Full Abstraction for PCF

Thijs Alkemade – 3285960

Master Thesis Mathematical Sciences

Universiteit Utrecht

Supervised by: Jaap van Oosten
Second reader: Wouter Swierstra

August 18, 2015

Contents

1 Preliminaries **3**
- 1.1 Untyped λ-calculus . . . 3
  - 1.1.1 β-reduction . . . 4
  - 1.1.2 η-equivalent . . . 5
  - 1.1.3 The relation $\twoheadrightarrow_\beta$ . . . 6
  - 1.1.4 Computability . . . 7
- 1.2 Simply typed λ-calculus . . . 7
- 1.3 Category theory . . . 9
- 1.4 From λ-calculus theories to Cartesian closed categories . . . 14
- 1.5 Domains . . . 16
- 1.6 Models of simply typed λ-calculus . . . 27

2 PCF **34**
- 2.1 Introduction . . . 34
- 2.2 Operational semantics for PCF . . . 35
- 2.3 Denotational semantics for PCF . . . 39
- 2.4 Scott-domain model . . . 40
- 2.5 The Full Abstraction Problem . . . 42
  - 2.5.1 Strengthening: parallel conditional . . . 42
  - 2.5.2 The stable function model . . . 43

3 Dialogue Games **45**
- 3.1 Example arenas . . . 46
- 3.2 O-view and P-view . . . 46
- 3.3 Product and function space arena . . . 47
- 3.4 Innocent strategies . . . 50
- 3.5 The model . . . 55
- 3.6 FCF . . . 56

A Proof of the Church-Rosser theorem **62**

B Proof of the weak normalization of the simply typed λ-calculus **66**

Symbols **70**

Chapter 1

Preliminaries

During the 1920s and 1930s a lot of interest in defining a mathematical model of computability emerged. It is easy to show that something can be computed by an algorithm by giving the algorithm, but proving that something can *not* be computed requires a precise mathematical definition of what algorithms can do. The operations that an algorithm is allowed to use had not been properly formalized yet.

1.1 Untyped λ-calculus

One of the models that was introduced was the *untyped lambda calculus*. It was developed by Alonzo Church in the 1930s. The syntax of λ-terms is given by the following grammar, from [Barendregt, 1984]:

T	::=	x	(Variable)
	|	$(\lambda x.T)$	(λ-abstraction, where x is a variable)
	|	$(T\ T)$	(Application of two λ-terms)

For example, these are λ-terms:

$$x$$
$$(\lambda x.x)$$
$$((\lambda x.y)\ y)$$
$$(\lambda x.(x\ x)\ \lambda x.(x\ x))$$

We shall use $x, y, \ldots$ to denote variables and $M, N, \ldots$ to denote λ-terms. When parentheses are omitted, application is assumed to be left-associative, so $M\ N\ O$ is equal to $((M\ N)\ O)$. Parentheses around λ-abstractions will be omitted when the result is unambiguous.

1.1.1 β-reduction

To define how the evaluation of these computations works, we must first define what the free variables in a λ-term are.

Definition 1.1 *The free variables of a λ-term are given by:*

$$\begin{aligned} fv(x) &= \{x\} \\ fv(\lambda x.M) &= fv(M) - \{x\} \\ fv(M\ N) &= fv(M) \cup fv(N) \end{aligned}$$

Definition 1.2 *We call a term M* closed *if it has no free variables, so $fv(M) = \emptyset$. A variable in a term that is not free is* bound *by a λ-abstraction.*

A variable is bound at most once. For example in:

$$\lambda x.(\lambda x.x)$$

the variable x is bound by the inner λ-abstraction.

Definition 1.3 *Evaluation of λ-terms works by applying a rewrite step called β-reduction:*

$$(\lambda x.M)\ N \to_\beta M[x := N]$$

By $M[x := N]$ we mean syntactically replacing every occurrence of x bound by the λ-abstraction by a literal copy of N. This can only be done if none of the free variables of N would get bound at any occurrence of x, otherwise the meaning of those variables would change. If this is not possible, those bindings have to be α-renamed first.

Definition 1.4 *The α-*renaming *rewrite step assigns a new name to all variables bound by a λ-abstraction:*

$$\lambda x.M \to_\alpha \lambda y.M[x := y]$$

This is only allowed if at no location y gets bound by a different λ-abstraction. For example, this is not allowed:

$$\lambda x.(\lambda y.x) \to_\alpha \lambda y.(\lambda y.y)$$

Definition 1.5 *Two λ-terms M and N are α-*equivalent*, noted $M \equiv_\alpha N$, if it is possible to rewrite M into N using α-renaming.*

A β-reduction is not only allowed at the outer level, anywhere inside a term where $(\lambda x.M)\ N$ (a *redex*) occurs, β-reduction is possible. We can formulate this as the following recursive relation:

$$\frac{}{(\lambda x.M)\ N \to_\beta M[x := N]}$$

$$\frac{M \to_\beta M'}{\lambda x.M \to_\beta \lambda x.M'}$$

$$\frac{M \to_\beta M'}{M\ N \to_\beta M'\ N}$$

$$\frac{N \to_\beta N'}{M\ N \to_\beta M\ N'}$$

The first, third and fourth rule may overlap: in an application $M\ N$ it is possible both M and N can be β-reduced, so the evaluation may have to "choose" between different options and is therefore not deterministic.

Definition 1.6 *Given a relation $\to$, the* transitive closure *of $\to$, denoted $\twoheadrightarrow$ is defined as $a \twoheadrightarrow b = \exists x_i.x_1 \to \ldots \to x_n.x_1 = a \wedge x_n = b$. Note that $a \twoheadrightarrow a$, even for non-reflexive relations $\to$.*

1.1.2 η-equivalent

We can also reduce terms in a different way: by *η-reduction*. This removes a redundant λ-abstraction and application from a term. It is defined by:

$$\lambda x.Mx \to_\eta M$$

if $x \notin \mathrm{fv}(M)$.

Definition 1.7 *Two terms M and N are* η-equivalent *(denoted $M \equiv_\eta N$) if there is an O such that $M \twoheadrightarrow_\eta O$ and $N \twoheadrightarrow_\eta O$.*

We say that a term is in *normal form* if it can not be reduced any further by a reduction relation. Relations can be *non-normalizing*, *weakly normalizing* or *strongly normalizing*. If a relation is non-normalizing, then there are terms that will never reach a normal form, regardless of the order of reductions. A relation is weakly normalizing if, for every term, there is a sequence of reductions that reaches a normal form. A relation is strongly normalizing if every sequence of reductions reaches a normal form.

It is easy to see that λ-terms are strongly normalizing under $\to_\eta$: when $M \to_\eta N$, then N is strictly smaller (by considering the number of symbols) than M.

1.1.3 The relation $\twoheadrightarrow_\beta$

The relation $\twoheadrightarrow_\beta$ is the transitive closure of $\rightarrow_\beta$.

From now on, we will use "normal form" for a normal form using the $\rightarrow_\beta \cup \rightarrow_\eta$ relation. Not all λ-terms eventually reach a normal form, for example:

$$(\lambda x.(x\ x)\ \lambda x.(x\ x)) \rightarrow_\beta (\lambda x.(x\ x)\ \lambda x.(x\ x)) \rightarrow_\beta \ldots \tag{1.1}$$

This term has no normal form, as the only reduction possible yields the same term.

Sometimes, terms may reach a normal form, depending which redex is chosen:

$$((\lambda x.\lambda y.y)\ (\lambda x.(x\ x)\ \lambda x.(x\ x)))$$

If we start evaluating the $(\lambda x.x\ x)\ (\lambda x.x\ x)$ subterm first, it will never reach a normal form, as seen in Equation (1.1). On the other hand, if we β-reduce the entire expression, we will be left with only $\lambda y.y$, which is a normal form.

So it is not guaranteed that a computation terminates regardless of evaluation strategy. However, the Church-Rosser theorem ([Church and Rosser, 1936]) states that evaluation is *confluent*:

$$M \twoheadrightarrow_\beta N \text{ and } M \twoheadrightarrow_\beta O \Rightarrow \exists PQ.N \twoheadrightarrow_\beta P \text{ and } O \twoheadrightarrow_\beta Q \text{ and } P \equiv_{\alpha\eta} Q$$

In other words: if two evaluations of a single term result in two different terms, then a third term exist such that both terms can be reduced to it, or a term $\alpha\eta$-equivalent to it.

In particular this implies that each term has at most one normal form (up to $\alpha\eta$-equivalence): If $M \twoheadrightarrow_\beta N$, $M \twoheadrightarrow_\beta N'$ and N and N' are in normal form then there must be a term N'' such that $N \twoheadrightarrow_\beta N''$ and $N' \twoheadrightarrow_\beta N''$. But N is in normal form, so $N \equiv_{\alpha\eta} N''$ and similarly $N' \equiv_{\alpha\eta} N''$, which means $N \equiv_{\alpha\eta} N'$.

For a proof of the Church-Rosser theorem, taken from [Barendregt, 1984], see Appendix A.

Definition 1.8 *We define the relation $\equiv_{\beta\eta\alpha}$ as the transitive, reflexive and symmetric closure of $\twoheadrightarrow_\beta \cup \twoheadrightarrow_\eta \cup \twoheadrightarrow_\alpha$.*

We state the following lemma from [Church and Rosser, 1936]:

Lemma 1.9 *Let $\twoheadrightarrow_{\beta\eta}$ be the union of the relations $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_\eta$. If $M \twoheadrightarrow_{\beta\eta} N$, then $M \twoheadrightarrow_\beta N' \twoheadrightarrow_\eta N$.*

1.1.4 Computability

Church developed a method to encode numbers, booleans and tuples as λ-terms. With those encodings, he showed it was possible to find an equivalent λ-function for every partial recursive function. If f is a partial recursive function, $n \in \mathbb{N}$ and x a λ-term that encodes n, then there is a λ-term M such that $M\ x$ reaches a normal form if and only if $f(n)$ terminates and the normal form of $M\ x$ encodes $f(n)$.

A different model of computations is the *Turing machine*, developed by Alan Turing also in the 1930s. It was later proved that both models are equally strong: any computation that can be described by a λ-term can be evaluated by a Turing machine, and vice versa. They both correspond to the set of partial recursive functions.

The Church-Turing hypothesis states that partial recursive functions are exactly the computable functions, meaning they correspond to the informal concept of "algorithm" (which can never be proven, as the concept is informal).

1.2 Simply typed λ-calculus

Shortly after the untyped λ-calculus, the *simply typed λ-calculus* was developed. It uses types for every λ-term, the syntax of the types is given by [Barendregt et al., 2013]:

$$\begin{array}{rcll} \tau & ::= & \sigma & \text{(Where } \sigma \text{ is from a set of } \textit{ground types}\text{)} \\ & | & (\tau \to \tau) & \end{array}$$

The $\to$ is right associative, so we may omit parenthesis from types: $\alpha \to \beta \to \gamma = (\alpha \to (\beta \to \gamma))$.

The syntax of the simply typed λ-calculus is given by [van Oosten, 2002]:

$$\frac{}{c : \tau}\ \text{A constant of type } \tau.$$

$$\frac{}{x : \tau}\ \text{A variable } x \text{ with type } \tau.$$

$$\frac{N : \tau_1 \to \tau_2 \qquad M : \tau_1}{N\ M : \tau_2}\ \text{An application.}$$

$$\frac{M : \tau_1 \qquad x : \tau_2}{\lambda x.M : \tau_2 \to \tau_1}\ \text{A } \lambda\text{-abstraction, binding the variable } x \text{ in } M.$$

Definition 1.10 *An* equality judgment *is given by:*

$$\Gamma \vdash M = N : \tau$$

where M and N are λ-terms with type τ and Γ is a finite set of variables, including all the free variables in M and N.

Definition 1.11 *A* λ-calculus theory *is a set of equality judgments* $\mathcal{T}$ *such that:*

1. $\Gamma \vdash M = N : \tau \in \mathcal{T}$ *implies* $\Delta \vdash M = N : \tau \in \mathcal{T}$ *for all* $\Gamma \subseteq \Delta$.
2. $fv(M) \vdash M = M : \tau \in \mathcal{T}$ *for every term* M.
3. *If* $\Gamma \vdash M = N : \tau \in \mathcal{T}$ *and* $\Gamma \vdash N = O : \tau \in \mathcal{T}$, *then* $\Gamma \vdash M = O : \tau \in \mathcal{T}$.
4. *If* $M(x_1, \ldots, x_n) : \tau_0$ *is a term with free variables* $x_1 : \tau_1, \ldots, x_n : \tau_n$ *and* $\Gamma \vdash N_1 = M_1 : \tau_1, \ldots, \Gamma \vdash N_n = M_n : \tau_n \in \mathcal{T}$ *then:*

$$\Gamma \vdash M[x_1 := N_1, \ldots, N_n := N_n] = M[x_1 := M_1, \ldots, x_n := M_n] : \tau_0 \in \mathcal{T}$$

5. *If* N *and* M *are* λ*-terms of type* $\tau_1 \to \tau_2$ *and* $x : \tau_1$ *a variable that occurs in neither* N *nor* M *and* $\Gamma \cup \{x\} \vdash (M\ x) = (N\ x) : \tau_2 \in \mathcal{T}$, *then* $\Gamma - \{x\} \vdash M = N : \tau_1 \to \tau_2 \in \mathcal{T}$.
6. *If* $N : \tau_1$ *and* $M : \tau_2$ *are terms and* x *a variable of type* τ_2, *then:*

$$fv(N) - \{x\} \cup fv(M) \vdash ((\lambda x.N)\ M) = N[x := M] : \tau_1 \in \mathcal{T}$$

For example, we can create a λ-calculus theory with natural numbers. We introduce a single type, ι, and constants $z : \iota$ (for 0), $s : \iota \to \iota$ (the successor) and $i : \iota \to \alpha \to (\alpha \to \alpha) \to \alpha$ for every type α (bounded recursion). Informally, the interpretation of i is:

$$i\ n\ f\ x = f^n x$$

We define the λ-calculus theory $\mathcal{T}_N$ as the smallest set of judgments such that the requirements of definition 1.11 hold and the following conditions are satisfied:

$$\begin{aligned} \mathrm{fv}(f) \cup \mathrm{fv}(x) &\vdash i\ 0\ f\ x\ = x : \alpha \in \mathcal{T} \\ \mathrm{fv}(f) \cup \mathrm{fv}(x) &\vdash i\ (s\ n)\ f\ x\ = f\ (i\ n\ f\ x) : \alpha \in \mathcal{T} \end{aligned}$$

For example, we can define addition, subtraction and multiplication with these numbers as:

$$\begin{aligned} \mathrm{add}\ a\ b\ &= i\ a\ s\ b \\ \mathrm{mul}\ a\ b\ &= i\ a\ (\mathrm{add}\ b)\ 0 \\ \mathrm{pred}\ n\ &= (i\ n\ (\lambda f.\lambda g.f\ (\lambda a.\lambda b.g\ (s\ a)\ a))\ (\lambda f.f\ z\ z))\ (\lambda a.\lambda b.b) \\ \mathrm{sub}\ a\ b\ &= i\ b\ \mathrm{pred}\ a \end{aligned}$$

(with the convention that pred $0 = 0$ and sub $a\ b = \max(0, a - b)$).

Recall that in the untyped λ-calculus terms are in general not normalizing: not all terms eventually reach a normal form. In a simply typed λ-calculus theory without constants, all terms are strongly normalizing: any sequence of β-reductions will eventually reach a unique normal form.

This means it is impossible to write a program with an infinite loop in the simply typed λ-calculus. As a Turing machine can be stuck in an infinite loop, this means the simply typed λ-calculus can no longer compute everything a Turing machine can.

See Appendix B for a proof that the simply typed λ-calculus is weakly normalizing and [Barendregt et al., 2013] for a proof that it is strongly normalizing.

Adding a constant $c : \tau$ introduces the equality judgment $\emptyset \vdash c = c : \tau$, but it can also introduce new non-trivial equality judgments, for example there could be a constant that denotes the test for zero function z, such that $\emptyset \vdash z\ 0 = \mathrm{tt} : o$. These extra equalities can make the language non-terminating, for example, by adding the fix-point function:

$$\mathrm{fix} : (\iota \to \iota) \to \iota$$

with the equation:

$$\emptyset \vdash \mathrm{fix}\ f = f\ (\mathrm{fix}\ f) : \iota$$

the requirements in definition 1.11 can still be satisfied, but the theory is not normalizing.

1.3 Category theory

We will now introduce some definitions from category theory that will be needed.

Definition 1.12 *A* category $\mathbb{C}$ *is an object consisting of:*

- *A class of* objects $ob(\mathbb{C})$.
- *A class of* morphisms *(or: arrows)* $hom(\mathbb{C})$*, where each morphism has a source and a target object, denoted as* $f : X \to Y$ *(with* $X, Y \in ob(\mathbb{C})$*).*
- *A binary operation* $\circ$ *that composes two morphisms* $f : X \to Y$ *and* $g : Y \to Z$*, such that:*
  - $f \circ (g \circ h) = (f \circ g) \circ h$.
  - *For each object* X *there is an* identity morphism id_X *such that for all morphisms* $f : X \to Y$, $id_Y \circ f = f \circ id_X = f$.

For example:

- **0** is the category with no objects or morphisms.
- **1** is the category with a single object 1 and its identity morphism.
- Set is the category where the objects are sets and morphisms functions between those sets, and $\circ$ is normal function composition.
- Grp is the category where the objects are groups and morphisms are group homomorphisms.

Definition 1.13 *Given two categories* $\mathbb{C}$ *and* $\mathbb{D}$ *a* functor F *consists of two functions* $F_0 : ob(\mathbb{C}) \to ob(\mathbb{D})$ *and* $F_1 : hom(\mathbb{C}) \to hom(\mathbb{D})$*, such that:*

$$F_1(g \circ f) = F_1(g) \circ F_1(f)$$
$$\forall X \in ob(\mathbb{C}).F_1(id_X) = id_{F_0(X)}$$

- For example, there is the *forgetful functor* that maps Grp to Set:

$$F_0((G, \cdot)) = G$$
$$F_1(f) = f$$

  It is forgetful as it omits the group operation.

- For every category $\mathbb{C}$ there is a unique functor $\mathbb{C} \to \mathbf{1}$:

$$F_0(X) = 1$$
$$F_1(f) = \mathrm{id}_1$$

- For every category $\mathbb{C}$ there is a unique functor $\mathbf{0} \to \mathbb{C}$ (where F_0 and F_1 are functions with empty domains).
- We can also compose functors and for every category an identity functor exists, which means that we can make a category Cat where the objects are categories and the morphisms are functors.

Definition 1.14 *A* product $X \times Y$ *in a category* $\mathbb{C}$ *is a diagram with:*

- *Distinguished* projection morphisms $\pi_1 : X \times Y \to X$, $\pi_2 : X \times Y \to Y$. *See Figure 1.1.*

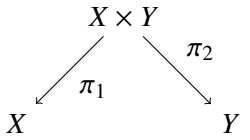


Figure 1.1: A product.

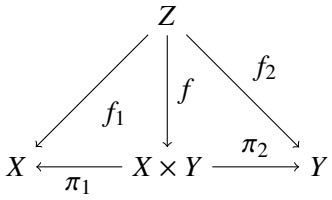


Figure 1.2: Commutativity of products.

- *For every object Z with morphisms $f_1 : Z \to X$ and $f_2 : Z \to Y$, there is a unique morphism $f : Z \to X \times Y$, such that the diagram in Figure 1.2 commutes.*

A diagram consists of nodes representing objects and directed edges representing morphisms between the nodes they connect. A diagram commutes if all directed paths between two objects are equal when composed. Figure 1.2, for example, commutes if $f_1 = \pi_1 \circ f$ and $f_2 = \pi_2 \circ f$.

In the category of sets, the products are Cartesian products of sets and π_i are the projection functions.

Given products $X \times Y$ and $A \times B$ with morphisms $f : X \to A$ and $g : Y \to B$, we will denote by $g \times f$ the morphism $X \times Y \to A \times B$, such that the diagram in Figure 1.3 commutes.

Given two morphisms $f : X \to A$ and $g : X \to B$, we define $\langle f, g \rangle$ such that the diagram in Figure 1.4 commutes.

