6
The Language PCF

In chapter 4, we have provided semantics for both typed and untyped λ-calculus. In this chapter we extend the approach to typed λ-calculus with fixpoints (λY-calculus), we suggest formal ways of reasoning with fixpoints, and we introduce a core functional language called PCF [Sco93, Plo77]. PCF has served as a basis for a large body of theoretical work in denotational semantics. We prove the adequacy of the interpretation with respect to the operational semantics, and we discuss the full abstraction problem, which has triggered a lot of research, both in syntax and semantics.

In section 6.1, we introduce the notion of *cpo-enriched* CCC's, which serves to interpret the λY-calculus. In section 6.2, we introduce fixpoint induction and show an application of this reasoning principle. In section 6.3, we introduce the language PCF, define its standard denotational semantics and its operational semantics, and we show a computational adequacy property: the meaning of a closed term of basic type is different from $\perp$ if and only if its evaluation terminates. In section 6.4, we address a tighter correspondence between denotational and operational semantics, known as the full abstraction property. In section 6.5, we introduce Vuillemin's sequential functions, which capture first order PCF definability. In section 6.6, we show how a fully abstract model of PCF can be obtained by means of a suitable quotient of an (infinite) term model of PCF.

6.1 The λY-calculus

The λY-calculus is the typed λ-calculus extended with a family of constants Y^σ of type $(\sigma \to \sigma) \to \sigma$ for each type σ (Y for short), with the following reduction rule:

$$(Y) \quad YM \to M(YM)\,.$$

It is also convenient to introduce a special constant Ω^σ at each type (to be interpreted by $\perp$).

Definition 6.1.1 (cpo-enriched-CCC) *A cartesian closed category* $\mathbf{C}$ *is called a cpo-enriched cartesian closed category if all its homsets are cpo's, if composition is continuous, if pairing and currying are monotonic, and if the following strictness conditions hold (for all f of the appropriate type):*

$$\perp \circ f = \perp \qquad ev \circ \langle \perp, f\rangle = \perp\,.$$

Remark 6.1.2 (1) *Notice that our definition of a cpo-enriched CCC involves the cartesian closed structure of the category: thus in our terminology a cpo-enriched CCC is not just a cpo-enriched category (see definition 7.1.5) that happens to be cartesian closed.*

(2) *The strictness conditions of definition 6.1.1 will be used in the proof of theorem 6.3.6.*

Lemma 6.1.3 *In a cpo-enriched CCC pairing and currying are continuous.*

PROOF. We consider the case of currying only (the argument is similar for pairing, which has to be established first). In order to prove $\Lambda(\bigvee \Delta) = \bigvee\{\Lambda(f) \mid f \in \Delta\}$, it is enough to check that $\bigvee\{\Lambda(f) \mid f \in \Delta\}$ satisfies the characterizing equation:

$$ev \circ (\bigvee\{\Lambda(f) \mid f \in \Delta\} \times id) = \bigvee \Delta\,.$$

The monotonicity of Λ guarantees that $\{\Lambda(f) \mid f \in \Delta\}$ is directed. Hence by continuity of composition and pairing we have:

$$ev \circ (\bigvee\{\Lambda(f) \mid f \in \Delta\} \times id) = \bigvee\{ev \circ (\Lambda(f) \times id) \mid f \in \Delta\} = \bigvee \Delta\,. \qquad \Box$$

The following definition was first given by Berry [Ber79].

Definition 6.1.4 (least fixpoint model) *A least fixpoint model is a cpo-enriched cartesian closed category where Ω and Y are interpreted as follows:*

$$[\![\Omega]\!] = \perp \qquad [\![Y]\!] = \bigvee_{n<\omega} [\![\lambda f.f^n\Omega]\!]\,,$$

where $M^n\Omega = M(\cdots(M\Omega)\cdots)$, n times.

The fact that the sequence of the $[\![\lambda f.f^n\Omega]\!]$'s is increasing follows from the assumptions of monotonicity in the definition of cpo-enriched CCC.

Proposition 6.1.5 *In a least fixpoint model, the (Y)-rule is valid.*

PROOF. Exploiting the continuity of the composition and pairing, we have:

$$\begin{array}{lllll} [\![YM]\!] & = & ev \circ \langle \bigvee_{n<\omega}[\![\lambda f.f^n\Omega]\!], [\![M]\!]\rangle & = & \bigvee_{n<\omega}[\![M^n\Omega]\!] \\ [\![M(YM)]\!] & = & ev \circ \langle [\![M]\!], \bigvee_{n<\omega}[\![M^n\Omega]\!]\rangle & = & \bigvee_{n<\omega}[\![M^{n+1}\Omega]\!]\,. \end{array} \qquad \Box$$

Proposition 6.1.6 **Cpo** *is a cpo-enriched CCC. In particular, for any cpo D, $Fix : (D \to D) \to D$, defined by $Fix(f) = \bigvee_{n\in\omega} f^n(\perp)$, is continuous.*

PROOF. Notice that in a category with enough points, the interpretation of Y given in definition 6.1.4 is precisely Fix. $\Box$

In chapter 2 we have seen so-called fixpoint combinators (cf. example 2.1.11 and exercise 2.1.16). Thus, the λY-calculus can be simulated in the untyped λ-calculus. But the semantics of λY-calculus is easier, since it involves only fixpoints of functions, not domain equations (cf. chapter 3).

