Sequentiality and Piece-wise affinity in Segments of Real-PCF

Amin Farjudian [1]

School of Computer Science
University of Birmingham
Birmingham, England
September 18, 2002

Abstract

Real PCF(**RPCF**) was proposed by Martín Escardó [Esc97] as a language for Real number computation. One of the key — and most controversial — constants is *parallel-if* (pif_I), the existence of which causes a serious inefficiency in the language leading to RPCF being impractical. While search is being undertaken to replace pif_I with a more efficient operator, one needs to be assured of the segment of RPCF without pif_I being sequential. A **positive** answer to this question is the main result of this paper. On the other hand, we show that non-affine functions — such as $f(x) \coloneqq x^2$ — are not definable in RPCF without pif_I.

[1] Email: `M.A.Farjudian@cs.bham.ac.uk`

Contents

1 A Reminder of the Definitions

1.1 PCF

In one of his seminal works [Plo77], Plotkin introduced **PCF** — **P**rogramming language for **C**omputable **F**unctions. Here we give a description of the language, taken almost entirely from Plotkin's original paper [Plo77], slightly modified to match our framework.

The set $\mathbb{T}$ of *types* of PCF is generated by the following grammar:

$$\sigma ::= \mathsf{bool} \mid \mathsf{nat} \mid \sigma \to \sigma$$

bool and nat are the ground types of *truth values* and *natural numbers*, respectively.

For each type σ we assume the existence of denumerably many variables $x_i^\sigma (i \geq 0)$. C_0, the set of *standard* constants of PCF, consists of the following:

$$\begin{aligned}
\mathsf{true} &: \mathsf{bool} \\
\mathsf{false} &: \mathsf{bool} \\
\mathsf{if}_{\mathsf{bool}} &: \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \\
\mathsf{if}_{\mathsf{nat}} &: \mathsf{bool} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \\
\mathsf{Y}_\sigma &: ((\sigma \to \sigma) \to \sigma) \text{ (one for each } \sigma)
\end{aligned}$$

To perform arithmetic computations, we also add the following constants to C_0, and call the new set of constants C_A:

$$\begin{aligned}
\overline{n} &: \mathsf{nat} \text{ (one for each natural number } n) \\
\mathsf{succ} &: (\mathsf{nat} \to \mathsf{nat}) \\
\mathsf{pred} &: (\mathsf{nat} \to \mathsf{nat}) \\
\mathsf{Zero} &: (\mathsf{nat} \to \mathsf{bool})
\end{aligned}$$

The set of *terms* of PCF is the least set $\mathcal{T}$ containing the following:

(1) Every variable x_i^σ is a term of type σ.
(2) Every constant $c \in C_\mathrm{A}$ of type σ is a term of type σ.
(3) If M and N are terms of types $(\sigma \to \tau)$ and σ, respectively, then (MN) is a term of type τ.
(4) If M is a term of type τ then $(\lambda x_i^\sigma . M)$ is a term of type $(\sigma \to \tau)$.

As the above rules impose an inductive structure on the set of terms $\mathcal{T}$, we can define functions over $\mathcal{T}$ using recursion. For instance, take:

$$Var := \{x_i^\sigma \mid i \geq 0 \text{ and } \sigma \in \mathbb{T}\}$$

to be the set of PCF variables, and:

$$\mathcal{P}_{fin}(Var) := \{A \subseteq Var \mid A \text{ is finite}\}$$

then the function $\mathrm{FV} : \mathcal{T} \to \mathcal{P}_{fin}(Var)$ which returns the set of *free variables* of any term $M \in \mathcal{T}$ can be defined by:

$$\begin{aligned}
\mathrm{FV}(x_i^\sigma) &= \{x_i^\sigma\} \ \ (x_i^\sigma \in Var) \\
\mathrm{FV}(c) &= \emptyset \ \ (c \in C_{\mathrm{A}}) \\
\mathrm{FV}((MN)) &= \mathrm{FV}(M) \cup \mathrm{FV}(N) \\
\mathrm{FV}((\lambda x_i^\sigma.M)) &= \mathrm{FV}(M) \setminus \{x_i^\sigma\}
\end{aligned}$$

where $\setminus$ is the relative complement symbol:

Notation 1.1 (relative complement (set difference) : $A \setminus B$) *For any two sets A and B we denote the **relative complement of B in** A by $A \setminus B$, i.e.*

$$A \setminus B := \{x \in A \mid x \notin B\}$$

A term $M \in \mathcal{T}$ is said to be *closed* if $\mathrm{FV}(M) = \emptyset$ and *open* otherwise.