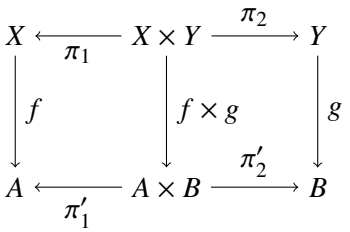


Figure 1.3: The diagram of $f \times g$

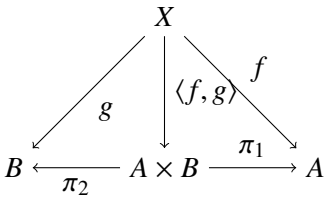


Figure 1.4: The diagram of $\langle f, g \rangle$

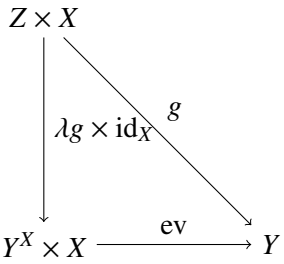


Figure 1.5: Commutativity of ev and λg.

Definition 1.15 *Let $\mathbb{C}$ be a category with products. For $X, Y \in ob(\mathbb{C})$, an exponential Y^X consists of an object Y^X together with a morphism $ev : Y^X \times X \to Y$ such that for any morphism $g : Z \times X \to Y$ there is a unique morphism $\lambda g : Z \to Y^X$ such that the diagram in Figure 1.5 commutes.*

In the example of the category of sets, an exponential Y^X is the set of functions from X to Y. The ev morphism is the function that sends the tuple (f, x) to $f(x)$.

Definition 1.16 *Given a function $g : (Z \times X) \to Y$, $\lambda g(z)(x) = g(z, x)$ is known as the* transpose *of g.*

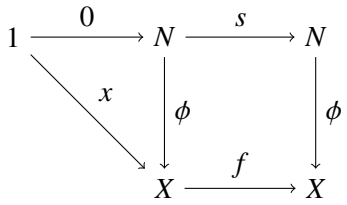
Definition 1.17 *An object T in a category $\mathbb{C}$ is* terminal*, if for every object X in $\mathbb{C}$, there exists exactly one morphism $X \to T$.*

In the category of sets, every singleton set is a terminal object, as there can only be a single function mapping a given set to a given singleton set.

Definition 1.18 *A* natural numbers object *in a category $\mathbb{C}$ with terminal object* 1 *is a triple $(0, N, s)$ where $N \in ob(\mathbb{C})$, $1 \xrightarrow{0} N$ and $N \xrightarrow{s} N$, such that for every diagram:*

$$1 \xrightarrow{x} X \xrightarrow{f} X$$

there is a unique map $\phi : N \to X$ such that the following diagram commutes:



In the category of sets, $\mathbb{N}$ is a natural numbers object, as we have morphisms $\mathbf{1} \mapsto 0$ and $x \mapsto x + 1$.

Definition 1.19 *Suppose $\mathbb{C}$ is a category of pre-orders and order-preserving maps, which has binary products. Then a* $\mathbb{C}$-enriched category *is a category $\mathbb{A}$ such that:*

- *For every two objects x, y in $\mathbb{A}$, the hom-set $\mathbb{A}(x, y)$ has the structure of object of $\mathbb{C}$.*
- *For every 3 objects x, y, z of $\mathbb{A}$, the composition map:*

$$\mathbb{A}(x, y) \times \mathbb{A}(z, y) \to \mathbb{A}(x, z)$$

is a morphism in $\mathbb{C}$.

From [Amadio and Curien, 1998]:

Definition 1.20 *A* Cartesian closed category *is a category $\mathbb{C}$ with:*

- *A terminal object* $\mathbf{1}$.
- *For any two objects X and Y there is a product $X \times Y$ in $\mathbb{C}$.*
- *For any two objects X and Y there is an exponential Y^X in $\mathbb{C}$.*

In [Lambek, 1980] Lambek showed that λ-calculus theories are equivalent to Cartesian closed categories. This is often called the *Curry-Howard-Lambek correspondence*: Curry and Howard showed λ-calculus and intuitionistic propositional logic are equivalent and Lambek added Cartesian closed categories.

Definition 1.21 *A Cartesian closed category $\mathbb{C}$ has* fixed-points *if for each object A there is a morphism $\mathbf{Y}^A : (A \Rightarrow A) \to A$ such that the diagram in Figure 1.6 commutes.*

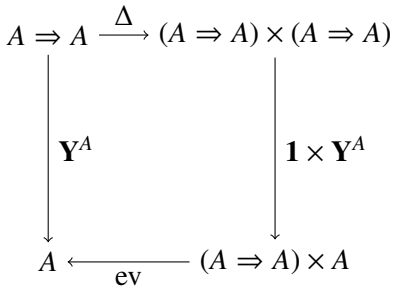


Figure 1.6: Commutativity fixed-points, Δ is the morphism that maps an object to a tuple of two copies of that object: $A \to A \times A$.

1.4 From λ-calculus theories to Cartesian closed categories

Given a λ-calculus theory, it can be turned into a Cartesian closed category as follows:

- The objects are tuples of types: $(\sigma_0, \ldots, \sigma_k)$.
- For every two objects, the morphisms $(\sigma_0, \ldots, \sigma_k) \to (\tau_0, \ldots, \tau_l)$ are equivalence classes (using $\equiv_{\beta\eta\alpha}$ elementwise) of l-tuples of λ-terms where the i-th term has type $\sigma_0 \to \cdots \to \sigma_k \to \tau_i$.

To verify that this is a category:

- Identity morphisms: for every tuple of types $(\sigma_0, \ldots, \sigma_k)$, we have a tuple of terms:

$$\begin{aligned}&(\lambda(x_0 : \sigma_0) \ldots \lambda(x_k : \sigma_k).x_0,\\ &\ldots,\\ &\lambda(x_0 : \sigma_0) \ldots \lambda(x_k : \sigma_k).x_k)\end{aligned}$$

  which is an identity arrow.

- Arrow composition: given morphisms $g : (\tau_0, \ldots, \tau_l) \to (\rho_0, \ldots, \rho_m)$ and $f : (\sigma_0, \ldots, \sigma_k) \to (\tau_0, \ldots, \tau_l)$, we define $g \circ f$ as:

$$\begin{aligned}&(\lambda(x_0 : \sigma_0).\lambda(x_1 : \sigma_1). \ldots .g_0\ (f_0\ x_0\ \ldots\ x_k)\ \ldots (f_l\ x_0\ \ldots\ x_k),\\ &\ldots,\\ &\lambda(x_0 : \sigma_0).\lambda(x_1 : \sigma_1). \ldots .g_m\ (f_0\ x_0\ \ldots\ x_k)\ \ldots (f_l\ x_0\ \ldots\ x_k))\end{aligned}$$

To verify that this is associative, given $h : (\rho_0, \ldots, \rho_m) \to (\kappa_0, \ldots, \kappa_n)$, $g : (\tau_0, \ldots, \tau_l) \to (\rho_0, \ldots, \rho_m)$ and $f : (\sigma_0, \ldots, \sigma_k) \to (\tau_0, \ldots, \tau_l)$:

(We will use $\overline{x}$ to denote $x_0\ x_1\ \ldots\ x_n$, where n is clear from the context.)

$$\begin{aligned}
h \circ (g \circ f) &= h \circ (\lambda(\overline{x} : \overline{\sigma}).g_0\ \overline{(f\ \overline{x})}, \ldots, \lambda(\overline{x} : \overline{\sigma}).g_m\ \overline{(f\ \overline{x})}) \\
&= (\lambda(\overline{x} : \overline{\sigma}).h_0\ \overline{(g\ \overline{(f\ \overline{x})})}, \ldots, \lambda(\overline{x} : \overline{\sigma}).h_n\ \overline{(g\ \overline{(f\ \overline{x})})}) \\
(h \circ g) \circ f &= (\lambda(\overline{x} : \overline{\sigma}).(h \circ g)_0\ \overline{(f\ \overline{x})}, \ldots, \lambda(\overline{x} : \overline{\sigma}).(h \circ g)_n\ \overline{(f\ \overline{x})}) \\
&\twoheadrightarrow_\beta (\lambda(\overline{x} : \overline{\sigma}).h_0\ \overline{(g\ \overline{(f\ \overline{x})})}, \ldots, \lambda(\overline{x} : \overline{\sigma}).h_n\ \overline{(g\ \overline{(f\ \overline{x})})})
\end{aligned}$$

So $h \circ (g \circ f) \equiv_{\beta\eta\alpha} (h \circ g) \circ f$.

To verify that this category is Cartesian closed:

- The terminal object is the zero-tuple, ().
- The product $(\sigma_0, \ldots, \sigma_k) \times (\tau_0, \ldots, \tau_l)$ is $(\sigma_0, \ldots, \sigma_k, \tau_0, \ldots, \tau_l)$. To verify that this is a product:
  - Given a product $(\sigma_0, \ldots, \sigma_k, \tau_0, \ldots, \tau_l)$, we can construct π_1 as a k-tuple of functions that project out only the σ-values $\sigma_0 \to \ldots \sigma_k \to \tau_0 \to \cdots \to \tau_l \to \sigma_i$. π_2 works the same way.
  - Given an object $(\rho_0, \ldots, \rho_m)$ with morphisms:

    $$\begin{aligned} f_1 &: (\rho_0, \ldots, \rho_m) \to (\sigma_0, \ldots, \sigma_k) \\ f_2 &: (\rho_0, \ldots, \rho_m) \to (\tau_0, \ldots, \tau_l) \end{aligned}$$

    we can construct a morphism:

    $$f : (\rho_0, \ldots, \rho_m) \to (\sigma_0, \ldots, \sigma_k, \tau_0, \ldots, \tau_l)$$

    as the concatenation of the terms in f_1 and f_2. It is easy to see that this makes the diagram in Figure 1.2 commute.
- The exponential Y^X of $X = (\tau_0, \ldots, \tau_l)$ and $Y = (\sigma_0, \ldots, \sigma_k)$ is $(\tau_0 \to \cdots \to \tau_l \to \sigma_0, \ldots, \tau_0 \to \cdots \to \tau_l \to \sigma_k)$.

  To verify that this satisfies the requirements of exponentials, let $Z = (\rho_0, \ldots, \rho_m)$ and let g be a morphism from $(\rho_0, \ldots, \rho_m, \tau_0, \ldots, \tau_l)$ to $(\sigma_0, \ldots, \sigma_k)$. Then g is a k tuple of terms with type $\rho_0 \to \ldots \rho_m \to \tau_0 \to \cdots \to \tau_l \to \sigma_i$. Because of the associativity of $\to$, we can also see this as a tuple of terms with type $\rho_0 \to \ldots \rho_m \to (\tau_0 \to \cdots \to \tau_l \to \sigma_i)$, which is a morphism from $Z \to Y^X$.

  We can define the $\text{ev} : Y^X \times X \to Y$ morphism as the k-tuple $\lambda f_0 : \tau_0 \to \cdots \to \tau_l \to \sigma_0 \ldots . \lambda f_k : \tau_0 \to \cdots \to \tau_l \to \sigma_k . \lambda x_0 : \tau_0 \ldots . \lambda x_l : \tau_l . f_i\ x_0\ \ldots\ x_l$.

1.5 Domains

Definition 1.22 *A* partial order *is a binary relation $\sqsubseteq$ that is:*

- Reflexive*: $\forall a.a \sqsubseteq a$.*
- Antisymmetric*: If $a \sqsubseteq b$ and $b \sqsubseteq a$, then $a = b$.*
- Transitive*: If $a \sqsubseteq b$ and $b \sqsubseteq c$, then $a \sqsubseteq c$.*

Definition 1.23 *A* poset *is a set together with a partial order.*

For example, $(\mathbb{R}, \leq)$ is a poset. For any set A, $(\mathcal{P}(A), \subseteq)$ is also a poset.

Definition 1.24 *Let $(P, \sqsubseteq)$ be a poset and $X \subseteq P$, then the* least upper bound *(or: supremum) of X (if it exists) is the smallest element $s \in P$ such that for all $x \in X$, $x \sqsubseteq s$. It is denoted $\bigsqcup X$. If $X = \{a, b\}$, then we write $a \sqcup b$.*

Definition 1.25 *Let $(P, \sqsubseteq)$ be a poset and $X \subseteq P$, then the* greatest lower bound *(or: infimum) of X (if it exists) is the greatest element s of P such that for all $x \in X$, $s \sqsubseteq x$. It is denoted $\bigsqcap X$. If $X = \{a, b\}$, then we write $a \sqcap b$.*

The supremum and infimum do not necessarily exist, but if they exist, then they are unique.

Definition 1.26 *A subset X of a poset $(P, \sqsubseteq)$ is* directed *if it is non-empty and every finite subset of that set has an upper bound in X: $\forall a, b \in X \exists c \in X. a \sqsubseteq c \wedge b \sqsubseteq c$.*

In $(\mathbb{R}, \leq)$ every non-empty set is directed. In $(\mathcal{P}(\mathbb{N}), \subseteq)$, the set $\{\{1\}, \{2\}\}$ is not directed, but $\{\{1\}, \{2\}, \{1, 2, 3\}\}$ is.

Definition 1.27 *A poset is* directed complete *if all directed subsets have a supremum.*

Definition 1.28 *A poset is* bounded complete *if all subsets that have an upper bound have a supremum.*

Definition 1.29 *We will use $x \uparrow y$ to denote that x and y are bounded.*

Definition 1.30 *If a poset has a unique smallest element, then it is called the* bottom *element. It is denoted $\bot$.*

For any set A, the poset $(\mathcal{P}(\mathbb{A}), \subseteq)$ has $\bot = \emptyset$.

Theorem 1.31 *A non-empty bounded complete poset has a $\bot$.*

Proof Any $x \in P$ is an upper bound for $\emptyset$, so, by the bounded completeness, it must have a least upper bound, y. As $y \sqsubseteq x$ for all $x \in P$, it must be the $\bot$ element of P. $\square$

Theorem 1.32 *In a bounded complete poset $(P, \sqsubseteq)$, $a \sqcap b$ exists for all $a, b \in P$.*

Proof The supremum of $a \sqcap b$ is given by:

$$a \sqcap b = \bigsqcup \{c \mid c \sqsubseteq a \wedge c \sqsubseteq b\}$$

This supremum exists, at the set is bounded (as a and b are both a bound) and P is bounded complete.

Definition 1.33 *An element c of a poset $(P, \sqsubseteq)$ is* compact *if for every directed subset X, if $\bigsqcup X$ exists and $c \sqsubseteq \bigsqcup X$, then $c \sqsubseteq d$ for some element $d \in X$.*

In $(\mathcal{P}(\mathbb{N}), \subseteq)$, the compact elements are exactly the finite sets:

- Let C be a finite subset of $\mathbb{N}$ and X a directed subset of $\mathcal{P}(\mathbb{N})$ such that $C \subseteq \bigsqcup X$. By induction:
  - If $C = \emptyset$, then $C \subseteq D$ for any $D \in X$.
  - If $C = C' \cup \{n\}$ and by the induction hypothesis there is a $D' \in X$ such that $C' \subseteq D'$, then there must be a $D'' \in X$ such that $n \in D''$ (as $C \subseteq \bigsqcup X$). As X is directed, there must be a $D \in X$ such that $D' \subseteq D$ and $D'' \subseteq D$. So $C \subseteq D$.
- If C is an infinite subset of $\mathbb{N}$, then let:

$$X = \{\{0\}, \{0, 1\}, \{0, 1, 2\}, \{0, 1, 2, 3\}, \dots\}$$

  It is easy to see that $C \subseteq \bigsqcup X = \mathbb{N}$, however, there is no $D \in X$ such that $C \subseteq D$ because all elements of X are finite sets. □

Definition 1.34 *A poset $(P, \sqsubseteq)$ is* algebraic *if for every element $x \in P$:*

$$x = \bigsqcup \{d \mid d \in P, d \sqsubseteq x, d \text{ compact}\}$$

For example, $(\mathcal{P}(\mathbb{N}), \subseteq)$ is algebraic: recall that the compact elements are the finite sets.

Definition 1.35 *An element c of a poset $(P, \sqsubseteq)$ is* prime *if for every subset X, if $\bigsqcup X$ exists and $c \sqsubseteq \bigsqcup X$, then $c \sqsubseteq d$ for some element $d \in X$.*

A prime element is always compact, but the converse does not hold in general.

Definition 1.36 *A poset* $(P, \sqsubseteq)$ *is* prime algebraic *if for every element* $x \in P$*:*

$$x = \bigsqcup \{d \mid d \in P, d \sqsubseteq x, d \text{ prime}\}$$

The posets we will mainly use are Scott-domains:

Definition 1.37 *A* Scott-domain *is a directed complete, bounded complete, algebraic poset.*

With the following functions between Scott-domains, [Abramsky and Jung, 1994]:

Definition 1.38 *A* Scott-continuous function *is a function between two posets that is monotone and preserves the suprema of directed subsets.*

In other words:

$$f\left(\bigsqcup X\right) = \bigsqcup f(X)$$

Definition 1.39 Pointwise function ordering*: Two functions* $f : X \to Y$ *and* $g : X \to Y$*, where* $(Y, \sqsubseteq)$ *is a poset, are ordered* $f \sqsubseteq_p g$ *by the pointwise ordering if and only if* $\forall x. f(x) \sqsubseteq g(x)$*.*

Lemma 1.40 *Let A and B be Scott-domains. Then the set of Scott-continuous functions from A to B, denoted* $[A \to B]$*, under the pointwise function ordering is a Scott-domain.*

Proof

- Let C be a directed subset of $[A \to B]$. Then for all functions $f, g \in C$, there must be a $h \in C$ such that for all $x \in A$, $f(x) \sqsubseteq_p h(x)$ and $g(x) \sqsubseteq_p h(x)$. So the set $\{f(x) \mid f \in C\}$ is directed, this means this set must have a supremum, as it is a directed subset of A, and A is a Scott-domain.

  So we define h as $h(x) = \bigsqcup\{f(x) \mid f \in C\}$. To show that h is Scott-continuous, let X be a directed subset of A:

$$\begin{aligned}
h\left(\bigsqcup X\right) &= \bigsqcup \left\{ f\left(\bigsqcup X\right) \mid f \in C \right\} \\
&= \bigsqcup \left\{ \bigsqcup_{x \in X} f(x) \mid f \in C \right\} \\
&= \bigsqcup \bigsqcup_{x \in X} \{f(x) \mid f \in C\} \\
&= \bigsqcup_{x \in X} \bigsqcup \{f(x) \mid f \in C\} \\
&= \bigsqcup_{x \in X} h(x)
\end{aligned}$$

  so h is indeed Scott-continuous.

- To prove that $[A \to B]$ is bounded complete, we will look at it pointwise. Let C be a subset of $[A \to B]$. If C has an upper bound, then the set $\{f(x) \mid f \in C\}$ has an upper bound for each x, which means that set has a supremum, as it is a subset of A, and A is a Scott-domain. So we define the function h as $h(x) = \bigsqcup\{f(x) \mid f \in C\}$. h is Scott-continuous, by the same argument as the previous case.

- First, we must find the compact Scott-continuous functions from $[A \to B]$.

  Define $f_a^c : A \to B$ as:

$$f_a^c(x) = \begin{cases} c & a \sqsubseteq x \\ \bot & \text{otherwise} \end{cases}$$

  for all c and compact a, f_a^c is Scott-continuous: if $a \sqsubseteq \bigsqcup X$, then $f(\bigsqcup X) = c$. As a is compact, there must be an $x \in X$ such that $a \sqsubseteq x$, so $\bigsqcup f(X) = c$. If $f_a^c(X) = \bot$, then $a \not\sqsubseteq \bigsqcup X$, so $\forall x \in X. a \not\sqsubseteq x$, so $f_a^c(X) = \{\bot\}$, so $\bigsqcup f(X) = \bot$.

  If c is compact, then f_a^c is compact: suppose $f_a^c \sqsubseteq \bigsqcup X$ for a directed set X. Then $f_a^c(a) \sqsubseteq \bigsqcup\{g(a) \mid g \in X\}$, because the order is pointwise. This means $c \sqsubseteq \bigsqcup\{g(a) \mid g \in X\}$. So for some $g \in X$, $c \sqsubseteq g(a)$. This means $f_a^c \sqsubseteq g$ (by the monotonicity of Scott-continuous functions).

