \emptyset \vdash fac : \iota \rightarrow \iota \rrbracket \rho &= \bigsqcup_n g(n)\rho \end{aligned}$$

where

$$g(0)\rho = \perp_{\llbracket \iota \rightarrow \iota \rrbracket},$$

the everywhere undefined function from \mathbb{N} to \mathbb{N} .

$$\begin{aligned} g(1)\rho &= \llbracket f : \iota \rightarrow \iota \vdash \lambda x : \iota. \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)) \rrbracket [\perp_{\llbracket \iota \rightarrow \iota \rrbracket} / f] \\ &= v \mapsto \llbracket f : \iota \rightarrow \iota, x : \iota \vdash \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)) \rrbracket [\perp_{\llbracket \iota \rightarrow \iota \rrbracket} / f, v/x] \\ &= v \mapsto \text{if } v = 0 \text{ then } 1 \text{ else } v \cdot \perp_{\llbracket \iota \rightarrow \iota \rrbracket}(v-1) \\ \\ g(2)\rho &= \llbracket f : \iota \rightarrow \iota \vdash \lambda x : \iota. \text{cond}^t(\text{zero? } x) \bar{1}(x \times f(\text{pred } x)) \rrbracket [g(1)\rho / f] \\ &= v \mapsto \text{if } v = 0 \text{ then } 1 \text{ else } v \cdot g(1)(v-1) \\ \\ &\vdots \end{aligned}$$

As n increases the functions $g(n)$ expand their domains. Let $\varphi = \bigsqcup_n g(n)\rho : \mathbb{N} \rightarrow \mathbb{N}$. To see that φ is the faculty function, we need to show that

¹Application is known to be continuous.

$$1. \varphi(n) \downarrow, \quad \forall n \in \mathbb{N}$$

$$2. \varphi(0) = 1$$

$$3. \varphi(n+1) = (n+1)\varphi(n)$$

1. We show by induction that for all n , $g(n+1)(n) \downarrow$, then because g is a chain, $\varphi(n) \downarrow$ too. The case $n = 0$ is immediate by the above calculation, $g(1)(0) = 1$. $g(n+2)(n+1) = (n+1)g(n+1)(n)$, which is defined as $g(n+1)(n) \downarrow$ by induction hypothesis.

2. It is easy to see that $g(n)(0) = 1$ for all n , so $\varphi(0) = 1$ too, as φ is the least upper bound.

3. We show by induction on n that for all $n > 0$, and for all $i < n$, that $g(n)(i) = g(n+1)(i)$.

- $n = 1$: By the above calculation, $g(1)(0) = 1 = g(2)(0)$.
- $n + 1$: $g(n+1)(i) = ig(n)(i-1)$ by the interpretation, and $ig(n)(i-1) = g(n+1)(i-1)$ by the induction hypothesis, so $g(n+2)(i) = ig(n+1)(i-1) = g(n+1)(i)$.

From this we see that $g(n+1)(i) = ig(n+1)(i-1)$ for all n and $0 < i < n$, so $\forall i > 0. \varphi(i) = i\varphi(i-1)$ too.

4.4 pCpo gives an adequate model for PCF-call-by-value

We have given two semantics for the language PCF. One of the purposes of a semantics is to define what the meaning (value) of a program should be, so we would very much like our two semantics to agree – at least on programs – and preferably on all typeable terms. We shall show the former and realize that the latter does not hold for our semantics.

Proposition 4.4.1 (Operational soundness). *For any closed term $t : \sigma$,*

$$t \Rightarrow c \text{ implies } \llbracket \Gamma \vdash t : \sigma \rrbracket = \llbracket \Gamma \vdash c : \sigma \rrbracket,$$

where c is some canonical form, and Γ is any context such that t is typeable in Γ .