Terms of the form (MN) are called *applications*[2] and sometimes the brackets are dropped, when they are understood as associating to the left. Terms of the form $(\lambda x_i^\sigma.M)$ are called *abstractions*.

$M[N_\sigma/x_i^\sigma]$ is the result of *substituting* the term N_σ (of type σ) for all free occurrences of x_i^σ in M, making appropriate changes in the bound variables of M so that no free variables of N become bound.

Programs are closed terms of ground type. The idea is that the ground types are the datatypes, and programs produce data via *operational semantics*.

1.1.1 Operational Semantics of PCF

We first define an *immediate reduction relation* $\to$ between terms:

[2] Plotkin called them *combinations*.

Definition 1.2 [Immediate Reduction Relation $\rightarrow$]

$$(i) \quad \begin{cases} \mathsf{if}_\sigma \ \mathsf{true}\ M\ N \ \rightarrow \ M \\ \mathsf{if}_\sigma \ \mathsf{false}\ M\ N \ \rightarrow \ N \end{cases} \quad (\sigma \text{ ground}).$$

$$(ii) \quad \mathsf{succ}\ \overline{m} \rightarrow \overline{m+1}\ (m \geq 0)$$

$$(iii) \quad \mathsf{pred}\ \overline{m+1} \rightarrow \overline{m}\ (m \geq 0)$$

$$(iv) \quad \begin{cases} \mathsf{Zero}\ \overline{0} \rightarrow \ \mathsf{true} \\ \mathsf{Zero}\ \overline{m+1} \rightarrow \ \mathsf{false}\ (m \geq 0) \end{cases}$$

$$(v) \quad \mathsf{Y}_\sigma M \ \rightarrow \ M(\mathsf{Y}_\sigma M)$$

$$(vi) \quad ((\lambda x_i^\sigma.M)N) \ \rightarrow \ M[N/x_i^\sigma]$$

$$(vii) \quad \frac{M \rightarrow M'}{(MN) \rightarrow (M'N)}$$

$$(viii) \quad \frac{N \rightarrow N'}{(MN) \rightarrow (MN')} \qquad (M \in \{\mathsf{if}, \mathsf{succ}, \mathsf{pred}, \mathsf{Zero}\})$$

Let $\stackrel{\star}{\rightarrow}$ denote the *reflexive and transitive* closure of $\rightarrow$. Then we can define the operational semantics by a partial function Eval which gives constants from programs:

Definition 1.3 [operational semantics for PCF : Eval]

$$\mathrm{Eval}(M) = c \text{ iff } M \stackrel{\star}{\rightarrow} c, \text{ for any program } M \text{ and constant } c$$

A closer look at the rules for $\rightarrow$ reveals that for each term there is *at most* one immediate reduction rule applicable. In particular $\rightarrow$ is a *partial function* which is undefined on constants. This implies that Eval is *well-defined*, i.e. $M \stackrel{\star}{\rightarrow} c$ and $M \stackrel{\star}{\rightarrow} c'$ implies that c and c' are identical.

1.1.2 Denotational Semantics of PCF

We will use some mathematical structures called *cpo's* in our treatment of the denotational semantics. Let us briefly go through some definitions and facts. For a more comprehensive account of cpo's, see [AJ94].

Let D be a set and $\sqsubseteq \subseteq D \times D$ a binary relation over D. $(D, \sqsubseteq)$ is called a *partial order* if it satisfies the following:

- *reflexivity* $\quad \forall x \in D\colon\ x \sqsubseteq x$
- *anti symmetry* $\quad \forall x, y \in D\colon\ x \sqsubseteq y$ and $y \sqsubseteq x \Rightarrow x = y$
- *transitivity* $\quad \forall x, y, z \in D\colon\ x \sqsubseteq y \sqsubseteq z \Rightarrow x \sqsubseteq z$

Where there is no confusion, we simply write D instead of $(D, \sqsubseteq)$.