  If f_a^c is compact, then c is compact: if $c \sqsubseteq \bigsqcup_i c_i$, then $f_a^c \sqsubseteq \bigsqcup f_a^{c_i}$, which means there must be an $f_a^{c_i}$ such that $f_a^c \sqsubseteq f_a^{c_i}$, which means $c \sqsubseteq c_i$.

  If $f : A \to B$ is Scott-continuous, then $f_a^c \sqsubseteq f$ for all compact a and compact c such that $c \sqsubseteq f(a)$.

  So:

$$f = \bigsqcup \{f_a^c \mid a \in A \text{ compact}, c \in B \text{ compact}, c \sqsubseteq f(a)\} \tag{1.2}$$

  Let $X = \{f_a^c \mid a \in A \text{ compact}, c \in B \text{ compact}, c \sqsubseteq f(a)\}$, then:

$$f = \bigsqcup_{X' \subseteq X,\ X' \text{ finite}} \bigsqcup X'$$

  So if f is compact, then f is equal to a the supremum of a finite set of $f_{a_n}^{c_n}$-functions.

  Now we show that $[A \to B]$ is algebraic. Let $f \in [A \to B]$, then g is a compact function below f if and only if g is a finite subset of:

$$\{f_a^c \mid a \in A \text{ compact}, c \in B \text{ compact}, c \sqsubseteq f(a)\}$$

As f_a^c are all compact, f must be equal to the set of compact elements below it.

□

Lemma 1.41 *The Cartesian product $A \times B$ of two Scott-domains A and B is again a Scott-domain.*

Proof First note that X is a directed subset of $A \times B$ if and only if $\pi_A(X)$ is a directed subset of A and $\pi_B(X)$ a directed subset of B: if X is non-empty, then $\pi_A(X)$ and $\pi_B(X)$ are non-empty. If a finite subset $Y \subseteq X$ has an upper bound in X, say z, then $\pi_A(z)$ is an upper bound for $\pi_A(Y)$ and vice versa for B.

Now to show that $A \times B$ is a Scott-domain:

- $A \times B$ is directed complete: let X be a directed subset of $A \times B$ and let $A' = \pi_A(X)$ and $B' = \pi(X)$. Thus, A' is a directed subset of A and B' a directed subset of B. As A and B are Scott-domains, they are directed complete, therefore A' has a supremum and B' has a supremum. The supremum of X is $(\bigsqcup A', \bigsqcup B')$.
- $A \times B$ is bounded complete: suppose X has an upper bound, (a, b). Then a is an upper bound of $\pi_A(X)$, so $\pi_A(X)$ has a supremum a' (as A is bounded complete) and b an upper bound for $\pi_B(X)$, so let b' be the supremum of $\pi_B(X)$. This means (a', b') is the supremum of X.
- $A \times B$ is algebraic: first note that the compact elements of $A \times B$ are:

  $$\{(a, b) \mid a \in A, a \text{ compact}, b \in B, b \text{ compact}\}$$

  As A and B are algebraic, for all elements $(a, b) \in A \times B$, $a = \bigsqcup\{d \mid d \in A, d \sqsubseteq a, d \text{ compact}\}$ and $b = \bigsqcup\{d \mid d \in A, d \sqsubseteq b, d \text{ compact}\}$, so $(a, b) = \bigsqcup\{(d, e) \mid d \in A, d \sqsubseteq a, d \text{ compact}, e \in B, e \sqsubseteq b, e \text{ compact}\}$.

□

Theorem 1.42 *The category of Scott-domains with Scott-continuous functions as morphisms is a Cartesian closed category.*

Proof First, we must show that this is indeed a category.

- For each object A, the identity function $A \to A$ is Scott-continuous, as it is monotone and preserves the suprema of directed subsets.

- The composition of two Scott-continuous functions is again Scott-continuous: the composition of two monotone functions is monotone, and:

$$f\left(g\left(\bigsqcup X\right)\right) = f\left(\bigsqcup g\left(X\right)\right) = \bigsqcup f\left(g\left(X\right)\right)$$

Now to show that this category is Cartesian closed:

- There is a terminal object: $\{\bot\}$. For every Scott-domain, there is a unique function that maps every element to $\bot$.
- Products are Cartesian products: $A \times B = \{(a, b) \mid a \in A, b \in B\}$. The partial order is given by $(a, b) \sqsubseteq (c, d) \Leftrightarrow a \sqsubseteq c \wedge b \sqsubseteq d$. This is a Scott-domain by lemma 1.41 and it is a valid (categorical) product, as projection morphisms are Scott-continuous.
- Exponentials B^A are the Scott-domain $[A \to B]$ under the pointwise function ordering. By lemma 1.41, if A and B are Scott-domains, then the set of all Scott-continuous functions $[A \to B]$ is itself a Scott-domain.

  We can simply define the ev morphism as:

$$\mathrm{ev}(f, x) = f(x)$$

  which is Scott-continuous and satisfies the requirements for exponentials.

Definition 1.43 *An element d of a poset $(P, \sqsubseteq)$ is* very finite *if the set $\{x \in P \mid x \sqsubseteq d\}$ is finite.*

The following comes from [Berry, 1978] and [van Oosten, 1997]:

Definition 1.44 *A* dI-domain *is an directed complete, bounded complete, algebraic poset $(P, \sqsubseteq)$ such that:*

Axiom d. *For every $x, y, z \in P$, if $y \uparrow z$, then $x \sqcap (y \sqcup z) = (x \sqcup y) \sqcap (x \sqcup z)$.*

Axiom I. *Every compact element is very finite.*

Definition 1.45 *A function $f : D \to E$, where D and E are dI-domains, is* stable *if it is Scott-continuous and satisfies:*

$$\forall x, y \in D. x \uparrow y \Rightarrow f(x \sqcap y) = f(x) \sqcap f(y)$$

Lemma 1.46 *The composition of two stable functions is again stable.*

Proof Let $f : B \to C$ and $g : A \to B$ be stable functions.

$$\forall x, y.x \uparrow y \Rightarrow f(g(x \sqcap y)) = f(g(x) \sqcap g(y)) = f(g(x)) \sqcap f(g(y))$$

□

Definition 1.47 *Two stable functions $f, g : D \to E$, where D and E are dI-domains, are ordered by the* stable function ordering $f \sqsubseteq_s g$ *if:*

- $f \sqsubseteq_p g$ *by the pointwise function ordering:* $\forall x \in D.f(x) \sqsubseteq_E g(x)$.
- $\forall x, y \in D.x \sqsubseteq_D y \Rightarrow f(x) = f(y) \sqcap g(x)$.

It is easy to see that for stable functions f, g with $f \sqsubseteq_p g$ and for all $x, y \in D$ with $x \sqsubseteq_D$, $f(x)$ is a lower bound for $f(y)$ and $g(x)$: by the monotonicity of f and the pointwise function ordering respectively. f is stable if this lower bound is the greatest lower bound.

The following lemma is from [Zhang, 1991]:

Lemma 1.48 *Suppose f and g are stable functions with $f \uparrow g$ under the stable function ordering and $x \uparrow y$. Then:*

$$f(x) \sqcap g(y) \ = f(x \sqcap y) \sqcap g(x \sqcap y) = f(y) \sqcap g(x)$$

Proof Because $f \uparrow g$, there must be a stable function h such that $f \sqsubseteq_s h$ and $g \sqsubseteq_s h$. Because $x \sqcap y \sqsubseteq x$ and $x \sqcap y \sqsubseteq y$, we can apply the definition of the stable function ordering to get:

$$\begin{aligned} f(x \sqcap y) &= f(x) \sqcap h(x \sqcap y) \\ g(x \sqcap y) &= g(y) \sqcap h(x \sqcap y) \end{aligned}$$

So, because $f \sqsubseteq_p h$ and $g \sqsubseteq_p h$:

$$\begin{aligned} f(x \sqcap y) \sqcap g(x \sqcap y) &= f(x) \sqcap g(y) \sqcap h(x \sqcap y) \\ &= f(x) \sqcap g(y) \sqcap h(x) \sqcap h(y) \\ &= f(x) \sqcap g(y) \end{aligned}$$

□

Definition 1.49 *Let $f : A \to B$ be a stable function between dI-domains A and B. Let:*

$$\mu f = \{(a, p) \mid a \in A, a \text{ compact}, p \in B, p \text{ prime}, f(a) \sqsupseteq p \wedge (\forall a' \sqsupseteq a.f(a') \sqsupseteq p \Rightarrow a = a')\}$$

We will state the following lemmas and theorems from [Zhang, 1991] without proof:

Lemma 1.50 *Let $f, g : A \to B$ be stable functions, then $\mu f \subseteq \mu g$ if $f \sqsubseteq_s g$.*

Theorem 1.51 *Suppose D is a Scott-domain satisfying axiom I, then D is prime algebraic if and only if it is a dI-domain.*

Lemma 1.52 *Let $f : A \to B$ be a stable function. Then for all $x \in A$:*

$$f(x) = \bigsqcup \{p \mid \exists a \sqsubseteq x.(a, p) \in \mu f\}$$

Theorem 1.53 *Let $f, g \in [A \to_s B]$. $f \sqsubseteq_s g$ if and only if $\mu f \subseteq \mu g$. Let $\{(a_i, p_i) \mid i \in I, a_i \in A, a \text{ compact}, p_i \in B, p_i \text{ prime}\}$. Then $\{(a_i, p_i) \mid i \in I\} = \mu f$ if and only if:*

- $\forall J \subset I.J$ *finite* $\wedge \{a_i \mid i \in J\}$ *directed* $\Rightarrow \{p_i \mid i \in J\}$ *directed.*
- $(a_i \uparrow a_j \wedge p_i = p_j) \Rightarrow a_i = a_j$.
- $\forall p \in B.p$ *prime* $\Rightarrow \left(p_i \sqsubseteq p \Rightarrow \exists j.p_j = p \wedge a_i \sqsupseteq a_j\right)$.

Theorem 1.54 *Given two dI-domains A and B, the poset of stable functions from A to B using the stable function ordering, denoted $[A \to_s B]$, is a dI-domain.*

Proof This proof is based on the proof-sketch from [Zhang, 1991].

- Bounded complete: Let C be a bounded subset of $[A \to_s B]$ and consider $F(x) = \bigsqcup_{f \in C} f(x)$. First, we show that F itself is a stable function. Let $x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \ldots$ be a chain in A. Then:

$$\begin{aligned} \bigsqcup_{f \in C} f\left(\bigsqcup_{i \in \omega} x_i\right) &= \bigsqcup_{f \in C}\left(\bigsqcup_{i \in \omega} f(x_i)\right) \\ &= \bigsqcup_{i \in \omega}\left(\bigsqcup_{f \in C} f(x_i)\right) \end{aligned}$$

  so F is Scott-continuous.

  Let $x, y \in A$ with $x \uparrow y$. Then, using lemma 1.48 and axiom d of A:

$$\begin{aligned}
\bigsqcup_{f\in C} f(x \sqcap y) &\sqsubseteq \left(\bigsqcup_{f\in C} f(x)\right) \sqcap \left(\bigsqcup_{g\in C} g(y)\right) \\
&= \bigsqcup_{f\in C}\bigsqcup_{g\in C} (f(x) \sqcap g(y)) \\
&= \bigsqcup_{f\in C}\bigsqcup_{g\in C} (f(x \sqcap y) \sqcap g(x \sqcap y)) \\
&\sqsubseteq \bigsqcup_{f\in C} f(x \sqcap y)
\end{aligned}$$

which means $\bigsqcup_{f\in C} f(x \sqcap y) = \left(\bigsqcup_{f\in C} f(x)\right) \sqcap \left(\bigsqcup_{g\in C} g(y)\right)$, so F is stable.

Let $x, y \in A$, $x \sqsubseteq y$ and $g \in C$. Then:

$$\begin{aligned}
g(x) &\sqsubseteq g(y) \sqcap \bigsqcup_{f\in C} f(x) \\
&= \bigsqcup_{f\in C} (g(y) \sqcap f(x)) \\
&= \bigsqcup_{f\in C} (g(x \sqcap y) \sqcap f(x \sqcap y)) \\
&\sqsubseteq g(x)
\end{aligned}$$

so $g(x) = g(y) \sqcap \bigsqcup_{f\in C} f(x)$, which implies $g \sqsubseteq_s F$, which means $[A \to_s B]$ is bounded complete.

- Directed complete: Let C be a directed subset of $[A \to_s B]$. Then for all functions $f, g \in C$, there must be a $h \in C$ such that for all $x \in A$, $f(x) \sqsubseteq_s h(x)$ and $g(x) \sqsubseteq_s h(x)$. So the set $\{f(x) \mid f \in C\}$ is directed, this means this set must have a supremum, as it is a directed subset of A, and A is a dI-domain.

  So we define h as $h(x) = \bigsqcup\{f(x) \mid f \in C\}$. h is Scott-continuous by the same argument as in the proof of lemma 1.40. To show that h is stable:

$$
\begin{aligned}
h(x \sqcap y) &= \bigsqcup \{f(x \sqcap y) \mid f \in C\} \\
&\sqsubseteq \left(\bigsqcup_{f \in C} f(x)\right) \sqcap \left(\bigsqcup_{g \in C} g(y)\right) \\
&= \bigsqcup_{f \in C} \bigsqcup_{g \in C} (f(x) \sqcap g(y)) \\
&= \bigsqcup_{f \in C} \bigsqcup_{g \in C} (f(x \sqcap y) \sqcap g(x \sqcap y)) \\
&\sqsubseteq \bigsqcup_{f \in C} f(x \sqcap y)
\end{aligned}
$$

so h is stable and it is the supremum of C.

- Prime algebraic: Instead of proving axiom d and algebraicity separately, we prove that $[A \to_s B]$ is prime algebraic and apply theorem 1.51.

  First, we characterize the compact elements of $[A \to_s B]$. We do this similarly to the proof of lemma 1.40. We again introduce the functions:

$$
f_a^c(x) = \begin{cases} c & a \sqsubseteq x \\ \bot & \text{otherwise} \end{cases}
$$

  where $a, c \in A$ and a compact.

  First, we show that f_a^c is stable for all a and c, the proof of lemma 1.40 already showed it is Scott-continuous. Let $x, y \in A$ with $x \uparrow y$. We can distinguish the following cases:

  - If $f_a^c(x) = c$ and $f_a^c(y) = c$.
    Then $a \sqsubseteq x$ and $a \sqsubseteq y$, which implies $a \sqsubseteq x \sqcap y$, so $f_a^c(x \sqcap y) = c = f_a^c(x) \sqcap f_a^c(y)$.
  - If $f_a^c(x) = c$ and $f_a^c(y) = \bot$.
    Then $a \not\sqsubseteq y$, so $a \not\sqsubseteq x \sqcap y$, so $f_a^c(x \sqcap y) = \bot = f_a^c(x) \sqcap f_a^c(y)$.
    The case $f_a^c(x) = \bot$ and $f_a^c(y) = c$ is the same.
  - If $f_a^c(x) = \bot$ and $f_a^c(y) = \bot$.
    Then $a \not\sqsubseteq x$ and $a \not\sqsubseteq y$, which implies $a \not\sqsubseteq x \sqcap y$, so $f_a^c(x \sqcap y) = \bot = f_a^c(x) \sqcap f_a^c(y)$.

  If f_a^c is prime, then c is prime: if $c \sqsubseteq \bigsqcup_i c_i$ for a set $\{c_i \mid i \in I\}$, then $f_a^c \sqsubseteq_s \bigsqcup_i f_a^{c_i}$ for all compact a, which means there must be an i such that $f_a^c \sqsubseteq_s f_a^{c_i}$, which means $c = f_a^c(a) \sqsubseteq f_a^{c_i}(a) = c_i$.

If $f : A \to B$ is stable, then $f_a^c \sqsubseteq_s f$ for all compact a and prime c such that $c \sqsubseteq f(a)$. So:

$$f = \bigsqcup \{f_a^c \mid a \in A \text{ compact}, c \in B \text{ prime}, c \sqsubseteq f(a)\}$$

Let $X = \{f_a^c \mid a \in A \text{ compact}, c \in B \text{ prime}, c \sqsubseteq f(a)\}$, then:

$$f = \bigsqcup_{X' \subseteq X,\ X' \text{ finite}} \bigsqcup X'$$

So if f is prime, then f is equal to a finite set of f_a^c-functions.

Now to show that $[A \to_s B]$ is prime algebraic. Suppose $f \in [A \to_s B]$, then g is a compact function below f if and only if g is a finite subset of:

$$\{f_a^c \mid a \in A \text{ compact}, c \in B \text{ prime}, c \sqsubseteq f(a)\}$$

As all sets are equal to the union of all finite subsets, f must be equal to $\bigsqcup \{g \mid g \in [A \to_s B] \text{ prime}, g \sqsubseteq_s f\}$, so $[A \to_s B]$ is prime algebraic.

- Axiom I: The compact elements of $[A \to_s B]$ are suprema of finite sets of f_a^c with a and c compact. For a set $X = \{f_a^c \mid a \in A \text{ compact}, c \in B \text{ compact}, c \sqsubseteq f(a)\}$, there are only finitely many subsets of X. As A and B are dI-domains, there can only be finitely many $f_{a'}^{c'}$ below a given f_a^c. So $[A \to_s B]$ satisfies axiom I.

□

Theorem 1.55 *The category of dI-domains with stable functions is Cartesian closed.*

Proof To show that it is a valid category:

- For each dI-domain A, it is trivial to see that the identity function $A \to A$ is stable: $\forall x, y \in A. x \uparrow y \Rightarrow f(x \sqcap y) = x \sqcap y = f(x) \sqcap f(y)$.
- See lemma 1.46.

Now to show that this category is Cartesian closed:

- There is a terminal object, namely the dI-domain with a single object: $\{\bot\}$. For every dI-domain, there is a unique function that maps every element to $\bot$.

- If E and D are dI-domains, then $E \times D$ is also a dI-domain. $E \times D$ is directed complete, bounded-complete and algebraic, as all dI-domains are Scott-domains and Scott-domains are Cartesian closed. All that's needed is to show it also satisfies axioms d and I.

  d. Note that $(y, y') \uparrow (z, z')$ if and only if $y \uparrow z$ and $y' \uparrow z'$. So for every (x, x'), (y, y') and $(z, z') \in E \times D$, if $(y, y') \uparrow (z, z')$ then $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ and $x' \sqcap (y' \sqcup z') = (x' \sqcap y') \sqcup (x' \sqcap z')$, so:

$$(x, x') \sqcap ((y, y') \sqcup (z, z')) = ((x, x') \sqcap (y, y')) \sqcup ((x, x') \sqcap (z, z'))$$

  I. Note that (a, b) is compact if and only if a is compact and b is compact. So for a compact (a, b): $\{x \in E \times D \mid x \sqsubseteq (a, b)\} = \{(x, y) \in E \times D \mid x \sqsubseteq a \wedge y \sqsubseteq b\} = \{x \in E \mid x \sqsubseteq a\} \times \{y \in D \mid y \sqsubseteq b\}$, which is finite because it is the product of two finite sets.

- If E and D are dI-domains, then the exponential D^E is the dI-domain of stable functions from E to D. This is a dI-domain using theorem 1.54.

Definition 1.56 *An* ω-complete poset *is a poset* $(P, \sqsubseteq)$ *where each countable chain* $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ *has a supremum.*

Every directed complete poset is an ω-complete poset, but not conversely.

1.6 Models of simply typed λ-calculus

A *model* of a simply typed λ-calculus theory interprets the types and terms in a Cartesian closed category $\mathbb{C}$. It uses a function $[\![.]\!]$ that interprets types and terms as objects respectively morphisms of the category.

We must give an interpretation of each ground type of our theory: $[\![.]\!] : \sigma \to \text{obj}(\mathbb{C})$. Given interpretations of all base types, the interpretation of the other types follows from the exponentials the chosen category: $[\![\tau_1 \to \tau_2]\!] = [\![\tau_2]\!]^{[\![\tau_1]\!]}$.

We also need an interpretation of all constants. For a constant $c : \tau$, we need to distinguish a morphism $\mathbf{1} \xrightarrow{[\![c]\!]} [\![\tau]\!]$.