Proof: By structural induction on the rules for the operational semantics. We just pick out a few cases. Consider the case

$$\frac{t \Rightarrow \bar{n}}{\text{succ } t \Rightarrow \overline{n+1}} :$$

the induction hypothesis is that $\llbracket \Gamma \vdash t : \iota \rrbracket = \llbracket \Gamma \vdash \bar{n} : \iota \rrbracket$. By the denotational semantics we have

$$\begin{aligned} \llbracket \Gamma \vdash \text{succ } t : \iota \rrbracket \rho &= (x \mapsto x+1) \llbracket \Gamma \vdash t : \iota \rrbracket \rho \\ &= (x \mapsto x+1)n && \text{by induction hypothesis} \\ &= n \end{aligned}$$

$$\frac{t_1 \Rightarrow \lambda x : \sigma. t'_1 \quad t_2 \Rightarrow c_2 \quad t'_1[c_2/x] \Rightarrow c}{(t_1 t_2) \Rightarrow c} :$$

We reason as follows:

$$\begin{aligned}
\llbracket \Gamma \vdash (t_1 t_2) : \sigma' \rrbracket \rho &= \llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho (\llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho) && \text{by denotational semantics} \\
&= \llbracket \Gamma \vdash \lambda x : \sigma. t'_1 : \sigma \rightarrow \sigma' \rrbracket \rho (\llbracket \Gamma \vdash c_2 : \sigma \rrbracket \rho) && \text{by induction hypothesis} \\
&= (v \mapsto \llbracket \Gamma, x : \sigma \vdash t'_1 : \sigma' \rrbracket [v/x]) (\llbracket \Gamma \vdash c_2 : \sigma \rrbracket \rho) && \text{by den. semantics} \\
&= \llbracket \Gamma, x : \sigma \vdash t'_1 : \sigma' \rrbracket [\llbracket \Gamma \vdash c_2 : \sigma \rrbracket \rho / x] \\
&= \llbracket \Gamma, x : \sigma \vdash t'_1 [c_2/x] : \sigma' \rrbracket \rho && \text{by Substitution Lemma 4.3.7} \\
&= \llbracket \Gamma \vdash c : \sigma' \rrbracket \rho && \text{by induction hypothesis.}
\end{aligned}$$

$$\frac{t[\mu f.t/f] \Rightarrow c}{\mu f.t \Rightarrow c} :$$

$$\begin{aligned}
\llbracket \Gamma \vdash c : \sigma \rrbracket &= \llbracket \Gamma \vdash t[\mu f.t/f] : \sigma \rrbracket \rho && \text{by induction hypothesis} \\
&= \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho [\llbracket \Gamma \vdash \mu f.t : \sigma \rrbracket \rho / f] && \text{by Substitution Lemma 4.3.7} \\
&= \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho [\bigsqcup_n g(n) \rho / f]
\end{aligned}$$

We need to show that

$$\bigsqcup_n g(n) \rho = \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho [\bigsqcup_n g(n) \rho / f].$$

This is not hard. Recall the function

$$g' : [\Gamma] \rightarrow ([\sigma] \rightarrow [\sigma])$$

defined by

$$g'(\rho)(v) = \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho [v/f],$$

which we showed was continuous in the proof of Proposition 4.3.6. Now,

$$\begin{aligned}
\bigsqcup_n g(n) \rho &= \bigsqcup_n g(n+1) \rho \\
&= \bigsqcup g'(\rho)(g(n) \rho) \\
&= g'(\rho)(\bigsqcup_n g(n) \rho) && \text{by continuity of } g'(\rho) \\
&= \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho [\bigsqcup_n g(n) \rho / f].
\end{aligned}$$

□

It would have been nice if $\llbracket \emptyset \vdash t : \sigma \rrbracket = \llbracket \emptyset \vdash c : \sigma \rrbracket$ implies $t \Rightarrow c$ were true. To see that it is not, consider the terms $\lambda x.x : \iota \rightarrow \iota$ and $\lambda x.\text{pred}(\text{succ } x) : \iota \rightarrow \iota$. Then $\llbracket \emptyset \vdash \lambda x.x : \iota \rightarrow \iota \rrbracket = n \mapsto n = \llbracket \emptyset \vdash \lambda x.\text{pred}(\text{succ } x) : \iota \rightarrow \iota \rrbracket$, but since they are both canonical forms, neither of them evaluates to the other.

The relationship between the semantics that we *do* have and will show in a moment is that termination in the denotational semantics matches definedness in the operational one.