Notation 1.4 *Throughout this paper $a \sqsubset b$ means a is **strictly** less than b, i.e:*

$$a \sqsubset b \Leftrightarrow a \sqsubseteq b \text{ and } a \neq b$$

Definition 1.5 [bounded (consistent) subsets] Let $(D, \sqsubseteq)$ be a partial order. Then $B \subseteq D$ is **bounded** (or **consistent**) if it has an upper bound, i.e.:

$$\exists d \in D\colon \forall b \in B\colon b \sqsubseteq d$$

We use the abbreviation $a \uparrow b$ for "$\{a, b\}$ is bounded".

Definition 1.6 [bounded complete] A partial order $(D, \sqsubseteq)$ is said to be **bounded complete** if any of its bounded subsets has a *least* upper bound in D.

For $X \subseteq D$ we write $\sqcup X$ for the **least upper bound (l.u.b.)** of X, and $\sqcap X$ for the **greatest lower bound (g.l.b.)** of X provided they exist. In case X has only two elements, we use the infix notation, i.e. we write $a \sqcup b$ for $\sqcup\{a, b\}$ and $a \sqcap b$ for $\sqcap\{a, b\}$.

A subset X of D is said to be directed if it is non-empty and every pair of its elements have an upper bound in X itself, i.e:

$$\forall x, y \in X\colon \exists z \in X\colon x \sqsubseteq z \ \wedge \ y \sqsubseteq z$$

We often write $X \subseteq_{dir} D$ to abbreviate 'X is a directed subset of D'.

Definition 1.7 [cpo] A partial order $(D, \sqsubseteq)$ is said to be a **complete partial order** (cpo for short) if it satisfies the following:

1. D has a least element $\bot_D$ under $\sqsubseteq$.
2. Every directed subset X of D has a least upper bound which we denote by $\sqcup_D X$ or just $\sqcup X$ where there is no confusion.

Definition 1.8 [finite element] Let $(D, \sqsubseteq)$ be a cpo and $a, b \in D$. a is said to be **way-below** b (or a **approximates** b) — written as $a \ll b$ — if the following is true:

$$\forall X \subseteq_{dir} D\ :\ b \sqsubseteq \sqcup_D X \Rightarrow \exists x \in X\colon a \sqsubseteq x$$

An element $d \in D$ is called **finite** (or **compact**) if $d \ll d$.

Definition 1.9 [basis] Let $(D, \sqsubseteq)$ be a cpo. $B \subseteq D$ is a **basis** for D if:

$$\forall x \in D\colon (\{y \in B \mid y \ll x\} \ \subseteq_{dir} D) \text{ and } (\ x = \sqcup\{y \in B \mid y \ll x\}\ )$$

A cpo $(D, \sqsubseteq)$ is called **continuous** if D is a basis for itself. Throughout this paper, by **domain** we mean continuous cpo. [3]

[3] Note that different people use different definitions for the concept of a domain.

Definition 1.10 [algebraic cpo] A cpo $(D, \sqsubseteq)$ is called **algebraic** if the collection of its finite elements forms a basis. In that case, we denote the basis by $K(D)$.

Definition 1.11 [ω-continuous, ω-algebraic cpo] A cpo is called **ω-continuous** if it has a countable basis. If $(D, \sqsubseteq)$ is algebraic and $K(D)$ is countable, then $(D, \sqsubseteq)$ is said to be **ω-algebraic**.

Definition 1.12 [continuous function between cpo's] A function $f\colon (D, \sqsubseteq_D) \to (E, \sqsubseteq_E)$ is **continuous** if:

1. f is monotone:

$$\forall x, y \in D\colon\ x \sqsubseteq_D y \Rightarrow f(x) \sqsubseteq_E f(y)$$

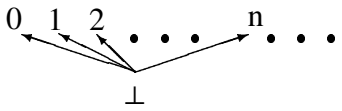
2. f preserves the suprema of directed sets:

$$\forall X \subseteq_{dir} D\colon\ f(\sqcup_D X) = \sqcup_E f(X)$$

The collection of all the continuous functions from D to E under the induced pointwise ordering forms a cpo which is often written as $[D \to E]$.