For example, given a λ-calculus theory $\mathcal{T}$ with base types ι and o and constants $z : \iota$, $s : \iota \to \iota$, $\text{tt} : o$ and $\text{ff} : o$ we can create a Cartesian closed category as described in Section 1.4. We start with the objects $\{\mathbb{N}, \mathbb{B}\}$ ($\mathbb{B} = \{\text{true}, \text{false}\}$) and define the interpretation as:

$$
\begin{aligned}
[\![\iota]\!] &= \mathbb{N} \\
[\![o]\!] &= \mathbb{B} \\
[\![z]\!] &= \mathbf{1} \xrightarrow{0} \mathbb{N} \\
[\![s]\!] &= \mathbf{1} \xrightarrow{s} \mathbb{N}^{\mathbb{N}} \\
[\![\mathrm{tt}]\!] &= \mathbf{1} \xrightarrow{\mathrm{true}} \mathbb{B} \\
[\![\mathrm{ff}]\!] &= \mathbf{1} \xrightarrow{\mathrm{false}} \mathbb{B}
\end{aligned}
$$

As a theory this example is quite limited, as it has no recursion or conditionals.

The interpretation of sets of free variables is a product of their types: $[\![\{x_1 : \tau_1, x_2 : \tau_2, \dots\}]\!] = [\![\tau_1]\!] \times [\![\tau_2]\!] \times \dots$. Using that, we interpret (non-constant) terms as morphisms between the interpretation of their free variables and their type. So given $t : \tau$, the interpretation $[\![M]\!]$ is a morphism $[\![\mathrm{fv}(M)]\!] \xrightarrow{[\![M]\!]} [\![\tau]\!]$, given by:

- For a variable $x : \tau$, the identity arrow for $[\![\tau]\!]$:

$$[\![\mathrm{fv}(x)]\!] = [\![\tau]\!] \xrightarrow{[\![x]\!]} [\![\tau]\!]$$

- For terms $[\![M]\!] : [\![\mathrm{fv}(M)]\!] \to [\![\tau_2]\!]^{[\![\tau_1]\!]}$ and $[\![N]\!] : [\![\tau_1]\!]$, $[\![M\ N]\!]$ is given by the composite:

$$[\![\mathrm{fv}(M\ N)]\!] \xrightarrow{\langle [\![M]\!] \circ \pi_M, [\![N]\!] \circ \pi_N \rangle} [\![\tau_2]\!]^{[\![\tau_1]\!]} \times [\![\tau_1]\!] \xrightarrow{\mathrm{ev}} [\![\tau_2]\!]$$

  where π_M and π_N are the projection morphisms such that $\pi_M : [\![\mathrm{fv}(M\ N)]\!] \to [\![\mathrm{fv}(M)]\!]$ and $\pi_N : [\![\mathrm{fv}(M\ N)]\!] \to [\![\mathrm{fv}(N)]\!]$.

- For a term $[\![M]\!] : [\![\mathrm{fv}(M)]\!] \to \tau_2$ and a variable $x : \tau_1$, we define $[\![\lambda x.M]\!]$ as the transpose of (see definition 1.16):

  - If x does not occur free in M:

$$[\![\mathrm{fv}(M)]\!] \times [\![\{\tau_1\}]\!] \xrightarrow{[\![M]\!] \circ \pi_M} [\![\tau_2]\!]$$

    where π_M is the projection function such that:

$$\pi_M : [\![\mathrm{fv}(M)]\!] \times [\![\{\tau_1\}]\!] \to [\![\mathrm{fv}(M)]\!]$$

– If x occurs free in M:

$$[\![\mathrm{fv}(M)]\!] \times [\![\{\tau_1\}]\!] \xrightarrow{[\![M]\!]} [\![\tau_2]\!]$$

The equality judgment $\Gamma \vdash N = M : \tau$ holds if and only if the diagram in Figure 1.7 commutes.

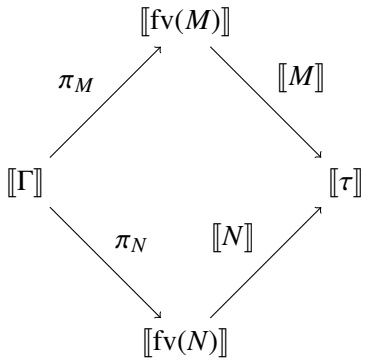


Figure 1.7: The equality judgment $\Gamma \vdash M = N : \tau$. π_M and π_N are projection functions.

To show that this model is a λ-calculus theory, we have to show that it satisfies the requirements from definition 1.11.

1. If $\Gamma \vdash M = N : \tau$, then the diagram in Figure 1.7 commutes. For Δ such that $\Gamma \subseteq \Delta$, we can find a projection morphism $[\![\Delta]\!] \xrightarrow{\pi_\Delta} [\![\Gamma]\!]$, so by composing π_Δ with π_M and π_N we get that $\Delta \vdash M = N : \tau$.

2. $\mathrm{fv}(M) \vdash M = M : \tau$ holds for every term M by the definition of $[\![M]\!]$.

3. If $\Gamma \vdash M = N : \tau$ and $\Gamma \vdash N = O : \tau$ then the diagram in Figure 1.9 commutes, so $\Gamma \vdash M = O : \tau$.

4. Suppose $M : \tau$ has free variables $\mathrm{fv}(M) = \{x_1, \ldots, x_n\}$ with $x_1 : \tau_1, \ldots, x_n : \tau_n$. Let $N_1 : \tau_1, \ldots, N_n : \tau_n$.

   - Suppose $M = x_1$. Then $\tau = \tau_1$, $\mathrm{fv}(M[x := N]) = \mathrm{fv}(N)$ and $[\![M]\!] = \mathrm{id}_{[\![\tau]\!]}$. So:

     $$[\![\mathrm{fv}(M[x := N])]\!] \xrightarrow{[\![N]\!]} [\![\tau_1]\!] \xrightarrow{[\![M]\!]} [\![\tau]\!]$$

     is equal to:

     $$[\![\mathrm{fv}(M[x := N])]\!] \xrightarrow{[\![N]\!]} [\![\tau]\!]$$

which is equal to:

$$[\![\mathrm{fv}(M[x \coloneqq N])]\!] \xrightarrow{[\![M[x:=N]]\!]} [\![\tau]\!]$$

- Suppose $M = (O\ P)$ with $O : \tau' \to \tau$ and $P : \tau'$, and the induction hypothesis:

$$[\![\mathrm{fv}(O[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{\tilde{N}} [\![\mathrm{fv}(O)]\!] \xrightarrow{[\![O]\!]} [\![\tau]\!]^{[\![\tau']\!]}$$

is equal to:

$$[\![\mathrm{fv}(O[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{[\![O[x_1:=N_1,\ldots,x_n:=N_n]]\!]} [\![\tau]\!]^{[\![\tau']\!]}$$

and

$$[\![\mathrm{fv}(P[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{\hat{N}} [\![\mathrm{fv}(P)]\!] \xrightarrow{[\![P]\!]} [\![\tau']\!]$$

is equal to:

$$[\![\mathrm{fv}(P[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{[\![P[x_1:=N_1,\ldots,x_n:=N_n]]\!]} [\![\tau']\!]$$

where $\tilde{N} = \langle N_1 \circ \pi_{N_1}, \ldots, N_n \circ \pi_{N_1} \rangle$ contains only those N_i such that x_i occurs in O and $\hat{N}$ is the same for P. We can apply 1. and take the product of these morphisms to obtain the commutative diagram in Figure 1.8.
By the definition of $[\![M\ N]\!]$, we know that $[\![(O\ P)[x_1 \coloneqq N_1, \ldots]]\!]$ is equal to:

$$\mathrm{ev} \circ \langle [\![O[x_1 \coloneqq N_1, \ldots]]\!] \circ \pi_O, [\![P[x_1 \coloneqq N_1, \ldots]]\!] \circ \pi_P \rangle$$

and $[\![O\ P]\!]$ is:

$$\mathrm{ev} \circ \langle [\![O]\!] \circ \pi_O, [\![P]\!] \circ \pi_P \rangle$$

So:

$$[\![\mathrm{fv}((O\ P)[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{\langle \tilde{N}, \hat{N} \rangle} [\![\mathrm{fv}(O\ P)]\!] \xrightarrow{[\![O\ P]\!]} [\![\tau]\!]$$

is equal to:

$$[\![\mathrm{fv}((O\ P)[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{[\![(O\ P)[x_1:=N_1,\ldots]]\!]} [\![\tau]\!]$$

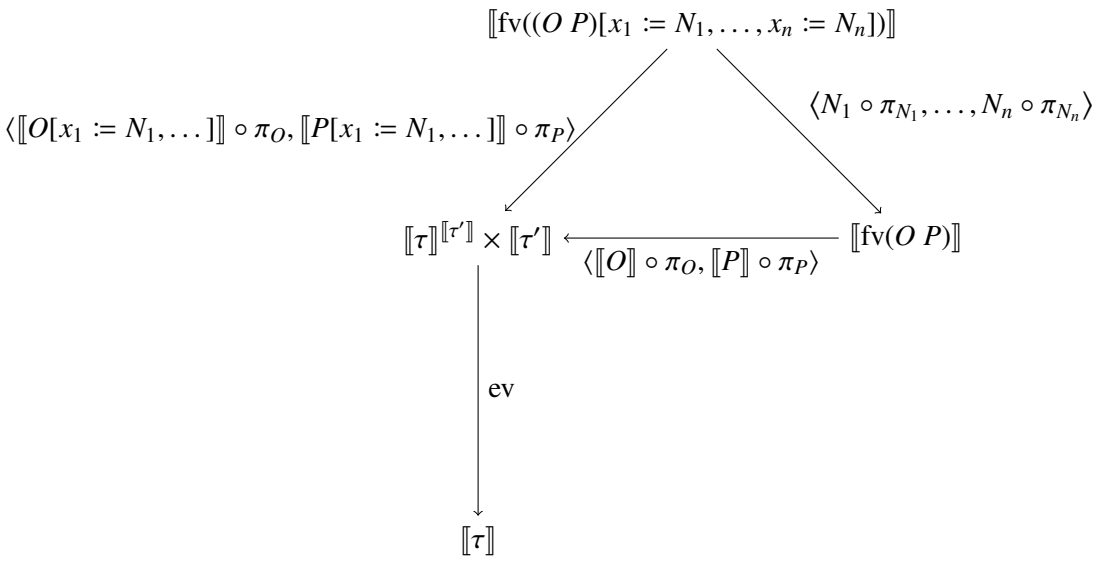


Figure 1.8: Substition for $[\![O\ P]\!]$.

- Suppose $M = \lambda x.O$ and has type $\tau' \to \tau$, and using the induction hypothesis we know:

$$[\![\mathrm{fv}(O[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{\langle N_1 \circ \pi_{N_1}, \ldots, N_n \circ \pi_{N_n} \rangle} [\![\mathrm{fv}(O)]\!] \xrightarrow{[\![O]\!]} [\![\tau]\!] \tag{1.3}$$

  is equal to:

$$[\![\mathrm{fv}(O[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \xrightarrow{[\![O[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n]]\!]} [\![\tau]\!] \tag{1.4}$$

  Note that $[\![\mathrm{fv}(O)]\!] = [\![\mathrm{fv}(M)]\!] \cup [\![\{x\}]\!]$ and:

$$[\![\mathrm{fv}(O[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] = [\![\mathrm{fv}(M[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \cup [\![\{x\}]\!]$$

  So we can rewrite the morphism in Equation (1.3) as:

$$[\![\mathrm{fv}(M[x_1 \coloneqq N_1, \ldots, x_n \coloneqq N_n])]\!] \cup [\![\{x\}]\!] \xrightarrow{\langle N_1 \circ \pi_{N_1}, \ldots, N_n \circ \pi_{N_n} \rangle} [\![\mathrm{fv}(M)]\!] \cup [\![\{x\}]\!] \xrightarrow{[\![O]\!]} [\![\tau]\!]$$

  By taking the transpose and by using that $[\![\{x\}]\!] = [\![\tau']\!]$ and that the transpose of $[\![O]\!]$ is $[\![M]\!]$ (by the definition of $[\![\lambda x.O]\!]$) we get:

$$[\![\mathrm{fv}(M[x_1 := N_1, \ldots, x_n := N_n])]\!] \xrightarrow{\langle N_1 \circ \pi_{N_1}, \ldots, N_n \circ \pi_{N_n} \rangle} [\![\mathrm{fv}(M)]\!] \xrightarrow{[\![M]\!]} [\![\tau]\!]^{[\![\tau']\!]}$$

We can do the same thing with the morphism in Equation (1.4) to obtain:

$$[\![\mathrm{fv}(M[x_1 := N_1, \ldots, x_n := N_n])]\!] \xrightarrow{[\![M[x_1 := N_1, \ldots, x_n := N_n]]\!]} [\![\tau]\!]^{[\![\tau']\!]}$$

5. Suppose M and N are terms of type $\tau_1 \to \tau_2$ and $x : \tau_1$ a variable that occurs in neither M nor N, and $\Gamma \cup \{x\} \vdash (M\ x) = (N\ x) : \tau_2$. Because $\Gamma \cup \{x\} = \Gamma - \{x\} \cup \{x\}$, the diagram in Figure 1.10 commutes. By taking the transpose of all of the morphisms we see that $\Gamma - \{x\} \vdash M = N : \tau_1 \to \tau_2$.

6. Suppose $M : \tau_1$ and $N : \tau_2$ are terms and $x : \tau_2$ a variable. By using 4., we know that:

$$[\![\mathrm{fv}(M[x := N])]\!] \xrightarrow{[\![M[x:=N]]\!]} [\![\tau_1]\!] = [\![\mathrm{fv}(M[x := N])]\!] \xrightarrow{\tilde{N}} [\![\mathrm{fv}(s)]\!] \xrightarrow{M} [\![\tau_1]\!]$$

This is equal to:

$$\begin{aligned} [\![\mathrm{fv}(M[x := N])]\!] &\xrightarrow{\langle \pi_M, \tilde{N} \rangle} [\![\mathrm{fv}(M) - \{x\}]\!] \times [\![\tau_2]\!] \\ &\xrightarrow{[\![\lambda x.M]\!] \times \mathrm{id}_{\tau_2}} [\![\tau_2 \to \tau_1]\!] \times [\![\tau_2]\!] \xrightarrow{\mathrm{ev}} [\![\tau_1]\!] \end{aligned}$$

which is:

$$[\![\mathrm{fv}((\lambda x.M)\ N)]\!] \xrightarrow{[\![(\lambda x.M)\ N)]\!]} [\![\tau_1]\!]$$

So our model generates a valid λ-calculus theory. □

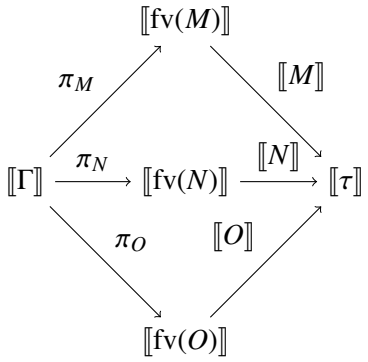


Figure 1.9: Transitivity of equality judgments.

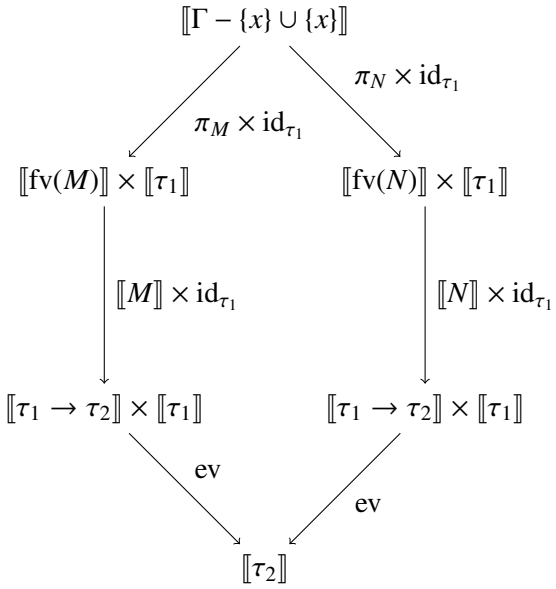


Figure 1.10: $\Gamma \cup \{x\} \vdash (M\ x) = (N\ x) : \tau_2$

Chapter 2

PCF

2.1 Introduction

The programing language PCF (*Programming Computable Functions*) was introduced by Dana Scott in 1967 and published in [Scott, 1993]. It is similar to a simply typed λ-calculus theory.

The set of ground types of PCF is:

$$\sigma = \{\iota, o\}$$

The type ι is the type of natural numbers, o the type of booleans.

The terms of PCF consist of the simply typed lambda calculus, using the following set of constants, C:

$\supset_o : o \to o \to o \to o$	(if-statement, result is a boolean)
$\supset_\iota : o \to \iota \to \iota \to \iota$	(if-statement, result is a number)
$Z : \iota \to o$	(test for zero)
$(+1) : \iota \to \iota$	(increment)
$(-1) : \iota \to \iota$	(decrement)
$\mathbf{Y}_\alpha : (\alpha \to \alpha) \to \alpha$	(least fixed-point)
$\Omega_\alpha : \alpha$	(undefined term)
$\mathrm{tt} : o$	(true)
$\mathrm{ff} : o$	(false)
$\{\overline{n} : \iota \mid n \in \mathbb{N}\}$	(number constant)

The α means that $\mathbf{Y}_\alpha$ and Ω_α exist for any type α.

2.2 Operational semantics for PCF

The *operational semantics* for PCF, which define the effect of executing a PCF program are similar to the β-reduction relation of the λ-calculus, except for the following: β-reduction may not (in general) be applied anywhere inside a PCF term. Inside the body of a λ-expression, β-reduction is not allowed (the λ-expression must be β-reduced first).

Formally, this is given by the following rewrite rules:

- $\dfrac{}{M \twoheadrightarrow_\beta M}$
- $\dfrac{M[x := N] \twoheadrightarrow_\beta O}{(\lambda x.M)\ N \twoheadrightarrow_\beta O}$
- $\dfrac{M \twoheadrightarrow_\beta M' \qquad M'\ N \twoheadrightarrow_\beta N'}{M\ N \twoheadrightarrow_\beta N'}$
- $\dfrac{M \twoheadrightarrow_\beta \mathrm{tt}}{\supset_o\ M\ N\ O \twoheadrightarrow_\beta N}$
- $\dfrac{M \twoheadrightarrow_\beta \mathrm{tt}}{\supset_\iota\ M\ N\ O \twoheadrightarrow_\beta N}$
- $\dfrac{M \twoheadrightarrow_\beta \mathrm{ff}}{\supset_o\ M\ N\ O \twoheadrightarrow_\beta O}$
- $\dfrac{M \twoheadrightarrow_\beta \mathrm{ff}}{\supset_\iota\ M\ N\ O \twoheadrightarrow_\beta O}$
- $\dfrac{M\ (\mathbf{Y}_\alpha\ M) \twoheadrightarrow_\beta N}{\mathbf{Y}_\alpha\ M \twoheadrightarrow_\beta N}$
- $\dfrac{}{\Omega_\alpha \twoheadrightarrow_\beta \Omega_\alpha}$
- $\dfrac{M \twoheadrightarrow_\beta \overline{0}}{Z\ M \twoheadrightarrow_\beta \mathrm{tt}}$
- $\dfrac{M \twoheadrightarrow_\beta \overline{n+1}}{Z\ M \twoheadrightarrow_\beta \mathrm{ff}}$
- $\dfrac{M \twoheadrightarrow_\beta \overline{n}}{(+1)M \twoheadrightarrow_\beta \overline{n+1}}$
- $\dfrac{M \twoheadrightarrow_\beta \overline{0}}{(-1)M \twoheadrightarrow_\beta \overline{0}}$

- $$\frac{M \twoheadrightarrow_\beta \overline{n+1}}{(-1)M \twoheadrightarrow_\beta \overline{n}}$$

Definition 2.1 Programs *in PCF are closed terms of ground type, so ι or o. From now on we shall use $N \Downarrow c$, with N a program and c a constant, to denote $N \twoheadrightarrow_\beta c$, and $N \Downarrow$ to mean $\exists c.N \Downarrow c$. $N \Uparrow$ means $\neg N \Downarrow$.*

Definition 2.2 *A* program context *$C[X]$ is a program using X as a meta-variable for a subterm. Contrary to normal λ-calculus application, filling in a program context does not use α-renaming to avoid free variables in X from becoming bound.*

The meaning of Ω_α is that it is a non-terminating computation, it can be seen as an infinite loop. This can be used for functions where no meaningful result is possible, but a term needs to be supplied with a given type anyway.