Exercise 6.1.7 *Consider the extension of the simply typed λ-calculus with a collection of constants Y_n $(n \geq 0)$ and rules:*

$$(Y_n) \quad Y_{n+1}M \to M(Y_nM) .$$

Prove that the system obtained by adding these rules to the β-rule is strongly normalizing. Hint: Adapt the proof of theorem 2.2.9.

Exercise 6.1.8 *Let* **C** *be a cpo-enriched cartesian-closed category such that currying is strict, i.e., $\Lambda(\bot) = \bot$. Adapt the definition of Böhm tree given in chapter 2 to the λY-calculus by setting $\omega(\lambda\vec{x}.YM_1 \ldots M_p) = \Omega$ $(p \geq 1)$. Show that the following holds:*

$$[\![M]\!] = \bigvee\{[\![\omega(N)]\!] \mid M \to^* N\} .$$

Hints: Extend the meaning function by setting: $[\![Y_n]\!] = [\![\lambda f.f^n\Omega]\!]$. Show that $[\![M]\!] = \bigvee_{n<\omega}[\![M_n]\!]$, where M_n is obtained from M by replacing all its occurrences of Y by Y_n. Consider the normal form N_0 of M_n. Show that it is the result of replacing all the occurrences of Y by Y_0 in a reduct N of M, and use the strictness assumptions to show $[\![N_0]\!] = [\![\omega(N)]\!]$.

Exercise 6.1.9 *A class of continuous functionals $F_D : (D \to D) \to D$, ranging over all cpo's D, is called a fixpoint operator if $F_D(f)$ is a fixpoint of f, for any D and $f : D \to D$. Moreover, it is called uniform if the following holds:*

$$\forall f : D \to D, g : E \to E, h : D \to E \quad (h \circ f = g \circ h \Rightarrow h(F_D(f)) = F_D(g)) ,$$

where h is supposed strict. Show that Fix is the unique uniform fixpoint operator.

The following exercise, based on [HP90], shows that fixpoints cannot coexist with coproducts in a cartesian closed category.

Exercise 6.1.10 *We say that a category* **C** *with a terminal object 1*

- *has fixpoints if for every A and $f : A \to A$ there exists a morphism $Y(f) : 1 \to A$ such that $f \circ Y(f) = Y(f)$;*
- *is inconsistent if all objects in* **C** *are isomorphic to 1.*

(1) Show that any cartesian closed category that has an initial object 0 and fixpoints is inconsistent. Hints: Consider $Y(id_0)$, and use exercise A2.7.4. (2) Show that any cartesian closed category that has binary coproducts and fixpoints is inconsistent. Hints: Show that the two injections, call them tt and ff, from 1 to $2 = 1 + 1$ are equal, by 'implementing' in the categorical language the following reasoning (where $\neg = [ff, tt] : 2 \to 2$):

$$\begin{array}{rclcl} tt & = & (Y(\neg) \text{ or } \neg(Y(\neg))) & = & (Y(\neg) \text{ or } Y(\neg)) & = & Y(\neg) \\ & = & (Y(\neg) \text{ and } Y(\neg)) & = & (Y(\neg) \text{ and } \neg(Y(\neg))) & = & ff . \end{array}$$

(In [HP90], incompatibility with equalizers and with a natural number object is also proved.)

6.2 Fixpoint induction

A key motivation for denotational semantics lies in its applications to the proof of properties of programs. An important tool is fixpoint induction. If we want to show that a property $\mathcal{P}$ holds of a term YM, then, knowing that the meaning of YM is the lub of the sequence $\bot, F(\bot), F(F(\bot)), \ldots$, where F is the meaning of M, it is enough to check the following properties.

- The (meaning of) property $\mathcal{P}$, is an ω-subdcpo of the domain D associated to the type of YM, i.e., $\mathcal{P}$ is closed under lub's of non-decreasing chains; such predicates are called *inclusive*.
- Both properties $\bot \in \mathcal{P}$ and $\forall x(x \in \mathcal{P} \Rightarrow F(x) \in \mathcal{P})$ hold.

This is summarized by the following inference rule, known as the fixpoint induction principle

$$\frac{\mathcal{P} \textit{ inclusive} \quad \bot \in \mathcal{P} \quad \forall x \ (x \in \mathcal{P} \Rightarrow F(x) \in \mathcal{P})}{\mathit{Fix}(F) \in \mathcal{P}} ,$$

where $\mathcal{P} \subseteq D$, for a given cpo D, and $F : D \to D$ is continuous. Such an inference rule is a step towards mechanizing proofs of programs. What is needed next is a formal theory for proving that some predicates are inclusive (see exercise 6.2.2).