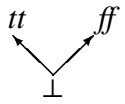
Of special interest are the two so-called *flat* cpo's of truth values and natural numbers, $\mathbb{B}_\perp$ and $\mathbb{N}_\perp$ respectively, defined as follows:

Example 1.13 (i) Let $\mathbb{N}_\perp = \{\perp\} \cup \{0, 1, 2, ...\}$, partially ordered as follows:
$x \sqsubseteq y \Leftrightarrow (x = y \text{ or } x = \perp)$



$\mathbb{N}_\perp$ is a cpo which is often called the **flat domain of natural numbers**.

(ii) Let $\mathbb{B}_\perp = \{\perp, tt, ff\}$ partially ordered as follows:
$x \sqsubseteq y \Leftrightarrow (x = y \text{ or } x = \perp)$



$\mathbb{B}_\perp$ is also a cpo which is often called the **flat Boolean domain**.

By a *standard collection of domains for PCF* we mean a family $\{D^\sigma\}$ of cpo's, one for each PCF-type $\sigma \in \mathbb{T}$ such that:

- $D^{\mathsf{bool}} = \mathbb{B}_\perp$
- $D^{\mathsf{nat}} = \mathbb{N}_\perp$
- $D^{\sigma\to\tau} = [D^\sigma \to D^\tau]$

The aim is to take $\{D^\sigma\}$ — the standard collection of domains for PCF — as a model and interpret the terms of PCF inside that model. Let us proceed step by step in order to make things clear. First we show how the constants are going to be interpreted via the function $\mathcal{A}$ which is type-respecting, i.e:

$$\forall c^\sigma \in C_{\mathrm{A}} \colon \mathcal{A}(c) \in D^\sigma$$

Definition 1.14 $\mathcal{A}\colon C_{\mathrm{A}} \to \cup\{D^\sigma\}$ is defined by:

$$\begin{aligned}
\mathcal{A}(\mathsf{true}) &= tt \\
\mathcal{A}(\mathsf{false}) &= ff \\
\mathcal{A}(\overline{n}) &= n \quad (n \geq 0) \\
\mathcal{A}(\mathsf{if}_\sigma)(p)(x)(y) &= \begin{cases} x & (\text{if } p = tt) \\ y & (\text{if } p = ff) \\ \perp & (\text{if } p = \perp) \end{cases} \quad (p \in \mathbb{B}_\perp,\ x, y \in D^\sigma \text{ and } \sigma \text{ ground}) \\
\mathcal{A}(\mathsf{succ})(x) &= \begin{cases} x+1 & (x \geq 0) \\ \perp & (x = \perp) \end{cases} \quad (x \in \mathbb{N}_\perp) \\
\mathcal{A}(\mathsf{pred})(x) &= \begin{cases} x-1 & (x \geq 1) \\ \perp & (x \in \{\perp, 0\}) \end{cases} \quad (x \in \mathbb{N}_\perp) \\
\mathcal{A}(\mathsf{Zero})(x) &= \begin{cases} tt & (x = 0) \\ ff & (x > 0) \\ \perp & (x = \perp) \end{cases} \quad (x \in \mathbb{N}_\perp) \\
\mathcal{A}(\mathsf{Y}_\sigma)(f) &= \sqcup\{f^n(\perp) \mid n \geq 0\} \qquad (f \in D^{\sigma\to\sigma})
\end{aligned}$$

Terms are interpreted with respect to *environments*. An environment is simply a type-respecting function from the set of variables to the model $\cup\{D^\sigma\}$. We let Env be the set of all the environments, ranged over by ρ. Hence for any $\rho \in \mathrm{Env}$:

$$\rho\colon Var \to \cup\{D^\sigma\}$$
$$\rho(x_i^\sigma) \in D^\sigma$$

For any $\rho \in \mathrm{Env}$, $x_i^\sigma \in Var$, $d \in D^\sigma$ we let $\rho_{x_i^\sigma \mapsto d}$ denote the environment ρ' such

that:

$$\rho'(x) = \begin{cases} d & (x = x_i^\sigma) \\ \rho(x) & (x \neq x_i^\sigma) \end{cases}$$