Theorem 4.4.2 (Adequacy). *For all typeable closed terms t .*

$$\llbracket \Gamma \vdash t : \sigma \rrbracket \rho \downarrow \text{ implies } \exists c.t \Rightarrow c,$$

for a canonical form c .

Before we give the proof of adequacy, we define a relation between elements of $[\sigma]$ and closed terms of type σ , which is a so called logical relation. The point is that a straight forward structural induction will fail, so we strengthen our induction hypothesis through this logical relation.

We use environments to assign values (elements of a cpo) to free variables, we now define the notion of a substitution which assigns closed terms to free variables. A substitution is thus purely syntactical.

Definition 4.4.3. A substitution $\eta : \text{Var} \rightarrow \text{ClosedTerms}$ is a finite mapping. We have an updating function:

$$\eta[s/x](y) \stackrel{\text{def}}{=} \begin{cases} s & \text{if } y = x \\ \eta(y) & \text{else} \end{cases}$$

And η induces a mapping $\hat{\eta} : \text{Terms} \rightarrow \text{Terms}$, defined as follows:

$$\begin{aligned} \hat{\eta}(x) &= \eta(x) \\ \hat{\eta}(t) &= t && \text{for any closed term } t \\ \hat{\eta}(t_1 t_2) &= \hat{\eta}(t_1) \hat{\eta}(t_2) \\ \hat{\eta}(\lambda x.t) &= \lambda x. \hat{\eta}[x/x](t) \\ \hat{\eta}(\mu f.t) &= \mu f. \hat{\eta}[f/f](t) \end{aligned}$$

Lemma 4.4.4. For all terms t ,

$$\text{FV}(t) \subseteq \text{dom}(\eta) \Rightarrow \hat{\eta}(t) \text{ is closed.}$$

Proof: By structural induction on t :

$t \equiv x$: x must be in $\text{dom}(\eta)$, so $\hat{\eta}(t) = \eta(x)$ which is closed.

$t \equiv (t_1 t_2)$: The free variables of t_1 and t_2 are all free in $(t_1 t_2)$, so the induction hypothesis apply. Then, as t_1 and t_2 are closed, $(t_1 t_2)$ is closed.

$t \equiv \lambda x : \sigma. t_1$: x is the only variable possible free in $\hat{\mu}[x/x](t_1)$, and so $\lambda x : \sigma. \hat{\mu}[x/x](t_1)$ must be closed.

$t \equiv \mu f : \sigma. t_1$: Just like the previous case.

□

Definition 4.4.5.

- For all $d \in \llbracket \sigma \rrbracket$ or d undefined and closed terms $t : \sigma$, $d \lesssim_\sigma t$ iff $d \downarrow \Rightarrow (\exists c. t \Rightarrow c \wedge d \lesssim_\sigma c)$.
- For all $d \in \llbracket \sigma \rrbracket$ and canonical forms $c : \sigma$, $d \lesssim_\sigma c$ is defined by the following:
 - $n \lesssim_\iota \bar{n}, \forall n$,
 - $\text{true} \lesssim_o \text{tt}$, $\text{false} \lesssim_o \text{ff}$,
 - $\phi \lesssim_{\sigma_1 \rightarrow \sigma_2} t$ iff $\forall v \in \llbracket \sigma_1 \rrbracket, \forall c. v \lesssim_{\sigma_1} c \Rightarrow \phi(v) \lesssim_{\sigma_2} (tc)$.
- For all environments ρ and substitutions η , $\rho \lesssim_\Gamma \eta$ iff $\rho(x) \lesssim_{\Gamma(x)} \eta(x)$, for all $x \in \text{dom}(\Gamma)$.

Lemma 4.4.6. Let $t : \sigma$

1. $d \uparrow \Rightarrow d \lesssim_\sigma t$
2. if $d \sqsupseteq d'$ and $d' \lesssim_\sigma t$ then $d \lesssim t$.
3. If $d_0 \sqsupseteq d_1 \sqsupseteq \dots$ is a chain in $\llbracket \sigma \rrbracket$ such that $d_n \lesssim_\sigma t$ for all n then $\bigsqcup_n d_n \lesssim_\sigma t$.

For a proof of this see [Win93, p.194].