Remark 6.2.1 *The sufficiency of the above conditions, hence the validity of fixpoint induction, follows immediately from the Peano induction principle on natural numbers. Thus, mathematically speaking, it is not strictly necessary to formulate the above principle explicitly. One can prove $\bot \in \mathcal{P}$, $F(\bot) \in \mathcal{P}$, $F(F(\bot)) \in \mathcal{P}, \ldots$ and use Peano induction to conclude (if $\mathcal{P}$ is inclusive). The interest of stating an explicit induction principle is to enable one: (1) to write lighter proofs, as $F(x)$ is easier to write than $F(F(\ldots(\bot)\ldots))$; and (2) to insert it in a mechanical proof-checker like LCF [Pau87].*

Exercise 6.2.2 *(1) Let D be a cpo. Show that $\emptyset$ and D are inclusive predicates in D. Show that $x = x$ and $x \leq y$ are inclusive in $D \times D$. (2) Let D and E be cpo's and $f : D \to E$ be continuous. Let $\mathcal{R}$ be inclusive in E. Show that $f^{-1}(\mathcal{R})$ is inclusive. (3) Let D be a cpo and $\mathcal{P}, \mathcal{Q}$ be inclusive in D. Then show that $\mathcal{P} \cap \mathcal{Q}$ and $\mathcal{P} \cup \mathcal{Q}$ are inclusive. (4) Let D and E be cpo's and $\mathcal{R}$ be inclusive on $D \times E$ in its first argument. Show that the predicate $\forall y \ (x\mathcal{R}y)$ is inclusive on D. (5) Let D and E be dcpo's and $\mathcal{P}, \mathcal{Q}$ be inclusive in D, E respectively. Show that $\mathcal{P} \times \mathcal{Q}$ is inclusive in $D \times E$, and that $\mathcal{P} \to \mathcal{Q}$ is inclusive in $D \to E$, where $\mathcal{P} \to \mathcal{Q} = \{f : D \to E \mid \forall d \in \mathcal{P} \ f(d) \in \mathcal{Q}\}$.*

As an illustration, we carry out in some detail the proof of the following proposition, due to Bekič, which shows that n-ary fixpoints can be computed using unary fixpoints.

Proposition 6.2.3 *Let D, E be cpo's and $f : D \times E \to D$, $g : D \times E \to E$ be continuous. Let (x_0, y_0) be the least fixpoint of $\langle f, g\rangle$. Let x_1 be the least fixpoint of $f \circ \langle id, h\rangle$, where $h = \mathit{Fix} \circ \Lambda(g) : D \to E$ (hence $h(x_1)$ is such that $g(x_1, h(x_1)) = h(x_1)$). Then $x_0 = x_1$ and $y_0 = h(x_1)$.*

PROOF. $(x_0, y_0) \leq (x_1, h(x_1))$. Define the predicate $\mathcal{Q}(u,v)$ as $(u,v) \leq (x_1, h(x_1))$. This is an inclusive predicate (see exercise 6.2.2). Thus we may start the fixpoint induction engine. The base case is obvious. Suppose that $(u,v) \leq (x_1, h(x_1))$. We want to show that $f(u,v) \leq x_1$ and $g(u,v) \leq h(x_1)$. By monotonicity we have $f(u,v) \leq f(x_1, h(x_1))$ and $g(u,v) \leq g(x_1, h(x_1))$. But $f(x_1, h(x_1)) = x_1$ since x_1 is a fixpoint of $f \circ \langle id, h\rangle$. This settles the inequality $f(u,v) \leq x_1$. By definition of h, we have $h(x_1) = g(x_1, h(x_1))$, which settles the other inequality.

$(x_1, h(x_1)) \leq (x_0, y_0)$. We define a second predicate $\mathcal{R}(u)$ as $(u, h(u)) \leq (x_0, y_0)$. We leave the base case aside for the moment, and suppose that $\mathcal{R}(u)$ holds. We have to prove $\mathcal{R}(f(u, h(u)))$. We have $f(u, h(u)) \leq f(x_0, y_0) = x_0$ by monotonicity, and by definition of (x_0, y_0). We need a little more work to obtain $h(f(u, h(u))) \leq y_0$. It is enough to check $h(x_0) \leq y_0$. We define a third inclusive predicate $\mathcal{S}(u)$ as $u \leq y_0$, remembering that $h(x_0)$ is the least fixpoint of $\Lambda(g)(x_0)$. The base case is obvious. Suppose that $u \leq y_0$. Then $g(x_0, u) \leq g(x_0, y_0) = y_0$. Hence fixpoint induction with respect to $\mathcal{S}$ allows us to conclude $h(x_0) \leq y_0$. We are left with the base case with respect to $\mathcal{R}$: $(\perp, h(\perp)) \leq (x_0, y_0)$ follows *a fortiori* from $h(x_0) \leq y_0$. □

In this proof: we have focused in turn on each of the least fixpoint operators involved in the statement, exploiting just the fact that the other least fixpoints are fixpoints.