Now we have all the necessary material to define the *denotational semantics* $[\![.]\!] : \mathcal{T} \to (\text{Env} \to \cup\{D^\sigma\})$ by:

Definition 1.15 [denotational semantics of PCF]

$$\begin{aligned} [\![x_i^\sigma]\!](\rho) &= \rho(x_i^\sigma) \qquad (x_i^\sigma \in Var) \\ [\![c]\!](\rho) &= \mathcal{A}(c) \qquad (c \in \mathcal{C}_{\mathrm{A}}) \\ [\![(MN)]\!](\rho) &= [\![M]\!](\rho)([\![N]\!](\rho)) \\ [\![(\lambda x_i^\sigma.M)]\!](\rho)(d) &= [\![M]\!](\rho_{x_i^\sigma \mapsto d}) \qquad (d \in D^\sigma) \end{aligned}$$

1.1.3 Matching Operational and Denotational Semantics: the necessity of parallel operators

Having both operational and denotational semantics side-by-side gives us a handy tool for studying properties of objects in one area by analysing the corresponding objects in the other. One such case — indeed an important one — is proving properties of programs via their corresponding object in the model. For instance, if M and N are programs written in PCF, one can prove their "equivalence" via their interpretation in the model. This in turn necessitates the two semantics to match up to a certain degree. This is an important subject with a rich literature available. Here we discuss this issue as far as needed for our own purposes. For a thorough treatment together with the proofs and details, again see [Plo77] from which we will take much of our material, unless stated otherwise.

Perhaps the following theorem ([Plo77]) is a good place to start with. Here, $\hat{\perp}$ denotes the environment which maps every variable to the bottom element of the corresponding cpo, i.e.

$$\forall x_i^\sigma \in Var : \hat{\perp}(x_i^\sigma) = \perp_{D^\sigma}$$

Theorem 1.16 *For any PCF-program $M : \sigma$ and constant $c : \sigma$:*

$$\text{Eval}(M) = c \Leftrightarrow [\![M]\!](\hat{\perp}) = \mathcal{A}(c)$$

To prove ($\Rightarrow$) direction, one needs to observe the so-called soundness of the operational semantics with respect to the denotational one, i.e:

$$\text{if } M \to N \text{ then } [\![M]\!](\rho) = [\![N]\!](\rho)$$

For ($\Leftarrow$) direction, see [Plo77].

This theorem demonstrates how the *behaviour* of a program regarding its *termination* and the constant it evaluates to is reflected in the denotational semantics. Now let us investigate the *equivalence* of programs. For that matter we need the following definition:

Definition 1.17 The set $\mathscr{C}$ of contexts of PCF with numbered holes — ranged over by C — is generated by the grammar:

$$C ::= [\,]_j \mid x_i^\sigma \mid c \mid CC \mid \lambda x_i^\sigma.C$$

where $j \in \mathbb{N}$, $x_i^\sigma \in Var$ and $c \in C_{\mathrm{A}}$.

If all the occurrences of the holes bear the same subscript in a term, we denote them [ ] for short.

In other words, contexts are just terms with holes in them. We usually write a context C as $C[.,\ldots,.]$ not to get confused with ordinary terms. These holes can be filled with terms of appropriate type to give a term. We denote a context $C[.,\ldots,.]$ filled with the terms $M_1,\ldots,M_k$ by $C[M_1,\ldots,M_k]$. Below we define the operation of *filling the holes of a context*. We abbreviate a vector of the form $[N_1,\ldots,N_n]$ simply as $[\overrightarrow{N}]$:

$$\begin{aligned}
[\,]_j[N_1,\ldots,N_n] &= N_j \\
x_i^\sigma[\overrightarrow{N}] &= x_i^\sigma \\
c[\overrightarrow{N}] &= c \\
(C_1C_2)[\overrightarrow{N}] &= C_1[\overrightarrow{N}]\; C_2[\overrightarrow{N}] \\
(\lambda x_i^\sigma.C)[\overrightarrow{N}] &= \lambda x_i^\sigma.(C[\overrightarrow{N}])
\end{aligned}$$

As our main objects of interest are programs, we regard two terms M and N as *operationally equivalent* — written as $M \cong N$ — if they can be substituted for each other in any program without affecting its behaviour:

Definition 1.18 [operational equivalence] $M \cong N$ if and only if for any context $C[.,\ldots,.]$ such that $C[M,\ldots,M]$ and $C[N,\ldots,N]$ are programs either both of Eval$(C[M,\ldots,M])$ and Eval$(C[N,\ldots,N])$ are undefined or else both are defined and equal.