Lemma 4.4.7. *Suppose t is a typeable term, i.e., $\Gamma \vdash t : \sigma$. Then $\rho \lesssim_{\Gamma} \eta$ implies $\llbracket \Gamma \vdash t : \sigma \rrbracket \rho \lesssim_{\sigma} \hat{\eta}(t)$, for $\rho \in \llbracket \Gamma \rrbracket$.*

Proof: Note that by Lemma 4.4.4, if $\llbracket \Gamma \vdash t : \sigma \rrbracket \rho \downarrow$ then the term $\hat{\eta}(t)$ is closed, because $\llbracket \Gamma \vdash t : \sigma \rrbracket \rho \downarrow$ implies $\text{FV}(t) \subseteq \text{dom}(\rho)$ and since $\rho \lesssim_{\Gamma} \eta$ also $\text{FV}(t) \subseteq \text{dom}(\eta)$, so the claim does indeed make sense. By structural induction on type derivations.

$\Gamma \vdash x : \sigma$: Then $\Gamma(x) = \rho$ by induction hypothesis, and

$$\llbracket \Gamma \vdash x : \sigma \rrbracket \rho = \rho(x) \lesssim_{\sigma} \hat{\eta}(x) = \eta(x).$$

$\Gamma \vdash \bar{n} : \iota$:

$$\llbracket \Gamma \vdash \bar{n} : \iota \rrbracket \rho = n \lesssim_{\iota} \bar{n} = \hat{\eta}(\bar{n}).$$

$\Gamma \vdash \text{tt} : o$:

$$\llbracket \Gamma \vdash \text{tt} : o \rrbracket \rho = \text{true} \lesssim_o \text{tt}.$$

$\Gamma \vdash \text{ff} : o$:

$$\llbracket \Gamma \vdash \text{ff} : o \rrbracket \rho = \text{false} \lesssim_o \text{ff}.$$

$\Gamma \vdash \text{pred} : \iota \rightarrow \iota$:

$$\llbracket \Gamma \vdash \text{pred} : \iota \rightarrow \iota \rrbracket \rho = x \mapsto x \dot{-} 1,$$

now suppose $v \lesssim_{\iota} c$, i.e., either $0 \lesssim_{\iota} \bar{0}$ or $n + 1 \lesssim_{\iota} \overline{n + 1}$ for some n . In the latter case we get $(x \mapsto x \dot{-} 1)n + 1 = n \lesssim_{\iota} \text{pred } \overline{n + 1}$ holds since $\text{pred } \overline{n + 1} \Rightarrow \bar{n}$. The former case is similar.

$\Gamma \vdash \text{succ} : \iota \rightarrow \iota$:

$$\llbracket \Gamma \vdash \text{succ} : \iota \rightarrow \iota \rrbracket \rho = x \mapsto x + 1.$$

Suppose $n \lesssim_{\iota} \bar{n}$, since $(x \mapsto x \dot{-} 1)n = n + 1$ and $\text{succ } \bar{n} \Rightarrow \overline{n + 1}$, we get $\llbracket \Gamma \vdash \text{succ} : \iota \rightarrow \iota \rrbracket \rho \lesssim_{\iota \rightarrow \iota} \text{succ}$

$\Gamma \vdash \text{zero?} : \iota \rightarrow o$: This case is similar to the above.

$\Gamma \vdash \text{cond}^{\beta} : o \rightarrow (\beta \rightarrow (\beta \rightarrow \beta))$:

$$\llbracket \Gamma \vdash \text{cond}^{\beta} : o \rightarrow (\beta \rightarrow (\beta \rightarrow \beta)) \rrbracket \rho = x_1 \mapsto (x_2 \mapsto (x_3 \mapsto \text{if } x_1 \text{ then } x_2 \text{ else } x_3)).$$

Again we have two cases: $\text{true} \lesssim_o \text{tt}$ then we must show that $x_2 \mapsto (x_3 \mapsto x_2) \lesssim_{\beta \rightarrow (\beta \rightarrow \beta)} \text{cond}^{\beta} \text{tt}$. To see that it holds, assume that $v_2 \lesssim_{\beta} c_2$ and $v_3 \lesssim_{\beta} c_3$, then

$$x_2 \mapsto (x_3 \mapsto x_2)v_2v_3 \lesssim_{\beta} \text{cond}^{\beta} \text{tt}c_2c_3$$

holds because $x_2 \mapsto (x_3 \mapsto x_2)v_2v_3 = v_2$ and $\text{cond}^{\text{tt}} c_2c_3 \Rightarrow c_2$. The case $\text{false} \lesssim_o \text{ff}$ is similar.