6.3 The programming language PCF

Scott [Sco93], and then Plotkin [Plo77], introduced a simply typed λY-calculus, PCF, which has become a quite popular language in studies of semantics. It has two basic types: the type ι of natural numbers, and the type o of booleans. Its set of constants is given in figure 6.1. The language PCF is interpreted in **Cpo** as specified in figure 6.2 (for the interpretation of Ω and Y, cf. definition 6.1.4). We use the same notation for the constants and for their interpretation. This interpretation is called the *continuous model* of PCF. More generally, we define the following notion of standard model.

Definition 6.3.1 (standard model) *Let* $\mathbf{C}$ *be a least fixpoint model. If we interpret* ι *and* o *by objects* D^ι *and* D^o *such that* $\mathbf{C}[1, D^\iota]$ *and* $\mathbf{C}[1, D^o]$ *are (order-isomorphic) to* $\omega_\perp$ *and* $\mathbf{B}_\perp$, *if the basic constants are interpreted as in figure 6.2, and if the first order constants behave functionally as specified in figure 6.2 (replacing, say,* $succ(x)$ *by* $ev \circ \langle succ, x\rangle$*), then we say that we have a* standard model of PCF.

Recall that if $\mathbf{C}$ has enough points (cf. definition 4.5.4), then the model is called extensional.

Definition 6.3.2 (order-extensional) *Let* $\mathbf{C}$, D^ι, *and* D^o *be as in definition 6.3.1. Moreover, suppose that* $\mathbf{C}$ *has enough points and that the order between the morphisms is the pointwise ordering, as in* **Cpo**. *Then the model is called order-extensional.*

n	$: \iota$	$(n \in \omega)$
$tt, f\!f$	$: o$	
$succ, pred$	$: \iota \to \iota$	
$zero?$	$: \iota \to o$	
$if\ \ then\ \ else$	$: o \to \iota \to \iota \to \iota$	
$if\ \ then\ \ else$	$: o \to o \to o \to o$	
Ω	$: \sigma$	for all σ
Y	$: (\sigma \to \sigma) \to \sigma$	for all σ

Figure 6.1: The constants of PCF

$D^o = \mathbf{B}_\perp$	where $\mathbf{B}_\perp = \{\perp, tt, f\!f\}$
$D^\iota = \omega_\perp$	flat domain on natural numbers
$D^{\sigma\to\tau} = D^\sigma \to_{cont} D^\tau$	exponent in **Cpo**

$$succ(x) = \begin{cases} \perp & \text{if } x = \perp \\ x+1 & \text{if } x \neq \perp \end{cases} \qquad pred(x) = \begin{cases} \perp & \text{if } x = \perp \text{ or } x = 0 \\ x-1 & \text{otherwise} \end{cases}$$

$$zero?(x) = \begin{cases} \perp & \text{if } x = \perp \\ tt & \text{if } x = 0 \\ f\!f & \text{otherwise} \end{cases} \qquad if\ x\ then\ y\ else\ z = \begin{cases} \perp & \text{if } x = \perp \\ y & \text{if } x = tt \\ z & \text{if } x = f\!f \end{cases}$$

Figure 6.2: Interpretation of PCF in **Cpo**

Operational semantics of PCF. We equip PCF with an operational semantics which is adequately modelled by any standard model. It is described in figure 6.3 by means of a deterministic evaluation relation $\to_{op}$.

Exercise 6.3.3 *Let* $add_l = Y(\lambda fxy.if\ zero?(x)\ then\ y\ else\ succ(f(pred(x))y))$. *Compute* $add_l\, 4\, 3$ *using the rules in figure 6.3.*

Exercise 6.3.4 *Imitate the techniques of chapter 2 to establish that the rewriting system* $\to$ *specified by the eight axioms of figure 6.2 (applied in any context) is confluent, and that if* $M \to^* N$ *and* N *is a normal form, then* $M \to^*_{op} N$. *Hint: Prove suitable versions of the standardization and Church-Rosser theorems presented in chapter 2.*

Next we investigate the relationships between the denotational and the operational semantics of PCF.

$$\begin{array}{ll}
(\lambda x.M)N \rightarrow_{op} M[N/x] & YM \rightarrow_{op} M(YM) \\
zero?(0) \rightarrow_{op} tt & zero?(n+1) \rightarrow_{op} ff \\
succ(n) \rightarrow_{op} n+1 & pred(n+1) \rightarrow_{op} n \\
\textit{if } tt \textit{ then } N \textit{ else } P \rightarrow_{op} N & \textit{if } ff \textit{ then } N \textit{ else } P \rightarrow_{op} P
\end{array}$$

$$\frac{M \rightarrow_{op} M'}{MN \rightarrow_{op} M'N} \qquad \frac{M \rightarrow_{op} M'}{\textit{if } M \textit{ then } N \textit{ else } P \rightarrow_{op} \textit{if } M' \textit{ then } N \textit{ else } P}$$

$$\frac{M \rightarrow_{op} M'}{f(M) \rightarrow_{op} f(M')} \quad (\text{for } f \in \{succ, pred, zero?\})$$

Figure 6.3: Operational semantics for PCF

Definition 6.3.5 (PCF program) *We call programs the terms of PCF which are closed and of basic type.*

For example, $(\lambda x.x)3$ and $add_l\, 4\, 3$ (cf. exercise 6.3.3) are programs.