It is easy to check that $\cong$ is an equivalence relation.

One of the reasons we define a denotational semantics for a language is to be able to resort to it as an easier alternative to the (usually) tedious operational semantics when it comes to proving the equivalence of programs, provided the equivalence is reflected in the denotational semantics. Unfortunately this is not the case with PCF and its model as we have defined it.

Take the two terms M_0 and M_1 defined as: [4]

$$M_i = \lambda x.\mathsf{if}_{\mathsf{nat}}(x\ \mathsf{true}\ \Omega_{\mathsf{bool}})\{\mathsf{if}_{\mathsf{nat}}(x\ \Omega_{\mathsf{bool}}\ \mathsf{true})\ [\mathsf{if}_{\mathsf{nat}}(x\ \mathsf{false}\ \mathsf{false})\ \Omega_{\mathsf{nat}}\ \bar{i}\ ]\ [\Omega_{\mathsf{nat}}]\}\ \{\Omega_{\mathsf{nat}}\}$$

where $i \in \{0,1\}$. Here x is of type $\mathsf{bool} \to (\mathsf{bool} \to \mathsf{bool})$, Ω_{bool} and Ω_{nat} are

[4] taken from [Plo77].

non-terminating terms of types bool and nat respectively. They could be:

$$\Omega_{\mathsf{bool}} = \mathsf{Y}_{\mathsf{bool}}(\lambda x^{\mathsf{bool}}.x)$$
$$\Omega_{\mathsf{nat}} = \mathsf{Y}_{\mathsf{nat}}(\lambda x^{\mathsf{nat}}.x)$$

It needs quite some effort to grasp what the M_i's do in the first place, and then the proof that they are operationally equivalent is a bit involved (see [Plo77]). Anyway the important fact for us is that:

$$M_0 \cong M_1$$

On the other hand:

$$[\![M_0]\!](\hat{\perp}) \neq [\![M_1]\!](\hat{\perp}) \tag{1}$$

The best way to verify the above inequality is to find and argument in the model over which $[\![M_i]\!](\hat{\perp}), (i \in \{0,1\})$ disagree.

Definition 1.19 The function $\widehat{por}\colon \mathbb{B}_\perp \to (\mathbb{B}_\perp \to \mathbb{B}_\perp)$ is defined by:

$$\widehat{por}\ x\ y := \begin{cases} tt & (x = tt \ \vee \ y = tt) \\ ff & (x = y = ff) \\ \perp & \text{otherwise} \end{cases}$$

It is not difficult to see that $\widehat{por} \in D^{\mathsf{bool}\to(\mathsf{bool}\to\mathsf{bool})}$, and to verify that:

$$[\![M_0]\!](\hat{\perp})(\widehat{por}) = \overline{0}$$
$$[\![M_1]\!](\hat{\perp})(\widehat{por}) = \overline{1}$$

hence the inequality (1).

Observing this mismatch, we say that the model is not *fully abstract* for the language. Full abstraction is an important criterion regarding the relation between a language and its model, and in our case in order to achieve full abstraction there could be two alternative ways ahead: either to purge the model from troublesome objects like $\widehat{por}$ or otherwise enrich the language. Both possibilities have been pursued but here we follow Plotkin's direction of enriching the language [Plo77] in order to get to the parallelism issues.