The $\mathbf{Y}_\alpha$ constant (also known as the *Y-combinator*) can be used to encode recursion. For example, in a language that allows recursive definitions, it is possible to write a function which has a call to itself:

$$f \coloneqq C[f]$$

To do this without recursive definitions, we can use $\mathbf{Y}_\alpha$ instead:

$$\mathbf{Y}_\alpha\ (\lambda f.C[f])$$

All other constants should be self-explanatory.

Definition 2.3 *Let M, N be terms of the same type α. Then M* observationally approximates *N, written $M \precsim N$, if for every program context $C[X]$, such that $C[M]$ and $C[N]$ are well-typed, if $C[M] \Downarrow v$, then $C[N] \Downarrow v$.*

If both $M \precsim N$ and $N \precsim M$, then M and N are observationally equivalent*, denoted $M \cong N$: in any context where M can be used, using N instead would lead to the same result.*

Observational equivalence is an important concept during program optimization: if M and N are observationally equivalent, then it is possible that evaluation using M requires fewer reduction steps than evaluation of N. Substituting M for N is allowed, because they are observationally equivalent, and it could improve the speed of the evaluation of the entire program.

Definition 2.4 *The* active subprogram *of a program M, if it exists, is defined by:*

- *If M has the form $((\lambda x.M_1)\ M_2\ M_3\ \ldots)$, $(\mathbf{Y}\ M_1)$, $(+1)\ \bar{n}$, $(-1)\ \bar{n}$, $Z\ \bar{n}$, $\supset_o\ c\ M_1\ M_2\ M_3$ then M is the active subprogram.*

- *If M has the form* $(+1)\ M_1$, $(-1)\ M_1$, $Z\ M_1$, $\supset_\iota\ M_1\ M_2\ M_3$ *or* $\supset_o\ M_1\ M_2\ M_3$ *(where M_1 is not a constant), then the active subprogram in M is the active subprogram in M_1 (which may not exist).*

- *If M has the form $\bar{n}$, tt or ff, then it has no active subprogram.*

Lemma 2.5 *If a program has an active subprogram and the program terminates, then the active subprogram terminates.*

Proof We prove this by case distinction on a term M.

- If M is the active subprogram of a program M, then this holds trivially.

- Suppose M is of the form $(+1)\ M_1$, $(-1)\ M_1$, $Z\ M_1$ or $\supset_\sigma\ \ M_1\ \ldots$, so the active subprogram is M_1. From the operational semantics rules at the beginning Section 2.2, it can be seen that any of these cases will recursively β-reduce M_1 first. So if M terminates, then the β-reduction of M_1 must have terminated.

- If the program has no active subprogram, then the lemma holds trivially.

□

We state the following lemma from [Plotkin, 1977, Ong, 1995]:

Lemma 2.6 (Activity Lemma) *Suppose:*

$$C\,[M_1, \ldots, M_m] \Downarrow c$$

where $M_1, \ldots, M_m$ are closed subterms of $C[M_1, \ldots, M_m]$. Then at least one of the following two statements holds:

1. $C\left[M'_1, \ldots, M'_m\right] \Downarrow c$ *for all closed terms $M'_1, \ldots, M'_m$ of appropriate types.*
2. *There is a context $D[, \ldots,]$, an integer $1 \leq i \leq m$ and integers $d_1, \ldots, d_k$ such that for all closed terms $M'_1, \ldots, M'_m$ of the appropriate types:*

$$C\left[M'_1, \ldots, M'_m\right] \twoheadrightarrow_\beta D\left[M'_{d_1}, \ldots, M'_{d_k}\right]$$

and the active subprogram in $D\left[M'_{d_1}, \ldots, M'_{d_k}\right]$ exists and either is the active subprogram in a term of the form $(M'_i\ \ldots)$, or has one of the forms $((\pm 1)\ M'_i)$, $(Z\ M'_i)$ or $(\supset_\sigma\ M'_i\ \ldots)$.

Lemma 2.7 *If $C[\Omega_\alpha] \Downarrow v$, then $C[M] \Downarrow v$ for all $M : \alpha$. Hence $\forall N.\Omega_\alpha \sqsubseteq N$.*

Proof Using the Activity Lemma, one of the following two statements must be true:

- $C[M'_1] \Downarrow v$ for all terms M'_1 of the appropriate type. In this case we are done.
- We are in the second case of the Activity Lemma. Note that M'_1 can not be the active subprogram in $D\left[M'_{d_1}\right]$ using lemma 2.5, as Ω_α does not terminate, but $C[\Omega_\alpha]$ does. So this case is impossible.

□

We will use the followig corollary of lemma 2.6:

Corollary 2.8 *If $M\ N_1\ \ldots\ N_n \Downarrow c$, then either $M\ \Omega_{\alpha_1}\ \ldots\ \Omega_{\alpha_n} \Downarrow c$, or there is an $1 \leq i \leq n$ such that $M\ N_1\ \ldots\ \Omega_{\alpha_i}\ \ldots\ N_n \Uparrow$. In other words, each function that uses its arguments, always reduces the same argument first.*

Proof Let $C[X_1, \ldots, X_n]$ be $M\ X_1\ \ldots\ X_n$ and apply lemma 2.6 to $C\ [N_1, \ldots, N_n] \Downarrow c$. Clearly, $X_1, \ldots, X_n$ are closed subterms of C. Now one of the two following statements is true:

- $C\left[X'_1, \ldots, X'_n\right] \Downarrow c$ for all closed terms X'_i of the appropriate types. This implies $M\ \Omega_{\alpha_1}\ \ldots\ \Omega_{\alpha_n} \Downarrow c$.
- There is a context $D[, \ldots,]$, an integer $1 \leq i \leq n$ and integers $d_1, \ldots, d_k$ such that, for all X' of the appropriate type:

$$C\left[X'_1, \ldots, X'_n\right] \twoheadrightarrow_\beta D\left[X'_{d_1}, \ldots, X'_{d_k}\right]$$

  and the active subprogram of D is the active subprogram in X'_i, or has one of the following forms: $((\pm 1)X'_i)$, $(Z\ X'_i)$ or $\supset_\sigma\ \ X'_i\ \ldots$. This means that $D \Uparrow$ when $X'_i \Uparrow$, which implies:

$$M\ X'_1\ \ldots\ \Omega_{\alpha_i}\ \ldots\ X'_n \Uparrow$$

  for all X'_j of the appropriate types.

□

2.3 Denotational semantics for PCF

The operational semantics defines the effect of evaluating a program, but that is often an inconvenient way of analyzing the behavior of programs. It is useful to construct *denotational semantics* instead. The goal of denotational semantics is to analyze the behavior of programs by giving an interpretation of the types and terms.

We shall restrict ourself to denotational semantics that form a λ-calculus theory from definition 1.11, by treating PCF as a λ-calculus with constants. We shall use $[\![.]\!]$ to denote the function that translates PCF types and terms to their interpretation in the denotational semantics. We assume there is a partial order $\sqsubseteq$ on the interpretation of terms.

Ideally, the partial order on the denotational semantics and the partial order generated by observational equivalence coincide: instead of evaluating programs to test for equivalence, we can check whether their interpretation is equal in the denotational semantics. If the partial order of some denotational semantics coincide with the observational equivalence, then it is said to be *fully abstract*:

$$[\![M]\!] \sqsubseteq [\![N]\!] \Leftrightarrow M \precsim N$$

If we only have the $\Rightarrow$ implication, then the denotational semantics are *adequate*:

$$[\![M]\!] \sqsubseteq [\![N]\!] \Rightarrow M \precsim N$$

Denotational semantics that are not adequate are not useful as a model: the model could indicate that two PCF terms are equivalent while they are not.

Suppose the adequate denotational semantics $[\![.]\!]$ contains a bottom element, $\bot$. Let M be a PCF term such that $[\![M]\!] = \bot$. Then:

$$\forall N.[\![M]\!] \sqsubseteq [\![N]\!]$$

So, by the adequacy:

$$\forall N.M \precsim N$$

Because $\forall N.\Omega_\alpha \precsim N$, this means $\Omega_\alpha \cong M$. This means:

$$[\![M]\!] = \bot \Leftrightarrow M \cong \Omega_\alpha$$

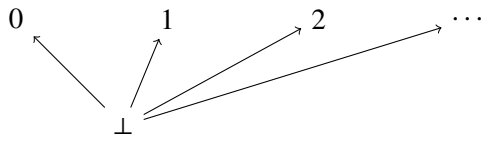


Figure 2.1: $\mathbb{N}_\perp$

2.4 Scott-domain model

The first denotational semantics for PCF we will consider is the *Scott-domain model*. Informally, in this model types are Scott-domains and functions are Scott-continuous functions between Scott-domains. This forms a Cartesian closed category according to theorem 1.42, and so also a λ-calculus theory following Section 1.4.

Definition 2.9 *A* flat domain *is a domain with a $\perp$ where all elements are incomparable, except $\perp$ which is below all other elements.*

For example, $\mathbb{N}_\perp$ is a flat domain where the elements are from $\mathbb{N}$, as can be seen in Figure 2.1. It is easy to see that this is a Scott-domain. First note that the directed subsets of $\mathbb{N}_\perp$ are $\{\perp\} \cup \{\{\perp, n\} \mid n \in \mathbb{N}\} \cup \{\{n\} \mid n \in \mathbb{N}\}$.

- $\mathbb{N}_\perp$ is directed complete: suppose D is a directed subset of $\mathbb{N}_\perp$. Then either there is a $n \in D$ such that $n \in \mathbb{N}$ or $D = \{\perp\}$. In the first case n is the supremum of D, otherwise, $\perp$ is.
- It is bounded complete: if a subset $D \subseteq \mathbb{N}_\perp$ has an upper bound, then D contains at most one $n \in \mathbb{N}$. If $D = \{\perp\}$, then $\perp$ is the supremum of D. If it contains a $n \in \mathbb{N}$, then n is the supremum.
- Every element of $\mathbb{N}_\perp$ is compact, so every element can (trivially) be obtained as the supremum of a directed set of compact elements of $\mathbb{N}_\perp$. So $\mathbb{N}_\perp$ is algebraic.

Similarly, $\mathbb{B}_\perp = \{\text{true}, \text{false}, \perp\}$, with $\perp \sqsubseteq \text{true}$, $\perp \sqsubseteq \text{false}$ is also a flat Scott-domain.

The category of Scott-domains is Cartesian closed and therefore, by the Curry-Howard-Lambek isomorphism, corresponds to a λ-calculus theory. The reduction relation for PCF is not as powerful as normal β-reduction, however, every PCF reduction is a valid β-reduction, so we can consider PCF as a λ-calculus theory, therefore also forms a Cartesian closed category. However, these categories and theories do not coincide. Instead, we will look for a subcategory of the category of Scott-domains that does coincide with PCF.

We do this by translating all types and terms as follows:

- $[\![\iota]\!] = \mathbb{N}_\perp$
- $[\![o]\!] = \mathbb{B}_\perp$
- $[\![\sigma \to \tau]\!]$ = the Scott-domain of Scott-continuous functions from $[\![\sigma]\!]$ to $[\![\tau]\!]$, using the pointwise function ordering.
- $[\![\supset_o]\!](b,t,e) = \begin{cases} [\![b]\!] = \text{true} & [\![t]\!] \\ [\![b]\!] = \text{false} & [\![e]\!] \\ [\![b]\!] = \perp & \perp \end{cases}$
- $[\![\supset_\iota]\!](b,t,e) = \begin{cases} [\![b]\!] = \text{false} & [\![e]\!] \\ [\![b]\!] = \perp & \perp \end{cases}$
- $[\![Z]\!](x) = \begin{cases} [\![x]\!] = 0 & \text{true} \\ [\![x]\!] = n+1 & \text{false} \\ [\![x]\!] = \perp & \perp \end{cases}$
- $[\![(+1)]\!](x) = \begin{cases} [\![x]\!] = n & n+1 \\ [\![x]\!] = \perp & \perp \end{cases}$
- $[\![(-1)]\!](x) = \begin{cases} [\![x]\!] = 0 & 0 \\ [\![x]\!] = n+1 & n \\ [\![x]\!] = \perp & \perp \end{cases}$
- $[\![\mathbf{Y}_\alpha]\!](f) = \bigsqcup_{i=0}^{\infty} \left\{ [\![f]\!]^i (\perp) \right\}$
- $[\![\Omega_\alpha]\!] = \perp$
- $[\![\text{tt}]\!] = \text{true}$
- $[\![\text{ff}]\!] = \text{false}$
- $[\![\overline{n}]\!] = n$

This model is adequate, but not fully abstract. For example, we can define the Scott-continuous function:

$$\text{por}(x,y) = \begin{cases} \text{true} & \text{if } x = \text{true} \\ \text{true} & \text{if } y = \text{true} \\ \text{false} & \text{otherwise} \end{cases}$$

To show that this function is Scott-continuous, see Figure 2.2. By moving up (following an arrow) from (a,b) to (a',b'), the result increases: $\text{por}(a,b) \sqsubseteq \text{por}(a',b')$, so por is monotone and preserves directed suprema.

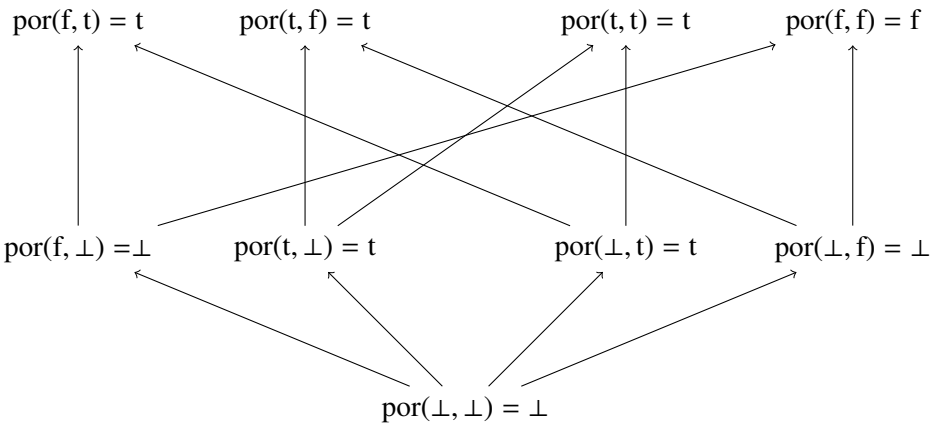


Figure 2.2: Scott-continuity of por.

However, we can not find a PCF-function that corresponds with por: using corollary 2.8:

- Either there is a v such that por x y $\Downarrow$ v for all x, y, which is not true as por tt tt $\Downarrow$ tt but por ff ff $\Downarrow$ ff.
- Or por Ω_o tt $\Uparrow$ or por tt Ω_o $\Uparrow$.

2.5 The Full Abstraction Problem

The problem of finding a programming language, similar to PCF, with fully abstract denotational semantics has had two approaches:

- "Strengthening" the language by adding more constructs such that the Scott-domain model becomes fully abstract for that new language.
- Finding "weaker" denotational semantics, the approach we will take in Chapter Chapter 3.

2.5.1 Strengthening: parallel conditional

In [Plotkin, 1977] the strengthening approach was taken, by introducing the parallel conditional, $:\supset_\sigma: o \to \sigma \to \sigma \to \sigma$. Its reduction satisfies:

$$:\supset_\sigma \; b\; t\; e \to_\beta \begin{cases} t & \text{if } b = \text{tt} \\ e & \text{if } b = \text{ff} \\ t & \text{if } b \Uparrow \text{ and } t = e \\ \Omega_\sigma & \text{otherwise} \end{cases}$$

Clearly, $:\supset_\sigma$ is non-deterministic. However, the outcome can not depend on the choice made. If $\supset_\sigma \;\; b\ t\ e \Downarrow c$, then $:\supset_\sigma \;\; b\ t\ e \Downarrow c$ for any sequence of choices.

We can define por as:

$$\mathrm{por}(x, y) = :\supset_o \;\; x\ \mathrm{tt}\ y$$

[Plotkin, 1977] showed that PCF+$:\supset_\sigma$ is fully abstract for the Scott-domain model.

2.5.2 The stable function model

In [Berry, 1978], another approach was introduced, which uses a category with dI-domains as objects and stable functions are morphisms instead of Scott-domains. However, recall that the exponentials of dI-domains were ordered by a different order, the stable function order, which does not coincide with the pointwise function order.

The por function is not stable, therefore does not exist in the stable function model. Consider $(\mathrm{ff}, \bot)$ and $(\bot, \mathrm{ff})$, which are bounded:

$$\begin{aligned} \mathrm{por}((\mathrm{ff}, \bot) \sqcap (\bot, \mathrm{ff})) &= \mathrm{por}(\mathrm{ff}, \mathrm{ff}) = \mathrm{ff} \\ \mathrm{por}(\mathrm{ff}, \bot) \sqcap \mathrm{por}(\bot, \mathrm{ff}) &= \bot \sqcap \bot = \bot \end{aligned}$$

However, there are other functions which are stable, but not PCF-definable. Consider the function, known as Gustave's function, attributed to [Berry, 1978] in [Huet, 1986], defined by:

$$f(x, y, z) = \begin{cases} \mathrm{tt} & \text{if } y = \mathrm{tt} \text{ and } z = \mathrm{ff} \\ \mathrm{tt} & \text{if } z = \mathrm{tt} \text{ and } x = \mathrm{ff} \\ \mathrm{tt} & \text{if } x = \mathrm{tt} \text{ and } y = \mathrm{ff} \\ \mathrm{ff} & \text{if } x = \mathrm{ff},\ y = \mathrm{ff} \text{ and } z = \mathrm{ff} \\ \bot & \text{otherwise} \end{cases}$$

Using corollary 2.8, it is easy to see that f is not PCF-definable: suppose F is a PCF term that encodes f, then there is no v such that $F\ x\ y\ z \Downarrow v$ for all x, y, z, so one of the following must be true:

- $F\ \Omega_o\ y\ z \Uparrow$, but $F\ \Omega_o\ \mathrm{tt}\ \mathrm{ff} \Downarrow \mathrm{tt}$.
- $F\ x\ \Omega_o\ z \Uparrow$, but $F\ \mathrm{tt}\ \Omega_o\ \mathrm{ff} \Downarrow \mathrm{tt}$.
- $F\ x\ y\ \Omega_o \Uparrow$, but $F\ \mathrm{tt}\ \mathrm{ff}\ \Omega_o \Downarrow \mathrm{tt}$.

This is not possible, so F can not exist. However, f is Scott-continuous: if $f(x, y, z) = \text{tt}$, then $f(x', y', z') = \text{tt}$ for all $x \sqsubseteq x'$, $y \sqsubseteq y'$ and $z \sqsubseteq z'$. If $f(x, y, z) = \text{ff}$, then $(x, y, z) = (\text{tt}, \text{tt}, \text{tt})$ or $(x, y, z) = (\text{ff}, \text{ff}, \text{ff})$.

By checking at all possible values for x and y (which we will omit here), it can be shown that:

$$x \uparrow y \Rightarrow f(x \sqcap y) = f(x) \sqcap f(y)$$

so f is stable.

So the stable function model is neither fully abstract, nor *order-extensional* (ordered by the pointwise order).

Chapter 3

Dialogue Games

The weaker model for PCF we will describe will represent types as *dialogue games* and PCF-terms as *strategies* for those games. The approach in this chapter is from [Hyland and Ong, 2000].

A dialogue game is a game played by two players, Player (P) and Opponent (O) who can either ask a question or answer a question each turn. The game starts with a question from Opponent and after that the turns alternate. In a turn, the player may either answer an unanswered question from the other player, or ask the other player a new question, if that question is allowed at that point in the game. Once the initial question has been answered, the game is over.