Theorem 6.3.6 (adequacy) *Any standard model* $\mathbf{C}$ *of PCF is adequate, i.e., for all programs of type* ι *(and similarly for type* o*):*

$$(\exists n \ P \rightarrow^*_{op} n) \Leftrightarrow [\![P]\!] = n\,.$$

PROOF. $(\Rightarrow)$ This follows by soundness of the continuous model.

$(\Leftarrow)$ We use the notation of section 4.5, and write $\underline{D^\sigma} = \mathbf{C}[1, D^\sigma]$. The induction on types comes into play by a definition of a family of so-called *adequacy relations* $\mathcal{R}^\sigma \subseteq \underline{D^\sigma} \times PCF^o_\sigma$, for each type σ, where PCF^o_σ is the set of closed terms of type σ. Here is the definition of these (logical-like) relations ($\mathcal{R}^o$ is analogous to $\mathcal{R}^\iota$):

$$\begin{array}{lcl}
\mathcal{R}^\iota & = & \{(x, M) \mid x = \bot \text{ or } (x = n \text{ and } M \rightarrow^*_{op} n)\} \\
\mathcal{R}^{\sigma\rightarrow\tau} & = & \{(f, M) \mid \forall e, N \ \ (e\, \mathcal{R}^\sigma N \Rightarrow ev \circ \langle f, e\rangle\, \mathcal{R}^\tau\, MN)\}\,.
\end{array}$$

The statement is a part of the following claim.

Claim. For each provable judgment $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma$, for each n-tuple $(d_1, N_1), \ldots, (d_n, N_n)$ such that $d_i\, \mathcal{R}^{\sigma_i} N_i$ for $i = 1, \ldots, n$, we have:

$$[\![\vec{x} : \vec{\sigma} \vdash M]\!] \circ \langle d_1, \ldots, d_n\rangle\, \mathcal{R}^\sigma\, M[N_1/x_1, \ldots, N_n/x_n]\,.$$

We set $M' = M[N_1/x_1, \ldots, N_n/x_n]$, etc. We proceed with the simplest cases first.

- $M = x_i$. Then $[\![M]\!] \circ \langle d_1, \ldots, d_n\rangle = d_i$, and $M[N_1/x_1, \ldots, N_n/x_n] = N_i$, hence the sought result is $d_i\, \mathcal{R}^{\sigma_i} N_i$, which is among the assumptions.

- $M = NQ$. By induction $[\![N]\!] \circ \langle d_1, \ldots, d_n\rangle\, \mathcal{R}^{\sigma\rightarrow\tau} N'$ and $[\![Q]\!] \circ \langle d_1, \ldots, d_n\rangle\, \mathcal{R}^\sigma Q'$. By definition of $\mathcal{R}^{\sigma\rightarrow\tau}$, $ev \circ \langle [\![N]\!] \circ \langle d_1, \ldots, d_n\rangle, [\![Q]\!] \circ \langle d_1, \ldots, d_n\rangle\rangle\, \mathcal{R}^\tau N'Q'$, i.e., $[\![M]\!] \circ \langle d_1, \ldots, d_n\rangle\, \mathcal{R}^\tau M'$.
- $M = \lambda x.Q$. We have to show, for each $d\, \mathcal{R}^\sigma N$:

$$ev \circ \langle [\![M]\!] \circ \langle d_1, \ldots, d_n\rangle, d\rangle\, \mathcal{R}^\tau\, M'N \ \ \textit{i.e.} \ \ [\![Q]\!] \circ \langle d_1, \ldots, d_n, d\rangle\, \mathcal{R}^\tau\, (\lambda x.Q')N.$$

By induction we have:

$$[\![Q]\!] \circ \langle d_1, \ldots, d_n, d\rangle\, \mathcal{R}^\tau\, Q[N_1/x_1, \ldots, N_n/x_n, N/x]\,.$$

Since $(\lambda x.Q')N \rightarrow_{op} Q[N_1/x_1, \ldots, N_n/x_n, N/x]$, we can conclude provided the following property holds, for all σ:

$$(Q_1) \ \ f\, \mathcal{R}^\sigma M \text{ and } M' \rightarrow_{op} M \ \ \Rightarrow \ \ f\, \mathcal{R}^\sigma M'\,.$$

- $M = n$. In this case, $n\, \mathcal{R}^\iota M$ holds trivially. Similarly for tt and ff.
- $M = succ$. Let $d\, \mathcal{R}^\iota P$. We have to show $ev \circ \langle succ, d\rangle\, \mathcal{R}^\iota\, succ(P)$. There are two cases:

  $d = \bot$. Then $ev \circ \langle succ, d\rangle = ev \circ \langle succ, \bot\rangle = \bot$

  $d = n$. Then $ev \circ \langle succ, d\rangle = n + 1$.

In both cases $ev \circ \langle succ, d\rangle\, \mathcal{R}^\iota\, succ(P)$. The reasoning is similar for *pred*, *zero?*, and *if then else*.