Let us first add constants:

$$\mathsf{pif}_o\colon \mathsf{bool} \to o \to o \to o \qquad (o \in \{\mathsf{bool}, \mathsf{nat}\})$$

to the set C_{A} to get the extended set of constants $C_{\mathrm{A}}{}^{+}$ and call the extended language based on that PCF^{+}. Then in order to get an operational semantics for PCF^{+} we extend the relation $\to$ of Definition 1.2 (see page 5) by the following rules for pif_o, $(o \in \{\mathsf{bool}, \mathsf{nat}\})$, and denote the new immediate reduction relation by $\to_{+}$:

$$(ix)\ \begin{cases} \mathsf{pif}_o\ P\ M\ M \to_+ M \\ \mathsf{pif}_o\ \mathsf{true}\ M\ N \to_+ M \\ \mathsf{pif}_o\ \mathsf{false}\ M\ N \to_+ N \end{cases}$$

$$(x)\ \begin{cases} \dfrac{P \to_+ P'}{\mathsf{pif}_o\ P \to_+ \mathsf{pif}_o\ P'} \\[2ex] \dfrac{M \to_+ M'}{\mathsf{pif}_o\ P\ M \to_+ \mathsf{pif}_o\ P\ M'} \\[2ex] \dfrac{N \to_+ N'}{\mathsf{pif}_o\ P\ M\ N \to_+ \mathsf{pif}_o\ P\ M\ N'} \end{cases}$$

Let us mention some notes about the constants pif_o, ($o \in \{\mathsf{bool}, \mathsf{nat}\}$). As you might have guessed, the prefix p stands for *parallel*, hence we read pif as "parallel-if". A closer look at the rules (ix) and (x) reveals the parallelism as pif_o looks at its three arguments at the same time. Rule (x) consists of three rows, with the first one being the only one having a counterpart for if, see rule (*viii*), Definition 1.2, page 5.

The extension of $\xrightarrow{\star}$ to $\xrightarrow{\star}_+$ and Eval to Eval$^+$ is straightforward and left to the reader. Although $\to_+$ is *non-deterministic*, still it can be proved that $\xrightarrow{\star}_+$ has the so-called *Church-Rosser* property, i.e.

$$\begin{aligned} \forall M_1, M_2, M_3 \ :\ & M_1 \xrightarrow{\star}_+ M_2 \text{ and } M_1 \xrightarrow{\star}_+ M_3 \\ & \Rightarrow \exists M, M' : M \cong_\alpha M' \text{ and } (M_2 \xrightarrow{\star}_+ M \wedge M_3 \xrightarrow{\star}_+ M') \end{aligned}$$

where $\cong_\alpha$ is the α-equivalence between terms, i.e. equivalence up to renaming of bound variables.

This implies that Eval$^+$ is again well-defined.

We extend $\mathcal{A}$ of Definition 1.14 (page 8) to $\mathcal{A}^+ : {C_\mathrm{A}}^+ \to \cup\{D^\sigma\}$ by:

$$\mathcal{A}^+(c) = \mathcal{A}(c) \qquad (c \in C_\mathrm{A})$$

$$(\mathcal{A})^+(\mathsf{pif}_o)(p)(x)(y) = \begin{cases} x & (p = tt) \\ y & (p = ff) \\ x & (p = \bot \text{ and } x = y) \\ \bot_o & (p = \bot \text{ and } x \neq y) \end{cases} \qquad (p \in \mathbb{B}_\bot \text{ and } x, y \in D^o)$$

The definition of the denotational semantics $[\![.]\!]_+$ for PCF$^+$ based on $\mathcal{A}^+$ should be straightforward.

Remark 1.20 Here we tried to follow Plotkin's original definitions and therefore added both $\mathsf{pif}_{\mathsf{bool}}$ and $\mathsf{pif}_{\mathsf{nat}}$ where any of the two would suffice. In fact we could have just as well added a constant $\mathsf{por}\colon \mathsf{bool} \to (\mathsf{bool} \to \mathsf{bool})$ with the following immediate reduction rule:

$$\begin{cases} \mathsf{por\ true}\ P \ \to\ \mathsf{true} \\ \mathsf{por}\ P\ \mathsf{true} \ \to\ \mathsf{true} \\ \mathsf{por\ false\ false} \ \to\ \mathsf{false} \end{cases}$$

and extended $\mathcal{A}$ to $\mathcal{A}^+$ by:

$$\mathcal{A}^+(\mathsf{por}) = \widehat{por}$$

where $\widehat{por}$ is the function defined in Definition 1.19, page 11. For a proof of the interdefinability of $\mathsf{pif}_{\mathsf{bool}}$, $\mathsf{pif}_{\mathsf{nat}}$ and por in PCF, see [Sto91].