$\Gamma \vdash (t_1 t_2) : \sigma'$: By induction hypothesis we have

$$\llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho \lesssim_{\sigma \rightarrow \sigma'} \hat{\eta}(t_1)$$

and

$$\llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho \lesssim_{\sigma} \hat{\eta}(t_2)$$

which means that there are canonical forms c_1, c_2 , (where c_1 has the form $\lambda x:\sigma.c'_1$) such that

$$\begin{aligned}\hat{\eta}(t_1) &\Rightarrow c_2 & \llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho &\lesssim_{\sigma \rightarrow \sigma'} c_1 \\ \hat{\eta}(t_2) &\Rightarrow c_2 & \llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho &\lesssim_{\sigma} c_2.\end{aligned}$$

This implies that

$$\llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho (\llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho) \lesssim_{\sigma'} c_1 c_2.$$

Now if we can show that $\hat{\eta}(t_1 t_2) \Rightarrow c_1 c_2$, we are done, but this follows from the operational semantics plus the fact that $\hat{\eta}(t_1 t_2) = \hat{\eta}(t_1) \hat{\eta}(t_2)$.

$\Gamma \vdash \lambda x:\sigma.t : \sigma \rightarrow \sigma' :$

$$\llbracket \Gamma \vdash \lambda x:\sigma.t : \sigma \rightarrow \sigma' \rrbracket \rho = v \mapsto \llbracket \Gamma, x : \sigma \vdash t : \sigma' \rrbracket \rho[v/x]$$

by definition. Assume $v_1 \lesssim_{\sigma} c_1$, then $\rho[v_1/x] \lesssim_{\Gamma, x:\sigma} \widehat{eta}[c_1/x]$ since, by assumption, $\rho \lesssim_{\Gamma} \eta$. By the induction hypothesis, we get

$$\llbracket \Gamma, x : \sigma \vdash t : \sigma' \rrbracket \rho[v_1/x] \lesssim_{\sigma'} \widehat{\eta}[c_1/x](t),$$

i.e, there is a canonical form c_2 such that $\widehat{\eta}[c_1/x](t) \Rightarrow c_2$ and $\llbracket \Gamma, x : \sigma \vdash t : \sigma' \rrbracket \rho[v_1/x] \lesssim_{\sigma'} c_2$. To see that $\hat{\eta}(\lambda x:\sigma.t)c_1 \Rightarrow c_2$ consider the following from the operational semantics:

$$\frac{\lambda x:\sigma.\widehat{\eta}[x/x](t) \Rightarrow \lambda x:\sigma.\widehat{\eta}[x/x](t) \quad c_1 \Rightarrow c_1 \quad (\widehat{\eta}[x/x](t))[c_1/x] = \widehat{\eta}[c_1/x](t) \Rightarrow c_2}{\lambda x:\sigma.\widehat{\eta}[x/x](t)c_1 = \hat{\eta}(\lambda x:\sigma.t)c_1 \Rightarrow c_2}.$$