- $M = Y$. We have to show $[\![Y]\!]\, \mathcal{R}^{(\sigma\rightarrow\sigma)\rightarrow\sigma}\, Y$, that is, $ev \circ \langle [\![Y]\!], g\rangle\, \mathcal{R}^\sigma\, YM$, for all $g\, \mathcal{R}^{\sigma\rightarrow\sigma} M$. We assume the following properties (cf. inclusive predicates), for all (fixed) σ, M:

(Q_2) $\ \bot\, \mathcal{R}^\sigma M$

(Q_3) $\ \{f_n\}_{n<\omega}$ non decreasing implies $(\forall n \ f_n\, \mathcal{R}^\sigma M) \Rightarrow (\bigvee_{n<\omega} f_n)\, \mathcal{R}^\sigma M$.

By (Q_3), the conclusion follows if we show:

$$ev \circ \langle [\![\lambda f.f^n\Omega]\!], g\rangle\, \mathcal{R}^\sigma\, YM \quad (\text{for all } n)\,.$$

We set $d_n = ev \circ \langle [\![\lambda f.f^n\Omega]\!], g\rangle$. Since $d_n = [\![f^n\Omega]\!] \circ g$, we have $d_{n+1} = ev \circ \langle g, d_n\rangle$ for all n. Therefore, we only have to show:

(1) $d_0\, \mathcal{R}^\sigma\, YM$. Since $d_0 = [\![\Omega]\!] \circ g$, this follows from (Q_2) and from the left strictness of composition.

(2) $(d\, \mathcal{R}^\sigma\, YM) \Rightarrow (ev \circ \langle g, d\rangle\, \mathcal{R}^\sigma\, YM)$. Since $g\, \mathcal{R}^{\sigma\rightarrow\sigma} M$ by assumption, we have $ev \circ \langle g, d\rangle\, \mathcal{R}^\sigma\, M(YM)$, and the conclusion then follows by (Q_1).

Properties (Q_1) and (Q_2) are obvious at basic types. For a type $\sigma \rightarrow \tau$, (Q_1) follows by induction from the inference: $(M' \rightarrow_{op} M) \Rightarrow (M'N \rightarrow_{op} MN)$ and (Q_2) follows from the strictness equation $ev \circ \langle \bot, d\rangle = \bot$. Property (Q_3) follows at basic types from the fact that non-increasing sequences are stationary in a flat domain, and at functional types from the preservation of limits by continuity. This completes the proof of the claim. □

Remark 6.3.7 *The adequacy relations used in the proof of theorem 6.3.6 combine ideas from the computability technique (cf. theorem 3.5.18, remark 3.5.17, and remark 4.5.2) and from the* inclusive predicates *technique (cf. section 6.2).*

6.4 The full abstraction problem for PCF

In general, given a programming language, the specification of the operational semantics is given in two steps:

(1) Evaluation: a collection of *programs* is defined, usually a collection of closed terms, on which a partial relation of evaluation is defined. The evaluation is intended to describe the dynamic evolution of a program while running on an abstract machine.

(2) Observation: a collection of admissible observations is given. These observations represent the only means to record the behavior of the evaluation of a program.

In this fashion, an observational equivalence can be defined on arbitrary terms M and N as follows: M is observationally equivalent to N if and only if whenever M and N can be plugged into a piece of code P, so to form correct programs $P[M]$ and $P[N]$, then M and N are not separable (or distinguishable) by any legal observation. On the other hand any interpretation of a programming language provides a theory of program equivalence. How does this theory compare to observational equivalence? We will say that an interpretation (or a model) is adequate whenever it provides us with a theory of equivalence which is contained in the observational equivalence. Moreover, we call an adequate model (equationally) fully abstract if the equivalence induced by the model coincides with the observational equivalence.

In section 3.2, we discussed a full abstraction result for D_∞ models. There, all terms were considered programs, and the observation was the existence of a head normal form. In this section, we discuss the situation for PCF. We have defined the programs as the closed terms of basic type. We have defined an evaluation relation $\rightarrow_{op}$. What can be observed of a program is its convergence to a natural number or to a boolean value. The principal reason for focusing on programs is that they lead to observable results. This stands in contrast with expressions like $\lambda x.x$, which are just code, and are not evaluated by $\rightarrow_{op}$ unless they are applied to an argument, or more generally unless they are plugged into a program context. A program context for a PCF term is a context C (cf. definition 2.1.6) such that $C[M]$ is a program.

Definition 6.4.1 (observational preorder) *We define a preorder $\leq_{obs}$, called the observational preorder, between PCF terms M, N of the same type, as follows:*

$$M \leq_{obs} N \quad\Leftrightarrow\quad \forall C\ (C[M] \rightarrow^*_{op} c \ \Rightarrow\ C[N] \rightarrow^*_{op} c)\ ,$$

where C ranges over all the contexts which are program contexts for both M and N, and where $c ::= n \mid tt \mid ff$.