We can formalize this as:

Definition 3.1 *A* dialogue game *A consists of a forest of questions* $\mathrm{Qn}(A)$*, a set of possible answers* $\mathrm{Ans}(A)$ *and a function* $\mathrm{qn}_A : \mathrm{Ans}(A) \to \mathrm{Qn}(A)$ *that maps answers to the question they are appropriate for.*

The roots of the forest are the *initial questions*. Every answer and every question that is not an initial question has a *justifying question*: for questions, the justifying question is the parent in the tree of questions, for answers a, $\mathrm{qn}(a)$ is the justifying question. A question or answer may only be given if its justifying question is still unanswered.

We shall use "(" to denote questions asked by P, ")" for answers by O, "[" for questions by O and "]" for answers by P. This syntax is chosen because well-formed sequences of a full game will be well-formed sequences of parentheses.

We also need to store the information about which question justifies each question or answer. We shall do this by a sequence of *justification pointers*. The initial question has a justification pointer of 0, all other questions have a justification pointer that is equal to the index of their justifying question. A move r is *indirectly justified* by a question s if s occurs by following the chain of justification pointers from r. If we modify a sequence in any way, we must make sure the justification pointers are updated accordingly.

For example:

$$[\cdot (\cdot) \cdot (\cdot [\cdot] \cdot) \cdot] \qquad 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 5 \cdot 4 \cdot 1$$

Definition 3.2 *A sequence of moves is* well-formed *if:*

- Alternating play*: The moves alternate between players.*
- Explicit justification*: Except the initial question, each move has a valid justification pointer (the justification pointer points to an instance of the justifying question). An answer or question may only be given or asked if its justifying question has been asked, but not yet answered.*
- Last asked, first answered*: A question may only be answered if its justifying question is the last question which is still unanswered in the sequence of moves.*

Definition 3.3 *A well-formed sequence is* maximal *if the initial question is answered. By the "Last asked, first answered" condition, this must mean every question has been answered.*

3.1 Example arenas

As an example, the boolean computational arena $\mathbb{B}$ is given by:

- A single question, which is the initial question [.
- Two answers, $]_\mathrm{t}$ and $]_\mathrm{f}$.

The natural numbers computational arena $\mathbb{N}$ is defined as:

- The only question is the initial question [.
- The answers are $\{]_i \mid i \in \mathbb{N}\}$.

3.2 O-view and P-view

Before we can define the strategies that the players apply to the game, we must define the *O-view* and the *P-view* of a sequence of moves.

Informally, the P-view (respectively O-view) leaves out the questions from O (resp. P) that have already been answered. In other words, to answer a question asked by O, P may only use the *answers* given by O to the questions P asks, justified by O's question. P may not use the counter-questions (or any question indirectly justified by those) that O asked.

Definition 3.4 *The P-view* $\ulcorner . \urcorner$ *of a sequence of moves is defined by recursion on the last move:*

$$
\begin{aligned}
\ulcorner [\urcorner &= [&& \text{If “[” is the initial question.} \\
\ulcorner q \cdot (\cdot\ r \cdot [\urcorner &= \ulcorner q \urcorner \cdot (\cdot [&& \text{If “(” is the justifying question of “[”.} \\
\ulcorner q \cdot) \urcorner &= \ulcorner q \urcorner \cdot) \\
\ulcorner q \cdot [\cdot r \cdot] \urcorner &= \ulcorner q \urcorner && \text{If “]” answers “[”.} \\
\ulcorner q \cdot (\urcorner &= \ulcorner q \urcorner \cdot (
\end{aligned}
$$

Notice that the P-view removes two things: sequences [. . .], where] is justified by [and r in $(\cdot r \cdot [$ if (justifies [. This means [is justified by (. This implies the P-view is idempotent: $\ulcorner \ulcorner p \urcorner \urcorner = \ulcorner p \urcorner$.

Similarly, the O-view $\llcorner . \lrcorner$ is defined as:

$$
\begin{aligned}
\llcorner q \cdot [\cdot r (\lrcorner &= \llcorner q \lrcorner \cdot [\cdot (&& \text{If “[” justifies “(”.} \\
\llcorner q \cdot] \lrcorner &= \llcorner q \lrcorner \cdot] \\
\llcorner q \cdot (\cdot r \cdot) \lrcorner &= \llcorner q \lrcorner && \text{If “)” answers “(”.} \\
\llcorner q \cdot [\lrcorner &= \llcorner q \lrcorner \cdot [
\end{aligned}
$$

The P-view or the O-view of a well-formed sequence is not always a well-formed sequence. We shall therefore also define sequences that are legal.

Definition 3.5 *A sequence r is* legal *if it satisfies the* visibility condition*: for every initial subsequence $s \cdot ($ the O-question “[” justifying “(” occurs in the P-view of s and for every initial subsequence $s \cdot [$ the P-question “(” justifying “[” occurs in the O-view of s.*

3.3 Product and function space arena

We can take the product of two computational arenas as:

$$
\begin{aligned}
\mathrm{Qn}(A \times B) &\coloneqq \mathrm{Qn}(A) + \mathrm{Qn}(B) \\
\mathrm{Ans}(A \times B) &\coloneqq \mathrm{Ans}(A) + \mathrm{Ans}(B) \\
\mathrm{qn}(A \times B) &\coloneqq \mathrm{qn}(A) + \mathrm{qn}(B)
\end{aligned}
$$

In other words, Opponent can choose from the initial questions of A and B. After choosing a question the game continues only in that question's arena (as the only questions justified by an initial question of A are in A).

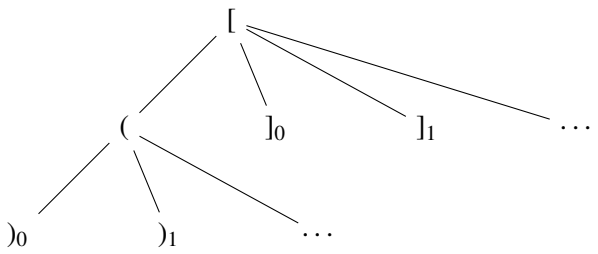


Figure 3.1: The function space arena $\iota \Rightarrow \iota$.

We define the function space arena $A \Rightarrow B$ as follows:

$$\begin{aligned} \mathrm{Qn}(A \Rightarrow B) &:= (\mathrm{Qn}(A) \times M_B) + \mathrm{Qn}(B) \\ \mathrm{Ans}(A \Rightarrow B) &:= (\mathrm{Ans}(A) \times M_B) + \mathrm{Ans}(B) \\ \mathrm{qn}(A \Rightarrow B) &:= (\mathrm{qn}(A) \times \mathrm{Id}_M) + \mathrm{qn}(B) \end{aligned}$$

where M_B are the initial questions of B.

A function space arena $A \Rightarrow B$ consists of the forest of questions of B, with underneath each initial question in B a copy of the forest of questions of A. Note that this means that the Opponent and Player are reversed in the copies of A: the initial questions of A have become P-questions. For example, see Figure 3.1. Possible plays in this arena include $[\cdot]_0$, $[\cdot(\cdot)_5\cdot]_8$ and $[\cdot(\cdot)_4\ \cdot(\cdot)_6\cdot]_{10}$.

We will use $s \upharpoonright B$ to denote the projection of all the B-moves of a function space arena and $s \upharpoonright (A, a)$ to denote the projection of the A-subgame justified by the move a. We shall use $s \upharpoonright (A, a)^+$ to denote $m \cdot s \upharpoonright (A, a)$, where m is the initial B-move of s.

Lemma 3.6 *Let $b_0 \cdot s$ be a legal position of the arena $A \Rightarrow B$ such that all moves in s belong to the component (A, a). Then:*

- $\ulcorner b_0 \cdot s \urcorner^{A \Rightarrow B} = b_0 \cdot \llcorner s \lrcorner_A$.
- $\llcorner b_0 \cdot s \lrcorner^{A \Rightarrow B} = b_0 \cdot \ulcorner s \urcorner_A$.

Proof This follows from the definition of $\ulcorner . \urcorner$ and $\llcorner . \lrcorner$ and the definition of the $A \Rightarrow B$ arena.

Theorem 3.7 (Switching convention) *In a function space arena, Player, not Opponent is allowed to switch components, either between a B-component and an (A, a)-component or between different (A, a)-components.*

Proof To prove this, we use the following three lemmas:

Let s be a legal position of the arena $A \Rightarrow B$ beginning with an initial move b and ending with a P-move m.

Lemma 3.8 *For every two successive moves $m \cdot m'$ in s, if m and m' are in different components, then m is an O-move and m' a P-move.*

Lemma 3.9 *If m is in B, then $\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright B = \llcorner s \upharpoonright B \lrcorner = \llcorner s \lrcorner_{A \Rightarrow B}$.*

Lemma 3.10 *If m is in an (A, a)-component, then $\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright (A, a)^+ = b \cdot \ulcorner s \upharpoonright (A, a)^{\neg A} = \llcorner s \lrcorner_{A \Rightarrow B}$.*

We will prove lemmas 3.8 to 3.10 together using recursion on the length of s. Theorem 3.7 follows directly from lemma 3.8.

Suppose $s = b \cdot m$. Lemma 3.8 is trivially true. Suppose m is in B, then $\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright B = \llcorner s \upharpoonright B \lrcorner = \llcorner s \lrcorner_{A \Rightarrow B} = b \cdot m$, so lemma 3.9 is true. Suppose m is in (A, a), then $\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright (A, a)^+ = b \cdot \ulcorner s \upharpoonright (A, a)^{\neg A} = \llcorner s \lrcorner_{A \Rightarrow B} = b \cdot m$.

Now for the inductive case. Suppose m is a B-move, let m^- be the move explicitly justifying m. Let $s_{<m^-}$ be s up to m^-. Because $s_{<m^-}$ is strictly smaller than s, the induction hypothesis of lemma 3.8 implies that $s_{<m^-}$ ends with a B-move.

$$
\begin{aligned}
\llcorner s \lrcorner \upharpoonright B &= \llcorner s_{<m^-} \lrcorner \upharpoonright B \cdot m^- \cdot m \qquad \text{(Using the induction hypothesis of lemma 3.9.)} \\
&= \llcorner s_{<m^-} \upharpoonright B \lrcorner \cdot m^- \cdot m \\
&= \llcorner \{ s_{<m^-} \cdot m^- \cdot m \} \upharpoonright B \lrcorner \\
&= \llcorner s \upharpoonright B \lrcorner
\end{aligned}
$$

Suppose m is a initial move in a new (A, a)-component. Then $\llcorner s \lrcorner = b \cdot m$, so $\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright (A, a)^+ = b \cdot m = b \cdot \ulcorner s \urcorner \upharpoonright (A, a)$.

Suppose m is a move in an existing (A, a)-component. Let m^- be the O-move explicitly justifying m, which must also be in the same (A, a)-component. Then $\llcorner s \lrcorner = \llcorner s_{<m^-} \lrcorner \cdot m^- \cdot m$, so we can apply the induction hypothesis of lemma 3.10:

$$
\begin{aligned}
\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright (A, a)^+ &= b \cdot \ulcorner s_{<m^-} \upharpoonright (A, a)^{\neg A} \cdot m^- \cdot m \\
&= b \cdot \ulcorner s \upharpoonright (A, a)^{\neg A}
\end{aligned}
$$

and:

$$
\llcorner s \lrcorner_{A \Rightarrow B} \upharpoonright (A, a)^+ = \llcorner s \lrcorner_{A \Rightarrow B}
$$

Now to prove lemma 3.8, suppose s is a legal position where the last move m is an O-move and let m^- be the P-move preceding m. Suppose m^- is in (A, a), then by the induction hypothesis of lemma 3.10, $\llcorner s_{\leq m^-} \lrcorner = b \cdot p$, where p is a sequence of moves entirely in (A, a). There must be a move that explicitly justifies m in the O-view, so either that move is b, or some move in p. Because m is an O-move, it can not be justified by b. So m is explicitly justified by a move in the (A, a)-component, so must be in the (A, a)-component.

Now suppose m^- is in B, then by the induction hypothesis of lemma 3.9, $\llcorner s_{\leq m^-} \lrcorner_{A \Rightarrow B} = \llcorner s_{\leq m^-} \lrcorner_{A \Rightarrow B} \upharpoonright B$. Then m must be explicitly justified by one of the moves in $\llcorner s_{\leq m^-} \lrcorner_{A \Rightarrow B} \upharpoonright B$, therefore must be in B. □

We will state without proof the following lemma from [Hyland and Ong, 2000].

Lemma 3.11 *Let s be a legal position of an arena $A \Rightarrow B$ ending with the move m.*

- *If m is in B, then $\ulcorner s \urcorner^{A \Rightarrow B} \upharpoonright B$ is a subsequence of $\ulcorner s \upharpoonright B \urcorner^B$.*
- *If m is in the component (A, a), then $\ulcorner s \urcorner^{A \Rightarrow B} \upharpoonright (A, a)^+$ is a subsequence of $b \cdot \ulcorner s \upharpoonright (A, a) \urcorner^A$.*

Theorem 3.12 (Projection convention) *For any legal position s in $A \Rightarrow B$, $s \upharpoonright B$ is a legal position in B. $s \upharpoonright (A, a)$ is a legal position in A.*

Proof We will show that $s \upharpoonright B$ satisfies definitions 3.2 and 3.5. By the definition of $A \Rightarrow B$ it must start with an initial B-question, so $s \upharpoonright B$ starts with an initial question too. By using theorem 3.7, it is easy to see that the moves in $s \upharpoonright B$ must be alternating: if P switched to a different component, then it must also be P who switches back to B.

Every move "(" (resp. "[") in s must have a justification pointer pointing to a move "[" (resp. "(") in B. Though the pointer may have changed, that move is still in $s \upharpoonright B$, so it has a justification pointer.

To prove the visibility condition, let "(" be a P-question in $s \upharpoonright B$. Then there must be a corresponding question "(" in s with a justifying question "[" in $\ulcorner s_{<(} \urcorner$. As "[" must be a B move, it will also occur in $\ulcorner s_{<(} \urcorner \upharpoonright B$, so by lemma 3.11 it occurs in $\ulcorner s_{<(} \upharpoonright B \urcorner$.

Now for $s \upharpoonright (A, a)$. This is by definition the sequence of moves justified by an instance of a initial A-move, so it starts with an initial move. Theorem 3.7 implies again that the sequence of moves $s \upharpoonright (A, a)$ is alternating: only P may switch to a different component and only P may switch back to (A, a). Every non-initial move in (A, a) must be explicitly justified by another move in (A, a), so every move in $s \upharpoonright (A, a)$ also has a valid justification pointer. Now to show the visibility condition, let s be a legal position in $A \Rightarrow B$ and b its initial B-move. Let m be a P-question in $s \upharpoonright (A, a)$, regarded as an A-arena. By regarding it as an O-question in the $A \Rightarrow B$-arena there must be a P-question $\underline{m}$ in $\llcorner s_{<m} \lrcorner_{A \Rightarrow B}$ that explicitly justifies it. $\underline{m}$ is in $\llcorner s_{<m} \lrcorner \upharpoonright (A, a)^+$, which is $\llcorner b \cdot \{s_{<m}\} \upharpoonright (A, a) \lrcorner$. Because every move in $s_{<m} \upharpoonright (A, a)$ is in (A, a), we can apply lemma 3.6 to see that $\underline{m}$ is in $\ulcorner s_{<m} \upharpoonright (A, a) \urcorner$. □

3.4 Innocent strategies

Definition 3.13 *A* P-strategy σ *determines at every position where P needs to move what P-move to play.*

We shall represent P-strategies as a subtree of the game tree, such that:

- *Determinacy*: For every position $s \in \sigma$ where P needs to move, the strategy selects at most one move: if $s \cdot a \in \sigma$ and $s \cdot b \in \sigma$ then $a = b$.
- *Contingent completeness*: For any $s \in \sigma$ where O needs to move, $s \cdot a$ is in σ for every O-move a such that $s \cdot a$ is a legal position.

For example, a strategy in the natural number computational arena that always returns 0 is represented as:

$$\{[, [\cdot]_0\}$$

The identity-strategy in the computational arena $\mathbb{N} \Rightarrow \mathbb{N}$ is represented as:

$$\{[\cdot () \cup \{[\cdot (\cdot)_n \cdot]_n \mid n \in \mathbb{N}\}$$

And the strategy in the arena $\mathbb{N} \Rightarrow \mathbb{B}$ that returns true if the input is 0, and false otherwise:

$$\{[\cdot (, [\cdot (\cdot)_0 \cdot]_\mathrm{t}\} \cup \{[\cdot (\cdot)_n \cdot]_\mathrm{f} \mid n > 0\}$$

Definition 3.14 *Let σ be a strategy in the computational arena $A \Rightarrow B$, τ a strategy in $B \Rightarrow C$ and s a legal position of the arena $A \Rightarrow C$. We define the* uncovering *of s in accord with σ and τ, denoted $\mathbf{u}(s, \sigma, \tau)$, as the unique maximal sequence u satisfying:*

$$u \upharpoonright (A, C) \leq s$$
$$u \upharpoonright (B, C) \in \tau$$
$$u \upharpoonright (A, B)_b \in \sigma$$

where $\leq$ in the first condition is the prefix ordering between legal positions and b in the third condition ranges over all instances of initial moves in B occurring in u.

In the second and third condition the projected part of u could be infinite. In that case, the condition requires all finite prefixes of $u \upharpoonright (B, C)$ and $u \upharpoonright (A, B)_b$ to be in τ and σ, respectively.

The sequence u consists of moves from the arenas A, B and C. Informally, the uncovering sequence represents the following:

1. O opens with an initial question m from C.

2. P starts a temporary game in the arena $B \Rightarrow C$ on a "scratch-pad", where P is playing against themselves. P copies the opening move m and uses the strategy τ to find a response:

   (a) If τ gives an C-answer, then P uses the same move in the $A \Rightarrow C$ arena, which means the game finishes.

   (b) If τ asks an initial B-question m' in a B-component, then P starts a new temporary game, now in $A \Rightarrow B$, using the strategy σ and initial move m':

      i. If σ gives a B-answer, then P uses that answer in the $B \Rightarrow C$ arena.

      ii. If σ asks an initial A-question, then P copies that question to a new A-component of the "real" $A \Rightarrow C$ arena to get an answer from O for that question. P copies that answer back to the temporary $A \Rightarrow B$ arena.

      P continues following strategy σ until 2.b.i is reached.

   P continues following strategy τ until 2.a is reached, creating new $A \Rightarrow B$ "scratch-pads" for each initial B-move from τ in $B \Rightarrow C$.

The sequence u, by itself, is not a legal position, as the B-moves don't belong to a player. However, by projecting out just the moves from B and C we get a legal position in $B \Rightarrow C$. By projecting just the moves from A and B we get a number of legal positions in $A \Rightarrow B$ (each with a different instance of an initial B-move) and by projecting out just the moves from A and C we get a legal position that is a prefix of s.

Definition 3.15 *The* composition *of strategies σ in the arena $A \Rightarrow B$ and τ in $B \Rightarrow C$ is given by:*

$$\{\mathbf{u}(s, \sigma, \tau) \upharpoonright (A, C) \mid s \text{ is a legal position of } A \Rightarrow C\}$$

It is denoted $\sigma; \tau$.

Definition 3.16 *A strategy is* innocent *if there is a partial function f, such that for any legal position $s \in \sigma$ at which P is to move, and for any P-move a with justification pointer ρ, $s \cdot a$ is in σ if and only if $f(\ulcorner s \urcorner) = \langle a, \rho' \rangle$ and ρ coincides with the transposed pointer of ρ'.*

In other words, the strategy may only use the P-view of the history to determine P's next move, not the entire history. The transposition of the justification pointer is necessary because the P-view $\ulcorner s \urcorner$ has different pointers than the normal sequence s.

Definition 3.17 *For any innocent strategy σ, there is a least partial function f_σ that corresponds to σ, called the* representing innocent function.

Theorem 3.18 *The innocent strategies in an arena A form a dI-domain, denoted* $\underline{A}$*, ordered by inclusion.*

Proof First, we show that $\underline{A}$ is bounded complete. Let S be a set of innocent functions with an upper bound τ. Then for all $\sigma \in S$, $f_\sigma \subseteq f_\tau$, so $F = \bigcup_{\sigma \in S} f_\sigma$ is a representing function for an innocent strategy and $\sigma_F = \bigsqcup S$.