$\Gamma \vdash \mu f.t : \sigma :$ This is probably the most interesting case. Recall that $\llbracket \Gamma \vdash \mu f.t : \sigma \rrbracket \rho = \bigsqcup_n g(n)\rho$, where $g(0)\rho \uparrow$ and $g(n+1)\rho = \llbracket \Gamma, f : \sigma \vdash t_{\sigma} \rrbracket \rho[g(n)\rho/f]$, where $\rho \lesssim_{\Gamma} \eta$. We show by induction on n , that for all n , $g(n)\rho \lesssim_{\sigma} \hat{\eta}(\mu f.t)$, then by Lemma 4.4.6, $\bigsqcup_n g(n) \lesssim_{\sigma} \hat{\eta}(\mu f.t)$. $g(0)\rho \uparrow$ so $g(0)\rho \lesssim_{\sigma} \hat{\eta}(\mu f.t)$. By induction hypothesis we have $g(n)\rho \lesssim_{\sigma} \hat{\eta}(\mu f.t)$ for all $\rho \lesssim_{\Gamma} \eta$. Let $\rho \in \llbracket \Gamma \rrbracket$ and $\rho \lesssim_{\Gamma} \eta$ then $\rho[g(n)\rho/f] \lesssim_{\Gamma, f:\sigma} \eta[\hat{\eta}(\mu f.t)/f]$, so by the main induction hypothesis we have

$$\llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho[g(n)\rho/f] \lesssim_{\sigma} \eta[\widehat{\eta}(\mu f.t)/f](t).$$

To see that

$$g(n+1)\rho \lesssim_{\sigma} \hat{\eta}(\mu f.t),$$

we reason as follows:

$$\begin{aligned}g(n+1)\rho &= \\ \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket \rho[g(n)\rho/f] &\lesssim_{\sigma} \eta[\widehat{\eta}(\mu f.t)/f](t) && \text{by the above} \\ &= \widehat{\eta}[f/f](t[\mu f.t/f]) \\ &= (\widehat{\eta}[f/f](t))[\widehat{\eta}(\mu f.t)/f] \\ &= (\widehat{\eta}[f/f](t))[\mu f.(\widehat{\eta}[f/f](t))/f] && \text{by definition of } \hat{\eta},\end{aligned}$$

so there is a c such that

$$(\widehat{\eta}[f/f](t))[\mu f.(\widehat{\eta}[f/f](t))/f] \Rightarrow c$$

and

$$g(n+1)\rho \lesssim_{\sigma} c.$$

By operational semantics then $\mu f.\widehat{\eta}[f/f](t) = \hat{\eta}(\mu f.t) \Rightarrow c$ too and we are done.

Now adequacy follows from the definition of \lesssim_σ . □

Corollary 4.4.8. *For wellformed terms $\Gamma \vdash t : \beta$ of ground type,*

$$t \Rightarrow c \text{ iff } \llbracket \Gamma \vdash t : \beta \rrbracket = \llbracket c \rrbracket.$$

Proof: The “only if” part follows from operational soundness (Prop.4.4.1). The converse direction follows from the fact that if two canonical terms of type o or type ι have the same denotation, then they must be identical. □

Chapter 5

Call-by-name

We now give a brief summary of the differences between the Call-by-name and the call-by-value semantics of PCF, and we also state an adequacy result for the call-by-name semantics.

The terms, the canonical forms and the type system are the same as in call-by-value.

Call-by-name operational semantics for PCF Only difference is the function application rule, which becomes

$$\frac{t_1 \Rightarrow \lambda x : \sigma. t'_1 \quad t'_1[t_2/x]ac}{(t_1 t_2) \Rightarrow c} .$$

Call-by-name denotational semantics for PCF Types are interpreted as objects of Cppo as follows:

- $\llbracket \iota \rrbracket = \mathbb{N}_\perp$
- $\llbracket \rho \rrbracket = \mathbb{T}_\perp$
- $\llbracket \sigma_1 \rightarrow \sigma_2 \rrbracket = \llbracket \sigma_1 \rrbracket \rightarrow \llbracket \sigma_2 \rrbracket$

Where \mathbb{N}_\perp and \mathbb{T}_\perp are the cpos defined in Section 4.3 now with a bottom element added. $\llbracket \sigma_1 \rrbracket \rightarrow \llbracket \sigma_2 \rrbracket$ is the cppo $\llbracket \sigma_2 \rrbracket^{\llbracket \sigma_1 \rrbracket}$ of continuous functions from $\llbracket \sigma_1 \rrbracket$ to $\llbracket \sigma_2 \rrbracket$. (Here we use the fact that Cppo has exponentials, being a full sub-ccc of Cpo as stated in Theorem 3.0.12).