Remark 6.4.2 *By exercise 6.3.4 and by theorem 6.3.6, equivalent definitions for $\leq_{obs}$ are:*

$$M \leq_{obs} N \quad\Leftrightarrow\quad \forall C\ (C[M] \rightarrow^* c \ \Rightarrow\ C[N] \rightarrow^* c)$$
$$M \leq_{obs} N \quad\Leftrightarrow\quad \forall C\ ([\![C[M]]\!] \leq [\![C[N]]\!])\ .$$

Definition 6.4.3 (fully abstract) *A cpo-enriched CCC is said to yield an inequationally fully abstract (fully abstract for short) model of PCF if the following equivalence holds for any PCF terms of the same type:*

$$M \leq_{obs} N \Leftrightarrow [\![M]\!] \leq [\![N]\!]\ .$$

It is a consequence of the adequacy theorem that the direction ($\Leftarrow$) holds for the continuous model (and in fact for any standard model). But the converse direction does not hold for the continuous model. There are several proofs of this negative result, all based on a particular continuous function $por : \mathbf{B}_\perp \times \mathbf{B}_\perp \rightarrow \mathbf{B}_\perp$ defined by:

$$por(x,y) = \begin{cases} tt & \text{if } x = tt \text{ or } y = tt \\ ff & \text{if } x = ff \text{ and } y = ff \\ \perp & \text{otherwise}\ . \end{cases}$$

(1) Plotkin first proved that the continuous model is not fully abstract. He gave the following terms, both of type $(o \rightarrow o \rightarrow o) \rightarrow o$:

$$M_1 = \lambda g.\mathit{if}\ P_1\ \mathit{then}\ \mathit{if}\ P_2\ \mathit{then}\ \mathit{if}\ P_3\ \mathit{then}\ \Omega\ \mathit{else}\ tt\ \mathit{else}\ \Omega\ \mathit{else}\ \Omega$$
$$M_2 = \lambda g.\mathit{if}\ P_1\ \mathit{then}\ \mathit{if}\ P_2\ \mathit{then}\ \mathit{if}\ P_3\ \mathit{then}\ \Omega\ \mathit{else}\ ff\ \mathit{else}\ \Omega\ \mathit{else}\ \Omega\ ,$$

where $P_1 = g\, tt\, \Omega$, $P_2 = g\, \Omega\, tt$, and $P_3 = g\, ff\, ff$. These terms are designed in such a way that:

$$tt = [\![M_1]\!](por) \neq [\![M_2]\!](por) = ff\ .$$

On the other hand $M_1 =_{obs} M_2$. This is proved thanks to two key syntactic results:

(a) Milner's context lemma [Mil77]. This lemma, proposed as exercise 6.4.4, states that in the definition of $\leq_{obs}$ it is enough to let C range over so-called applicative contexts, of the form $[\,]N_1 \dots N_p$. Applying this lemma to M_1, M_2, we only have to consider contexts $[\,]N$. By the definition of $\rightarrow_{op}$, we have for $i = 1, 2$:

$$[M_i]N \rightarrow^*_{op} c \Rightarrow \begin{cases} N\, tt\, \Omega \rightarrow^*_{op} tt \\ N\, \Omega\, tt \rightarrow^*_{op} tt \\ N\, ff\, ff \rightarrow^*_{op} ff\ . \end{cases}$$

(b) There is no N such that $N\, tt\, \Omega \rightarrow^*_{op} tt$, $N\, \Omega\, tt \rightarrow^*_{op} tt$ and $N\, ff\, ff \rightarrow^*_{op} ff$. This result is a consequence of the following more general result. PCF is a *sequential* language, in the following sense: If C is a program context with several holes, if

$$[\![\vdash C[\Omega, \dots, \Omega]]\!] = \perp \quad\text{and}\quad \exists M_1, \dots, M_n\ [\![\vdash C[M_1, \dots, M_n]]\!] \neq \perp,$$

then there exists an i called a sequentiality index, such that

$$\forall N_1, \dots, N_{i-1}, N_{i+1}, \dots, N_n\ \ [\![\vdash C[N_1, \dots, N_{i-1}, \Omega, N_{i+1}, \dots, N_n]]\!] = \perp\ .$$

This result is an easy consequence of (the PCF version of) Berry's syntactic sequentiality theorem 2.4.3 (see exercise 6.4.5) and of the adequacy theorem 6.3.6. Here, it is applied to $C = N[\,][\,]$, observing that we can use $N\, ff\, ff \rightarrow^*_{op} ff$ to deduce that there is no c such that $N\Omega\Omega \rightarrow^*_{op} c$.

Another way to prove the non-existence of N is by means of logical relations. We have treated essentially the same example when we discussed Very Finitary PCF (cf. section 4.5).

(2) Milner has shown that in an extensional standard fully abstract model of PCF, the interpretations of all types are algebraic, and their compact elements must be definable, i.e., the meaning of some closed term. This is called the definability theorem (for a proof, we refer to [Cur86]). One can use this result to cut down the path followed in (1) and go directly to step (b). In reality, there is no cut down at all, since the proof of the definability theorem uses the context lemma, and exploits terms in the style of M_1, M_2.