Now we show that $\underline{A}$ is directed complete. It is easy to see that the inclusion ordering on innocent strategies is equal to the pointwise ordering on the respective representing functions. As every poset of functions ordered pointwise is directed complete, $\underline{A}$ must also be directed complete.

Now we show that $\underline{A}$ is algebraic, and even prime algebraic. Let $\boldsymbol{\sigma}[s]$ denote the least innocent strategy containing a position s. Then for any innocent strategy σ:

$$\bigsqcup_{(p,a) \in f_\sigma} \boldsymbol{\sigma}[p \cdot a] = \sigma \tag{3.1}$$

If σ is prime, then $\sigma \sqsubseteq \boldsymbol{\sigma}[p \cdot a]$ for some $(p, a) \in f_\sigma$, but as $\boldsymbol{\sigma}[p \cdot a]$ is the least strategy containing $p \cdot a$, this means $\sigma = \boldsymbol{\sigma}[p \cdot a]$. So every prime strategy is equal to $\boldsymbol{\sigma}[p \cdot a]$ for some P-view p and P-move a. Using Equation (3.1) it is easy to see that $\underline{A}$ is prime algebraic.

From lemma 3.19 it follows easily that $\underline{A}$ satisfies axiom I from definition 1.44, as a finite graph can only have finitely many subgraphs.

We already showed that $\underline{A}$ is prime algebraic, so it satisfies axiom d by theorem 1.51.

So $\underline{A}$ is a dI-domain. □

Lemma 3.19 *An innocent strategy* $\sigma \in \underline{A}$ *is a compact element if and only if its representing function* f_σ *is a finite graph.*

Theorem 3.20 *The category of computational arenas using innocent strategies as morphisms is Cartesian closed.*

Proof

- The identity functions are the following strategy in $A \to A$: when O starts with an initial move m, P responds by creating a new A-component using a copy of m. P copies O's moves from the new A-component to the main A-component and vice versa. This strategy is innocent, as the initial O-move in the main A-component and the O-questions in the secondary A-component are in the P-view.

- The composition of strategies is given in definition 3.15. The composition of two innocent strategies is innocent.

- We define the terminal object $\mathbf{1}$ as the computational arena with a single question "[" (the initial question) with a single answer "]". For every arena A there is an innocent strategy in $A \Rightarrow \mathbf{1}$, namely the strategy given by $\{[, [\cdot]\}$.

- The categorical product $A \times B$ is given by the product of the computational arenas A and B. The projection function $\pi_A : A \times B \to A$ is given by the following (informal) strategy:

  - O opens with an initial question in A.
  - P copies that question to the A-component of a new $A \times B$-component.
  - P copies every O-question and O-answer in the A-component of $A \times B$ to the main A component.

  π_B is defined similarly.

  To prove that this is a product, suppose there is an object X such that σ is a strategy in the arena $X \to A$ and τ a strategy in $X \to B$, then we can form a strategy in the arena $X \to A \times B$ as follows:

  - O must make an initial move in either A or B.
  - P copies the initial move to a "scratch-pad"-arena in either $X \to A$ or $X \to B$, depending on the component chosen by O using strategy σ and τ respectively.
  - P copies X-questions posed by σ or τ to the normal X arena, and O-answers in that arena to the scratch-pad.
  - Once σ or τ answers the initial question, the game is finished.

- The exponential B^A is given by the function space arena $A \Rightarrow B$.

  To show that this is an exponential, for every strategy in an arena $A \times B \to C$ there must be a corresponding strategy in the arena $A \to B \Rightarrow C$ and vice versa. However, by using the definitions of the product and function space arena, it turns out $A \times B \Rightarrow C$ and $A \Rightarrow B \Rightarrow C$ are identical:

$$
\begin{aligned}
\mathrm{Qn}(A \times B \Rightarrow C) &= \mathrm{Qn}(A \times B) \times M_C + \mathrm{Qn}(C) \\
&= (\mathrm{Qn}(A) + \mathrm{Qn}(B)) \times M_C + \mathrm{Qn}(C) \\
&= \mathrm{Qn}(A) \times M_C + \mathrm{Qn}(B) \times M_C + \mathrm{Qn}(C) \\
\mathrm{Qn}(A \Rightarrow (B \Rightarrow C)) &= \mathrm{Qn}(A) \times M_{B \Rightarrow C} + \mathrm{Qn}(B \Rightarrow C) \\
&= \mathrm{Qn}(A) \times M_C + \mathrm{Qn}(B) \times M_C + \mathrm{Qn}(C) \qquad (\text{Using } M_{B \Rightarrow C} = M_C.) \\
\mathrm{Ans}(A \times B \Rightarrow C) &= (\mathrm{Ans}(A \times B) \times M_C) + \mathrm{Ans}(C) \\
&= ((\mathrm{Ans}(A) + \mathrm{Ans}(B)) \times M_C) + \mathrm{Ans}(C) \\
&= \mathrm{Ans}(A) \times M_C + \mathrm{Ans}(B) \times M_C + \mathrm{Ans}(C) \\
\mathrm{Ans}(A \Rightarrow (B \Rightarrow C)) &= (\mathrm{Ans}(A) \times M_{B \Rightarrow C}) + \mathrm{Ans}(B \Rightarrow C) \\
&= \mathrm{Ans}(A) \times M_C + \mathrm{Ans}(B) \times M_C + \mathrm{Ans}(C) \\
\mathrm{qn}(A \times B \Rightarrow C) &= (\mathrm{qn}(A \times B) \times \mathrm{Id}_M) + \mathrm{qn}(C) \\
&= (\mathrm{qn}(A) + \mathrm{qn}(B)) \times \mathrm{Id}_M + \mathrm{qn}(C) \\
&= \mathrm{qn}(A) \times \mathrm{Id}_M + \mathrm{qn}(B) \times \mathrm{Id}_M + \mathrm{qn}(C) \\
\mathrm{qn}(A \Rightarrow (B \Rightarrow C)) &= (\mathrm{qn}(A) \times \mathrm{Id}_M) + \mathrm{qn}(B \Rightarrow C) \\
&= \mathrm{qn}(A) \times \mathrm{Id}_M + \mathrm{qn}(B) \times \mathrm{Id}_M + \mathrm{qn}(C)
\end{aligned}
$$

So the transpose of a strategy is identical to the strategy itself.

Corollary 3.21 *The category of computational arenas using innocent strategies as morphisms is a Cartesian closed category enriched over dI-domains.*

3.5 The model

We now establish a PCF model which models types as arenas and terms as innocent strategies:

$$
\begin{aligned}
[\![o]\!] &= \mathbb{B} \\
[\![\iota]\!] &= \mathbb{N} \\
[\![A \Rightarrow B]\!] &= [\![A]\!] \Rightarrow [\![B]\!]
\end{aligned}
$$

where $\mathbb{B}$ and $\mathbb{N}$ are the computational arenas defined in Section 3.1.

Translation of λ-abstraction and application follow directly from theorem 3.20. As an example of the translation for the other terms, consider $\supset_o$:

$$[\![\supset_o \colon o_1 \to o_2 \to o_3 \to o_4]\!] = \begin{cases} [^{o_4} & \mapsto (^{o_1} \\ [^{o_4} \cdot (^{o_1} \cdot)^{o_1}_{\mathrm{tt}} & \mapsto (^{o_2} \\ [^{o_4} \cdot (^{o_1} \cdot)^{o_1}_{\mathrm{tt}} \cdot (^{o_2} \cdot)^{o_2}_{b} & \mapsto]^{o_4}_{b} \\ [^{o_4} \cdot (^{o_1} \cdot)^{o_1}_{\mathrm{ff}} & \mapsto (^{o_3} \\ [^{o_4} \cdot (^{o_1} \cdot)^{o_1}_{\mathrm{ff}} \cdot (^{o_3} \cdot)^{o_3}_{b} & \mapsto]^{o_4}_{b} \end{cases}$$

The constants $\supset_\iota$, Z, $(+1)$, (-1), tt, ff and $\overline{n}$ have similar straightforward definitions. Ω_α is modeled by the everywhere undefined partial function.

$[\![\mathbf{Y}^\alpha]\!]$ is harder to define. To denote the different components, we will consider it a function with type $(\alpha_1 \Rightarrow \alpha_2) \to \alpha_3$. The strategy works as follows:

1. After the initial O-move in α_3, P opens a new α_2-component (the dual of the α_3-component) using the same initial move.

2. If O opens a new α_1-component with initial move m, then P opens a new α_2-component as dual to the α_1-component with initial move m.

3. If O makes a move in a an existing component, then P copies that move to its dual component.

We must show that this is an innocent strategy that satisfies definition 1.21. After an O move, the possibilities are:

1. The game is [. Because $\ulcorner[\urcorner$ = [, the reply from P will always be innocent.

2. The game is $p \cdot (\cdot r \cdot [$, where "(" explicitly justifies "[". Here, the P-view is $\ulcorner p \urcorner \cdot (\cdot [$ where "(" and "[" are in the same component. Because P only makes moves that are copies of O-moves in the dual component, $\ulcorner p \urcorner$ must end with a move "$[_1$" of which "(" is a copy. O responded to that move with "[", so P copies that move. As it occurs in the P-view, this strategy is innocent.

3. The game is $p \cdot (\cdot r \cdot)$, where "(" explicitly justifies ")", so the P-view is $\ulcorner p \urcorner \cdot (\cdot)$. Here, "(" must also be a copy of a O-move "[", which must be the last move in $\ulcorner p \urcorner$. P copies the answer ")" to "]", which is innocent because "[" occurs in the P-view.

So the strategy is innocent.

3.6 FCF

To prove that the dialogue games model is fully abstract, we will follow the approach from [Hyland and Ong, 2000], by providing a new language extended from PCF, and show that it is fully abstract and equivalent to PCF.

We create the language **P** by adding a **case**-construct:

$$\frac{t_0 : \beta \quad \ldots \quad t_k : \beta \quad s : \iota}{\mathbf{case}_k^\beta\, s[0 \Rightarrow t_0 \mid 1 \Rightarrow t_1 \mid \ldots \mid k \Rightarrow t_k] : \beta}$$

The operational semantics for **case**-constructs is given by:

$$\frac{s \Downarrow j \quad t_j \Downarrow v}{\mathbf{case}\; s[0 \Rightarrow t_0 \mid 1 \Rightarrow t_1 \mid \ldots \mid k \Rightarrow t_k] \Downarrow v} 0 \leq j \leq k$$

In other words, a **case**-construct evaluates a number and it returns the corresponding value from a (finite) list of cases. If no case is specified for the number, then the computation does not terminate.

We introduce the shorthands:

$$\begin{aligned}
\mathbf{case}\; s[a_1 \Rightarrow t_0 \mid \ldots \mid a_k \Rightarrow t_k] &= \mathbf{case}_k^\beta\, s\left[0 \Rightarrow t'_0 \mid \ldots \mid k' \Rightarrow t'_{k'}\right] \\
\text{where } t'_i &= \begin{cases} t_j & \text{if } i = a_j \text{ for some } j \\ \Omega & \text{otherwise} \end{cases} \\
\mathbf{case}\; s[t_0 \mid \ldots \mid t_k] &= \mathbf{case}_k^\beta\, s[0 \Rightarrow t_0 \mid \ldots \mid k \Rightarrow t_k]
\end{aligned}$$

We can give an interpretation of a **case**-construct in the arena $\iota \Rightarrow \cdots \Rightarrow \iota$ as an innocent strategy as follows:

$$\begin{cases} [& \mapsto (_1 \\ [\cdot (_1 \cdot)_i & \mapsto (_{i+1} \\ [\cdot (_1 \cdot)_i \cdot (_{i+1} \cdot)_m & \mapsto]_m \end{cases}$$

Definition 3.22 *We define the* Ω-match ordering *for terms of* ***P*** *as:*

$$s \equiv C[\Omega, \ldots, \Omega] \leq_\Omega t$$

if $t \equiv C[u_1, \ldots, u_n]$ *for some* ***P****-terms* $u_1, \ldots, u_n$*, where* C *is a* ***P****-context.*

We will now define a subset of **P**-terms called finite canonical forms. Informally, finite canonical forms can be Ω, a number constant, or a **case**-construct where the examined value is a finite canonical form, applied to a number of λ-abstractions of finite canonical forms, and all case-alternatives are finite canonical forms. We will assume that ι is the only program type.

Definition 3.23 *For PCF-types* $A_1, \ldots, A_n$ *we define the set of* finite canonical forms *with free variables from* $f_1, \ldots, f_n$*,* $\mathrm{FCF}\,[f_1 : A_1, \ldots, f_n : A_n]$*, as:*

- Ω *and* $\overline{n} \geq 0$ *are in* $\mathrm{FCF}\left[\overrightarrow{f} : \overrightarrow{A}\right]$.
- *For any* $\overrightarrow{f} : \overrightarrow{A} \equiv f_1 : A_1, \ldots, f_n : A_n$, *and:*

$$\begin{array}{ll} A_i \equiv C_1 \Rightarrow \cdots \Rightarrow C_m \Rightarrow \iota & \textit{for each } 1 \leq i \leq n \\ C_j \equiv D_{j1} \Rightarrow \cdots \Rightarrow D_{jp_j} \Rightarrow \iota & \textit{for each } 1 \leq j \leq m \\ r_c \in \mathrm{FCF}\left[\overrightarrow{f} : \overrightarrow{A}\right] & \textit{for each } 0 \leq c \leq k \\ t_j \in \mathrm{FCF}\left[\overrightarrow{f} : \overrightarrow{A}, \overrightarrow{y_j} : \overrightarrow{D_j}\right] & \textit{for each } 1 \leq j \leq m \end{array}$$

$$\textbf{\textit{case}}\ f_i \left(\lambda y_1 \overrightarrow{:} \overrightarrow{D_1}.t_1\right) \ldots \left(\lambda y_m \overrightarrow{:} \overrightarrow{D_m}.t_m\right)[r_0 \mid \ldots \mid r_k] \in \mathrm{FCF}\left[\overrightarrow{f} : \overrightarrow{A}\right]$$

We define a function θ that maps finite canonical forms to compact innocent strategies in the corresponding arena. Let $A = A_1 \Rightarrow \ldots A_n \Rightarrow \iota$ and $s \in \mathrm{FCF}\left[f_1 : A_1, \ldots, f_n : A_N\right]$, then:

$$\theta\left[\lambda\overrightarrow{f} : \overrightarrow{A}.s\right] : \{\text{P-views of } A\} \to \{\text{P-moves of } A\}$$

Given by:

- If $s = \Omega$, then $\theta\left[\lambda\overrightarrow{f}.\Omega\right]$ is undefined everywhere.
- If $s = \overline{n}$, then $\theta\left[\lambda\overrightarrow{f}.\overline{n}\right]$ is the function that maps "$[_A$" to "$]_n$".
- If s is a **case**-construct $\textbf{case}\ f_1 \left(\lambda\overrightarrow{y_1}.t_1\right) \ldots \left(\lambda\overrightarrow{y_m}.t_m\right)[r_0 \mid \ldots \mid r_k]$ where $1 \leq i \leq n$ and:

  $$\begin{array}{l} A_i \equiv C_1 \Rightarrow \cdots \Rightarrow C_m \Rightarrow \iota \\ C_j \equiv D_{j1} \Rightarrow \cdots \Rightarrow D_{jp_j} \Rightarrow \iota \end{array}$$

  with $r_c \in \mathrm{FCF}\left[\overrightarrow{f} : \overrightarrow{A}\right]$ for each $0 \leq c \leq k$ and $t_j \in \mathrm{FCF}\left[\overrightarrow{f} : \overrightarrow{A}, \overrightarrow{y_j} : \overrightarrow{D_j}\right]$ for each $1 \leq j \leq m$, then $\theta\left[\lambda\overrightarrow{f}.s\right]$ is given by:

  - $\theta\left[\lambda\overrightarrow{f}.s\right]$ maps "$[^A$" to the initial question "$(^{A_i}$" of A_i in A.

– For each $1 \leq j \leq m$, if $\theta\left[\lambda\overrightarrow{f}\overrightarrow{y_j}.t_j\right]$ maps $[^{B_j} \cdot p$ to m, then $\theta\left[\lambda\overrightarrow{f}.s\right]$ maps $[^{A} \cdot (^{A_i} \cdot [^{C_j} \cdot p$ to m.

– For each $0 \leq c \leq k$, if $\theta\left[\lambda\overrightarrow{f}.r_c\right]$ maps $[^{A} \cdot p$ to m, then $\theta\left[\lambda\overrightarrow{f}.s\right]$ maps $[^{A} \cdot (^{A_i} \cdot)_c^{A_i} \cdot p$ to m.

Theorem 3.24 *For any $s, s' \in \mathrm{FCF}[f_1 : A_1, \ldots, f_n : A_n]$:*

- $\theta\left[\lambda\overrightarrow{f}.s\right]$ *is a compact innocent function of the arena* $A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \iota$.
- $s \leq_\Omega s'$ *if and only if* $\theta\left[\lambda\overrightarrow{f}.s\right] \subseteq \theta\left[\lambda\overrightarrow{f}.s'\right]$.

We now define the reverse map: from finite innocent strategies to finite canonical forms.

Let σ be a compact innocent strategy and $A = A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \iota$ a PCF-type. $s_\sigma \in \mathrm{FCF}[f_1 : A_1, \ldots, f_n : A_n]$ is given by:

- If σ does not make a move, then $s_\sigma = \lambda\overrightarrow{f}.\Omega$.
- If σ immediately returns a number n, then $s_\sigma = \lambda\overrightarrow{f}.\overline{n}$.
- Or σ makes a move in one of the A_i components: $[^{\iota} \cdot (^{A_i}$. Suppose $A_i \equiv C_1 \Rightarrow \ldots C_n \Rightarrow \iota$. Either O immediately gives an answer in A_i, or it opens a new C_j-component with an initial question $[^{C_j}$.

  Suppose the latter. The P-view is now $[^{\iota} \cdot (^{A_i} \cdot [^{C_j}$, suppose:

$$C_j \equiv D_{j1} \Rightarrow \ldots D_{jp_j}$$

  Until the answer to $[^{C_j}$ is given, P can only ask questions from:

$$(^{A_1}, \ldots, (^{A_n}, (^{D_{j1}}, \ldots, (^{D_{jp_j}}$$

  so we can derive from σ a strategy σ_j in the arena:

$$D_{j1} \Rightarrow \cdots \Rightarrow D_{jp_j} \Rightarrow A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \iota$$

  This strategy will be strictly smaller than σ, so we can apply induction to find a term:

$$\lambda y_{j1} : D_{j1}. \ldots . \lambda y_{jp_j} : D_{jp_j} . \lambda f_1 : A_1 . \ldots \lambda f_n : A_n . t_j \left(\overrightarrow{y}_j, \overrightarrow{f} \right)$$

that corresponds to σ_j. Until (^{A_i} is answered, σ_j determines every move. There can be only finitely many natural numbers that are possible answers for the question (^{A_i} (as the strategy is compact), say $c_1, \ldots, c_k$. After the move ")", the P-view becomes $[^{\iota} \cdot (^{A_i} \cdot)$. For each value $c_1, \ldots, c_k$, we get a new, smaller strategy $\rho_1, \ldots, \rho_k$, to which we can apply the induction hypothesis to find terms:

$$\begin{gathered} \lambda f_1 : A_1 . \ldots . \lambda f_n : A_n . u_1 \left(\overrightarrow{f} \right) \\ \vdots \\ \lambda f_1 : A_1 . \ldots . \lambda f_n : A_n . u_k \left(\overrightarrow{f} \right) \end{gathered}$$

that represent $\rho_1, \ldots, \rho_k$. Now we can define the interpretation of σ as:

$$\lambda \overrightarrow{f} : \overrightarrow{A} . \, \mathbf{case} \; f_1 \; (\lambda \overrightarrow{y_1} . t_1) \; \ldots \; (\lambda \overrightarrow{y_m} . t_m) \left[c_1 \Rightarrow u_1 \left(\overrightarrow{f} \right) \mid \ldots \mid c_k \Rightarrow u_k \left(\overrightarrow{f} \right) \right]$$

Theorem 3.25 (Strong definability) *There maps in both directions for every PCF-type $A = A_1 \Rightarrow \ldots \Rightarrow A_n \Rightarrow \iota$:*

$$\mathrm{FCF} \left[f_1 : A_1, \ldots, f_n : A_n \right] \rightleftarrows \{ \textit{ compact innocent strategies of } A \}$$

$$\begin{aligned} f_1 : A_1, \ldots, f_n : A_n \; &\vdash s \mapsto \theta \left[\lambda \vec{f} . s \right] \\ f_1 : A_1, \ldots, f_n : A_n \; &\vdash s_\sigma \leftarrow\!\shortmid \sigma \end{aligned}$$

This forms an isomorphism between finite canonical forms and innocent strategies.