An interpretation $\llbracket \Gamma \vdash t : \sigma \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket$ should be a morphism in the category Cppo, i.e., a total continuous function. Therefore environments have to be defined for all $x \in \text{dom}(\Gamma)$, so

$$\llbracket \Gamma \rrbracket = \{\rho \mid \forall x \in \text{dom}(\Gamma). \rho(x) \in \llbracket \Gamma(x) \rrbracket\}.$$

$\llbracket \Gamma \rrbracket$ is a cppo. The denotational semantics is the same as in Section 4.3, except for recursion as it uses partial functions. Instead we use the fact that Cppo has fixed points by Theorem 3.0.13.

Given $\varphi : \llbracket \Gamma, f : \sigma \rrbracket \rightarrow \llbracket \sigma \rrbracket$, we define $\tilde{\varphi} : \llbracket \Gamma \rrbracket \rightarrow \llbracket \sigma \rrbracket^{\llbracket \sigma \rrbracket}$ by

$$\tilde{\varphi}(\rho) = v \mapsto \varphi(\rho[v/f]),$$

we claim without proof that this defines a continuous function if φ is continuous. The interpretation of recursion is then

$$\llbracket \Gamma \vdash \mu f. t : \sigma \rrbracket \rho = \bigsqcup_n \tilde{\varphi}^n(\rho)(\perp_{\llbracket \sigma \rrbracket}),$$

where

$$\varphi = \llbracket \Gamma, f : \sigma \vdash t : \sigma \rrbracket : \llbracket \Gamma, f : \sigma \rrbracket \rightarrow \llbracket \sigma \rrbracket.$$

Remark 5.0.9. *A notable difference between the denotational semantics is in function application $\Gamma \vdash (t_1 t_2) : \sigma'$. In the call-by-value evaluation both $\llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket$ and $\llbracket \Gamma \vdash t_2 : \sigma \rrbracket$ are partial functions, so if one of them is undefined for an environment $\rho \in \llbracket \Gamma \rrbracket$, the interpretation $\llbracket \Gamma \vdash (t_1 t_2) : \sigma' \rrbracket \rho = \llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho (\llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho)$ is undefined. In the call-by-name evaluation, however, $\llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket$ is a morphism of Cppo, so it is total and not necessarily strict, meaning that even if $\llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho = \perp_{\llbracket \sigma \rrbracket}$, $\llbracket \Gamma \vdash t_1 : \sigma \rightarrow \sigma' \rrbracket \rho (\llbracket \Gamma \vdash t_2 : \sigma \rrbracket \rho)$ can still give a value different from \perp . This corresponds nicely with our intuitive ideas: We think of \perp as representing the undefined, and in a call-by-name evaluation, a function may be defined even if some of its argument are not, if for instance the arguments are not used in the evaluation of the function. In the call-by-value evaluation we always evaluate all arguments before we evaluate the function, so if an argument is undefined, the whole evaluation must be undefined. These ideas are reflected in the denotational semantics as well as in the operational semantics.*

5.1 Cppo gives an adequate model for PCF-call-by-name

We state the following results without proofs. The proofs are by the same principles as in the previous section and can be found in e.g. [Win93, p.204 ff].

Proposition 5.1.1 (Operational soundness). *For any closed term $t : \sigma$,*

$$t \Rightarrow c \text{ implies } \llbracket \Gamma \vdash t : \sigma \rrbracket = \llbracket \Gamma \vdash c : \sigma \rrbracket,$$

where c is some canonical form, and Γ is any context such that t is typeable in Γ .

Theorem 5.1.2 (Adequacy). *For all typeable closed terms t .*

$$\llbracket \Gamma \vdash t : \sigma \rrbracket \rho \downarrow \text{ implies } \exists c. t \Rightarrow c,$$

for a canonical form c .

We have seen how the models must incorporate structure suitable to interpret constructs such as type expressions and fixed points. If we were to consider a richer programming language with, for instance, product types (pairs) and exceptions, then our model would have to contain products and coproducts.

Bibliography

- [HP] H. Huwig and A. Poigne. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73(1):101–112.
- [Jac] Bart Jacobs. Semantics of weakening and contraction.
- [Oos] J. V. Oosten. Basic category theory.
- [Win93] Glynn Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, 1993.