Exercise* 6.4.4 (context lemma) *Let M and M' be two closed PCF terms of the same type such that, for all closed terms $N_1, \ldots, N_n$ such that $MN_1 \cdots N_n$ is of basic type, the following holds:*

$$MN_1 \cdots N_n \rightarrow_{op}^{*} c \quad \Rightarrow \quad M'N_1 \cdots N_n \rightarrow_{op}^{*} c\,.$$

Show that $M \leq_{obs} M'$, by induction on (length of the reduction $C[M] \rightarrow_{op}^{*} c$, size of $C[M]$).

Exercise 6.4.5 (syntactic sequentiality for PCF) *Prove the PCF version of theorem 2.4.3, and show the corresponding corollary along the lines of exercise 2.4.4.*

The converse of the definability theorem also holds, and is easy to prove.

Proposition 6.4.6 *If $\mathbf{C}$ is an order-extensional standard model of PCF in which all $\mathbf{C}[1, D^\sigma]$'s are algebraic and are such that all their compact elements are definable, then $\mathbf{C}$ is fully abstract.*

PROOF. Suppose that $M \leq_{obs} M'$. It is enough to check $[\![M]\!](\vec{d}) \leq [\![M']\!](\vec{d})$ for all compact $\vec{d} = d_1 \cdots d_n$. Then the conclusion follows using contexts of the form $[\,]N_1 \cdots N_n$. □

Exercise 6.4.7 (uniqueness) *Show, as a consequence of proposition 6.4.6 and of the definability theorem, that all order-extensional standard models of PCF are isomorphic (in a suitable sense).*

In fact, this (unique) fully abstract model exists, and was first constructed by Milner (essentially) as a quotient of the term model of PCF. (see section 6.6, and in particular remark 6.6.11). Since then, lots of efforts have been made in order to provide 'more semantic' constructions of this model (this is known as the full abstraction problem for PCF [BCL85]). In particular, the non-definability of *por* prompted the study of sequentiality, which is the subject of section 6.5 and of chapter 14. A weaker notion, stability, appeared on the way, and is the subject of chapter 12. By 'more semantic' constructions, the contributors in this area certainly meant at least the following two informal requirements:

(1) The model should consist explicitly of cpo's of a certain sort and of functions or function-like morphisms of a certain sort.

(2) The model should throw some light on the operational preorder of PCF. In particular, it was hoped to show that the operational preorder for PCF could be shown to be effectively presented.

As regards the last point, we now know that there is no hope: Loader has proved that the observational preorder on Finitary PCF is undecidable [Loa97]. But not all efforts have been lost: the sequential model of chapter 14 is fully abstract for an extension of PCF with a family of control operators (see section 14.4). The stable and sequential models are also interesting for other reasons. For example, stability is at the root of linear logic (see chapter 13).

Exercise 6.4.8 *Show that the undecidability of the observational equivalence for Finitary PCF implies the undecidability of the definability problem for Finitary PCF (cf. section 4.5).*

Remark 6.4.9 *Gunter has proposed a simple semantic proof of $M_1 =_{obs} M_2$. In the stable model of PCF, to be defined in chapter 12, one retains only functions which satisfy the following property (specified here for a type like $o \times o \rightarrow o$):*

$$\forall x \;\; f(x) \neq \bot \quad \Rightarrow \quad \exists y \text{ minimum s.t. } \;\; (y \leq x \text{ and } f(y) \neq \bot)\,.$$

In particular, por is rejected (take $x = (tt, tt)$, then $(\bot, tt)$ and $(tt, \bot)$ are both minimal, but there is no minimum), and as a consequence we have $[\![M_1]\!] = [\![M_2]\!]$. Now, because the direction ($\Leftarrow$) holds for the stable model, which is standard, we have $M_1 =_{obs} M_2$.

6.5 Towards sequentiality

We have already pointed out that the λ-calculus is sequential (theorem 2.4.3). In sections 4.5 and 6.4, we have exhibited examples of inherently parallel functions that are not definable in a (finitary version of) PCF. In this section, we give further evidence of the sequential nature of PCF. We define sequential functions in a restricted setting, which will be extended in chapter 14. We show that the compact definable first order functions of the continuous model of PCF are exactly the (compact) first order sequential functions.

But before we engage in the semantic investigation of sequentiality, we should mention that another approach to full abstraction is to stick to the continuous model and seek a suitable extension of PCF for which the continuous model is fully abstract. Such a simple extension exists, as was first shown by Plotkin, who added a parallel conditional to PCF [Plo77]. Later it was shown in [Cur86] that the more natural extension of PCF with a constant *por* (with the rules *por* x $tt \rightarrow_{op} tt$, *por* tt $x \rightarrow_{op} tt$ and *por* $f\!f$ $f\!f \rightarrow_{op} f\!f$) has the continuous model as fully abstract model. The proof goes via definability (cf. proposition 6.4.6). Plotkin has also given a strengthened form of definability: if the language is further extended with a *parallel existential* operator, then the language is universal for the continuous model, i.e., every computable element of the model (cf. definition 1.1.17) is definable (see [Plo77] for details, and see remark 14.4.18 for a similar, more satisfactory result in a sequential framework).