Theorem 3.26 (Strong adequacy) *For any **P**-program s and for any value v, $s \Downarrow v$ if and only if $[\![s]\!] \Downarrow v$ (in the category $\mathbb{CA}$).*

Theorem 3.27 *For any PCF terms $s : A$ and $t : A$, $s \sqsubseteq t$ in PCF if and only if $s \sqsubseteq_{\approx} t$ in **P**.*

Proof The $\Rightarrow$ implication is immediate, as any PCF-term is also a **P**-term.

To prove the other direction, define a translation of terms from **P** to PCF as follows:

$$\begin{aligned}
\overline{s\ t} &= \overline{s}\ \overline{t} \\
\overline{\lambda x : A.s} &= \lambda x : A.\overline{s} \\
\overline{\mathbf{Y}(s)} &= \mathbf{Y}(\overline{s}) \\
\overline{\mathbf{case}\ s\ [t_0 \mid \ldots \mid t_k]} &= \supset_\iota\ (\mathrm{eq}\ \overline{s}\ 0)\ \overline{t_0}\ (\supset_\iota\ (\mathrm{eq}\ \overline{s}\ 1)\ \overline{t_1}\ \ldots (\supset_\iota\ (\mathrm{eq}\ \overline{s}\ k)\ \overline{t_k}\ \Omega)\ldots)
\end{aligned}$$

where eq is a term that satisfies eq $u\ v \Downarrow \mathrm{tt}$ if and only if $u \Downarrow n$ and $v \Downarrow n$ for some n, and eq $u\ v \Downarrow \mathrm{tt}$ if and only if $u \Downarrow n$ and $v \Downarrow m$ with $n \neq m$. The interpretation for all other terms is straightforward.

Now suppose $C[X]$ is a context of **P** such that both $C\left[\lambda\vec{f}.s\right]$ and $C[t]$ are programs. Suppose $C[\lambda\vec{f}.s] \Downarrow v$, then $\overline{C\left[\lambda\vec{f}.s\right]}$ is $\overline{C}\left[\lambda\vec{f}.s\right]$. This means $\overline{C}\left[\lambda\vec{f}.s\right] \Downarrow v$ in PCF. As $s \sqsubseteq\!\!\!\sim\ t$, $\overline{C}[t] \Downarrow v$, so $C[t] \Downarrow v$ in **P**. □

We can use the following theorem, in [Hyland and Ong, 2000] attributed to Plotkin and Milner, to show that the model is fully abstract:

Theorem 3.28 (Plotkin and Milner) *Any continuous, order-extensional model of PCF which follows the standard interpretation is a system of Scott-domains. Further, such a model is fully abstract if and only if all compact elements of the model are PCF-definable.*

The standard interpretation means ground types are interpreted as the Scott-domains $\mathbb{N}_\perp$ and $\mathbb{B}_\perp$.

From theorem 3.25 it follows that all compact innocent strategies are **P**-definable. From theorem 3.27 it follows that **P**-definable implies PCF-definable. So this model is fully abstract.

Appendix A

Proof of the Church-Rosser theorem

We want to show that $\twoheadrightarrow_\beta$ has the *diamond property*: a relation R has the diamond property if for all a, b and c such that aRb and aRc there is a d such that bRd and cRd. See also Figure A.1. To do this, we first define a new relation $\twoheadrightarrow_1$, prove that it has the diamond property and show that the $\twoheadrightarrow_\beta$ is the transitive closure of $\twoheadrightarrow_1$.

But first:

Lemma A.1 *For any relation R with the diamond property, its transitive closure $\bar{R}$ will also have the diamond property.*

Proof We are given x, y and z such that $x\bar{R}y$ and $x\bar{R}z$. As $\bar{R}$ is the transitive closure of R, there must be a chain of elements such that $xRy_1R \ldots Ry_nRy$ and a chain such that $xRz_1R \ldots Rz_mRz$. By the diamond property of R, there must be an element s_1 such that y_1Rs_1 and z_1Rs_1. In the same way, there must be an element t_1 such that s_1Rt_1 and y_2Rt_1 and an element t_2 such that s_1Rt_2 and z_2Rt_2. We can continue this process until we have added every element in the chains $xRy_1R \ldots Ry_nRy$ and $xRz_1R \ldots Rz_mRz$, see Figure A.2. So r must exist and yRr and zRr. □

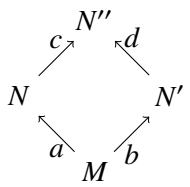


Figure A.1: The Diamond Property for binary relations: if the relation holds at a and b, then the object N'' must exist such that the relation holds at c and d.

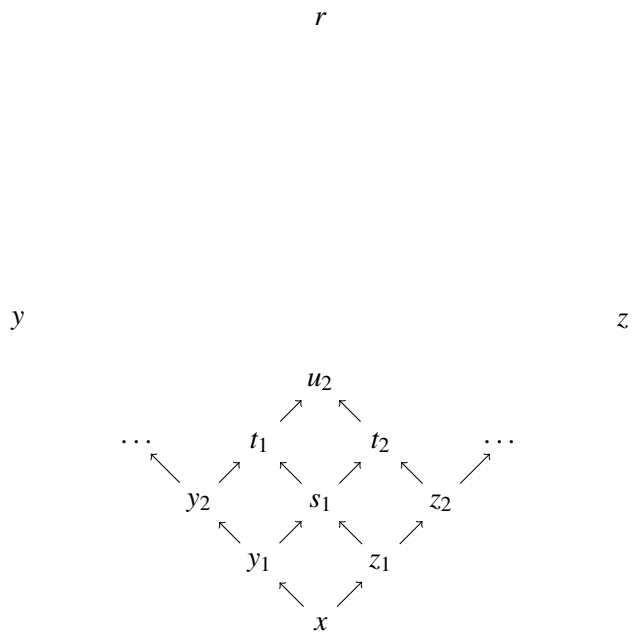


Figure A.2: Extending the diamond property.

We will define $\twoheadrightarrow_1$ as:

$$\frac{}{M \twoheadrightarrow_1 M} \tag{A.1}$$

$$\frac{M \twoheadrightarrow_1 M'}{\lambda x.M \twoheadrightarrow_1 \lambda x.M'} \tag{A.2}$$

$$\frac{M \twoheadrightarrow_1 M' \qquad N \twoheadrightarrow_1 N'}{MN \twoheadrightarrow_1 M'N'} \tag{A.3}$$

$$\frac{M \twoheadrightarrow_1 M' \qquad N \twoheadrightarrow_1 N'}{(\lambda x.M)N \twoheadrightarrow_1 M'[x ::= N']} \tag{A.4}$$

Note that Equation (A.4) overlaps with Equation (A.3) and Equation (A.1) overlaps with all other cases.

Informally, $\rightarrow_\beta$ can choose a single redex to reduce in a single step, but $\twoheadrightarrow_1$ can reduce any number of redexes in every step. After a step, new redexes may have been introduced, so the result of $\twoheadrightarrow_1$ is not necessarily in normal form.

Lemma A.2 $M \twoheadrightarrow_1 M' \Rightarrow M \twoheadrightarrow_\beta M'$.

Proof We prove this by induction on the definitions of $\twoheadrightarrow_1$.

Equation (A.1): Trivial.

Equation (A.2): Suppose $\lambda x.M \twoheadrightarrow_1 \lambda x.M'$, which was introduced by $M \twoheadrightarrow_1 M'$. Using the induction hypothesis $M \twoheadrightarrow_\beta M'$, which implies $\lambda x.M \twoheadrightarrow_\beta M'$.

Equation (A.3): Suppose $MN \twoheadrightarrow_1 M'N'$ was introduced by $M \twoheadrightarrow_1 M'$ and $N \twoheadrightarrow_1 N'$. Using the induction hypothesis, $M \twoheadrightarrow_\beta M'$ and $N \twoheadrightarrow_\beta N'$. Then also $MN \twoheadrightarrow_\beta M'N'$.

Equation (A.4): Suppose $(\lambda x.M)N \twoheadrightarrow_1 M'[x := N']$ was introduced by $M \twoheadrightarrow_1 M'$ and $N \twoheadrightarrow_1 N'$. The induction hypothesis states $M \twoheadrightarrow_\beta M'$ and $N \twoheadrightarrow_\beta N'$. This implies $(\lambda x.M)N \twoheadrightarrow_\beta (\lambda x.M')N'$, using the definition of $\rightarrow_\beta$: $(\lambda x.M')N' \rightarrow_\beta M'[x := N']$, so $(\lambda x.M)N \twoheadrightarrow_\beta M'[x := N']$. □

Lemma A.3 $M \rightarrow_\beta M' \Rightarrow M \rightarrow_1 M'$.

Proof This follows from the fact that $\rightarrow_1$ can choose any number of redexes to reduce, while $\rightarrow_\beta$ can only reduce one.

Lemma A.4 $\twoheadrightarrow_1$ *has the diamond property.*

Proof We prove by induction on $M \twoheadrightarrow_1 M_1$ that for all $M \twoheadrightarrow_1 M_2$ there is an M_3 such that $M_1 \twoheadrightarrow_1 M_3$ and $M_2 \twoheadrightarrow_1 M_3$.

Equation (A.1) This implies $M = M_1$, so take $M_3 = M_2$.

Equation (A.2) This means $M = \lambda x.P$ and $M_1 = \lambda x.P'$ and $M \twoheadrightarrow_1 M_1$ was deduced from $P \twoheadrightarrow_1 P'$. This means that $M_2 = \lambda x.P''$ (as the outer λ-abstraction could not have disappeared), by using the induction hypothesis we can find a P''' such that $P' \twoheadrightarrow_1 P'''$ and $P'' \twoheadrightarrow_1 P'''$, so we take $M_3 = \lambda x.P'''$.

Equation (A.3) This means $M = PQ$ and $M_1 = P'Q'$ and $M \twoheadrightarrow_1 M_1$ was deduced from $P \twoheadrightarrow_1 P'$ and $Q \twoheadrightarrow_1 Q'$. Here we must consider two different cases:

- $M_2 = P''Q''$. Using the induction hypothesis we can find P''' and Q''' such that $P' \twoheadrightarrow_1 P'''$, $P'' \twoheadrightarrow_1$, $Q' \twoheadrightarrow_1 Q'''$ and $Q'' \twoheadrightarrow_1 Q'''$. This means we can take $M_3 = P'''Q'''$.
- If $P = \lambda x.P_1$ for some P_1 then it is also possible that $M_2 = P_1''[x := Q'']$ with $P_1 \twoheadrightarrow_1 P_1''$ and $Q \twoheadrightarrow_1 Q''$. It is easy to check that this must mean $P' = \lambda x.P_1'$ for some P_1'. This means we can find P_1''' such that $P_1' \twoheadrightarrow_1 P_1'''$ and $P_1'' \twoheadrightarrow_1 P_1'''$ and Q''' such that $Q' \twoheadrightarrow_1 Q'''$ and $Q'' \twoheadrightarrow_1 Q'''$ and we take $M_3 = P_1'''[x := Q''']$.

Equation (A.4) This means $M = (\lambda x.P)Q$ and $M_1 = P'[x := Q']$. We gain have to consider two possibilities for M_2:

- $M_2 = P''[x := Q'']$. Using the induction hypothesis we can find P''' such that $P' \twoheadrightarrow_1 P'''$ and $P'' \twoheadrightarrow_1 P'''$, and Q''' such that $Q' \twoheadrightarrow_1 Q'''$ and $Q'' \twoheadrightarrow_1 Q'''$ and let $M_3 = P'''[x := Q''']$.

– $M_2 = (\lambda x.P'')Q''$. By the induction hypothesis we can find P''' such that $P' \twoheadrightarrow_1 P'''$ and $P'' \twoheadrightarrow_1 P'''$, and Q''' such that $Q' \twoheadrightarrow_1 Q'''$ and $Q'' \twoheadrightarrow_1 Q'''$. Now let $M_3 = P'''[x := Q''']$.

□

Corollary A.5 *The transitive closure of* $\twoheadrightarrow_1 \cup \equiv_\alpha \cup \equiv_\eta$ *is* $\twoheadrightarrow_\beta$.

Proof This follows from combining lemmas A.2 and A.3.

Corollary A.6 $\twoheadrightarrow_\beta$ *has the diamond property.*

Proof Using lemmas A.1 and A.4 and corollary A.5.

Theorem A.7 (The Church-Rosser Theorem) *For any* λ*-terms* M*,* N *and* N' *such that* N *and* N' *are in normal form, if* $M \twoheadrightarrow_\beta N$ *and* $M \twoheadrightarrow_\beta N'$*, then* $N \equiv_\alpha N'$.

Proof If $M \twoheadrightarrow_\beta N$ and $M \twoheadrightarrow_\beta N'$, then using corollary A.6 there must be a N'' such that $N \twoheadrightarrow_\beta N''$ and $N' \twoheadrightarrow_\beta N''$. But N and N' are in normal form, so $N'' \equiv_\alpha N' \equiv_\alpha N$. □

Appendix B

Proof of the weak normalization of the simply typed λ-calculus

The simply typed λ-calculus is strongly normalizing: any sequence of β-reductions will eventually reach a normal form. Here we will proof a slightly weaker result, namely that simply typed λ-calculus is *weakly normalizing*: for any λ-term, there is a series of β-reductions that will reach a normal form. This proof is taken from [Barendregt et al., 2013].

Definition B.1 *We define the* depth *of a type as the number arrows in the type:*

$$dpt(\alpha \to \beta) = 1 + dpt(\alpha) + dpt(\beta)$$
$$dpt(\sigma) = 0, \text{ where } \sigma \text{ is a ground type}$$

The depth of a λ-abstraction is the depth of its type and the depth of a redex is the depth of the λ-abstraction.

When a redex is reduced, then the result may contain some new redexes:

1. $(\lambda(x : \alpha \to \beta).x\ P)\ (\lambda(y : \alpha).Q) \to_\beta (\lambda(y : \alpha).Q)\ P$

   By the definition of depth, $\text{dpt}(\alpha) < \text{dpt}(\beta)$, so the new redex has a strictly lower depth.

2. $(\lambda(x : \alpha).(\lambda(y : \beta).P))\ Q\ R \to_\beta (\lambda(y : \beta).P[x := Q])\ R$

   The type of $(\lambda(x : \alpha).(\lambda(y : \beta).P))$ must be of the form $\alpha \to \beta \to \gamma$, which means $(\lambda(y : \beta).P[x := Q])$ has type $\beta \to \gamma$, which has a strictly lower depth.

3. $(\lambda(x : \alpha \to \beta).x)\ (\lambda(y : \alpha).P)\ Q \to_\beta (\lambda(y : \alpha).P)\ Q$

   The type of $(\lambda(x : \alpha \to \beta).x)$ is $(\alpha \to \beta) \to \alpha \to \beta$ and the type of $(\lambda(y : \alpha).P)$ is $\alpha \to \beta$, so the depth of the new redex is strictly lower.

These are the only ways new redexes can be created. It is also possible that the redexes in the argument are duplicated, when the abstracted variable occurs multiple times.

Let d be the highest depth of all redexes in a λ-term. Our reduction strategy will reduce the rightmost redex with depth d (by looking at the term as a string of characters). Its argument is to the right of it, so it can only contain redexes of depth $< d$, so all duplicated redexes are of lower depth. As argued before, the newly created redexes after this reduction must also be of lower depth. This means that the number of redexes with depth d must have gone down by one, while introducing some new redexes of strictly lower depth. This process will terminate, as in every step the number of redexes of maximal depth will go down. A redex of maximal depth can never be created. When no redexes remain, then the term has reached a normal form. □

Bibliography

[Abramsky and Jung, 1994] Abramsky, S. and Jung, A. (1994). Domain theory. In *Handbook of Logic in Computer Science*, pages 1–168. Clarendon Press. 18

[Amadio and Curien, 1998] Amadio, R. and Curien, P. (1998). *Domains and Lambda-Calculi*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press. 13

[Barendregt, 1984] Barendregt, H. P. (1984). *The Lambda Calculus Its Syntax and Semantics*, volume 103. North Holland, revised edition. 3, 6

[Barendregt et al., 2013] Barendregt, H. P., Dekkers, W., and Statman, R. (2013). *Lambda Calculus with Types*. Cambridge University Press, New York, NY, USA. 7, 9, 66

[Berry, 1978] Berry, G. (1978). Stable models of typed lambda-calculi. In *Proceedings of the Fifth Colloquium on Automata, Languages and Programming*, pages 72–89, London, UK, UK. Springer-Verlag. 21, 43

[Church and Rosser, 1936] Church, A. and Rosser, J. B. (1936). Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482. 6

[Huet, 1986] Huet, G. (1986). Formal structures for computation and deduction. 43

[Hyland and Ong, 2000] Hyland, J. and Ong, C.-H. (2000). On Full Abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285 – 408. 45, 50, 56, 61

[Lambek, 1980] Lambek, J. (1980). From Lambda Calculus to Cartesian Closed Categories. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 375–402. Academic Press. 13

[Ong, 1995] Ong, C. H. L. (1995). Correspondence between operational and denotational semantics. In Abramsky, S., Gabbay, D., and Maibaum, T. S. E., editors, *Handbook of Logic in Computer Science, Vol 4*, pages 269–356. Oxford University Press. 37

[Plotkin, 1977] Plotkin, G. (1977). LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223 – 255. 37, 42, 43

[Scott, 1993] Scott, D. S. (1993). A Type-theoretical Alternative to ISWIM, CUCH, OWHY. *Theor. Comput. Sci.*, 121(1-2):411–440. 34

[van Oosten, 1997] van Oosten, J. (1997). A combinatory algebra for sequential functionals of finite type. Technical report, University of Utrecht. 21

[van Oosten, 2002] van Oosten, J. (2002). *Basic Category Theory*. 7

[Zhang, 1991] Zhang, G.-Q. (1991). *Logic of Domains*. Birkhäuser Boston Inc., Cambridge, MA, USA. 22, 23

Symbols

(A P-question. 43

() Zero tuple 15

) An O-answer. 43

; Composition of two strategies 50

$\underset{\sim}{\sqsubset}$ Observationally approximates 35

$\sqsubseteq$ A partial order 16

$\cong$ Observationally equivalent 35

$\equiv_\alpha$ The α-equivalence relation 4

$\equiv_{\beta\eta\alpha}$ The $\beta\eta\alpha$-equivalence relation 6

$\twoheadrightarrow$ The transitive closure of a relation 5

$\twoheadrightarrow_1$ The $\twoheadrightarrow_1$-reduction relation 60

$\rightarrow_\eta$ The η-reduction relation 5

$\supset_\iota$ Conditional, returning a number 33

$\supset_o$ Conditional, returning a boolean 33

[An O-question. 43

$[\![.]\!]$ Interpretation function 27

$\Uparrow$ Does not terminate 35

] A P-answer. 43

$\uparrow$ Two elements are bounded 16, 21–23, 25, 26

$\restriction$ Projection of moves 46, 48

$\vdash$ An equality judgment 7

$\bot$ The bottom element from a poset 16

ff False term 33

$\bigsqcap$ Greatest lower bound 16

ι The type of natural numbers 33

λ The transpose of a function 12

$\bigsqcup$ Least upper bound 16

Ω Undefined term 33

$\circ$ Morphism composition 9, 33

$\ulcorner . \urcorner$ The P-view 45

θ Function mapping FCFs to compact innocent strategies 55

tt True term 33

$\Downarrow$ Terminates (to value) 35

Y Fixed point-combinator 33

Z Test for zero 33