Now the relation between the operational and denotational semantics is much better as the following theorem shows:

Theorem 1.21 *[Plo77]*
For any PCF$^+$*-terms* M_σ *and* N_σ*:*

$$M_\sigma \cong N_\sigma \Leftrightarrow \forall \rho \in \mathrm{Env}\ :\ [\![M_\sigma]\!]_+(\rho) = [\![N_\sigma]\!]_+(\rho)$$

In fact we have more:

Theorem 1.22 *[Plo77]*
Every finite element (see Definition 1.8, page 6) of any D^σ *is definable by a* PCF$^+$*-term.*

It seems intuitively reasonable to take the l.u.b.'s of recursively enumerable sets of finite elements of the model as the collection of *computable* elements. Any element defined by a PCF$^+$-term is computable (see [Plo77]) and as Theorem 1.22 says, any finite element of the model is captured by the language PCF$^+$. But it turns out that there are computable objects of the model not accounted for in the language. One such object is $\widehat{\exists} \in D^{(\mathsf{nat}\to\mathsf{bool})\to\mathsf{bool}}$ defined by:

Definition 1.23 [continuous existential quantifier : $\widehat{\exists}$]

$$\widehat{\exists}(g) = \begin{cases} \mathit{ff} & (g(\bot) = \mathit{ff}) \\ \mathit{tt} & (g(n) = \mathit{tt} \text{ for some } n \geq 0) \\ \bot & (\text{otherwise}) \end{cases}$$

We then proceed by adding a constant $\exists\colon (\mathsf{nat} \to \mathsf{bool}) \to \mathsf{bool}$ to ${C_{\mathrm{A}}}^+$ and denote the new set of constants as ${C_{\mathrm{A}}}^{++}$. We call the language based on this set

PCF^{++}. The following reduction rule is added to $\rightarrow_+$ to obtain $\rightarrow_{++}$. Note that $\overset{\star}{\rightarrow}_{++}$ is the transitive-reflexive closure of $\rightarrow_{++}$:

$$(xi)\begin{cases}\dfrac{F\Omega_{\mathsf{nat}} \overset{\star}{\rightarrow}_{++} \mathsf{false}}{\exists F \rightarrow_{++} \mathsf{false}} \\ \\ \dfrac{F\overline{m} \overset{\star}{\rightarrow}_{++} \mathsf{true}}{\exists F \rightarrow_{++} \mathsf{true}} \qquad (m \geq 0)\end{cases}$$

It can be shown that $\exists F \rightarrow_{++} \mathsf{true}$ and $\exists F \rightarrow_{++} \mathsf{false}$ cannot both hold at the same time, moreover $\overset{\star}{\rightarrow}_{++}$ satisfies Church-Rosser. Therefore Eval_{++} defined over PCF^{++} programs by:

$$\text{Eval}_{++}(M) = c \text{ iff } M \overset{\star}{\rightarrow}_{++} c$$

is well-defined.

Finally we define $\mathcal{A}^{++} : {C_{\text{A}}}^{++} \rightarrow \cup\{D^\sigma\}$ by:

$$\begin{aligned}\mathcal{A}^{++}(c) &= \mathcal{A}^{+}(c), \qquad (c \in {C_{\text{A}}}^{+}) \\ \mathcal{A}^{++}(\exists) &= \widehat{\exists}\end{aligned}$$

Again extending $[\![.]\!]_+$ to a denotational semantics $[\![.]\!]_{++}$ for PCF^{++} using $\mathcal{A}^{++}$ is straightforward.

Now the language is rich enough to define all computable elements of the model:

Theorem 1.24 ([Plo77]) *An element of D^σ is computable if and only if it is definable by a* PCF^{++}*-term.*

1.1.4 Theory versus Practice

Designing a language and its operational semantics, presenting a model, interpreting the language inside it and studying the relation between the operational and denotational semantics — one might like to categorize them as *theoretical issues* — are just part of a bigger challenge, though quite an important one. In practice other issues arise such as efficiency regarding time and/or space. Though in Part 1.1.3 we tried to summarize the theoretical issues and demonstrate the success in that regard, we never cared about the efficiency of computations. Whenever we felt a deficiency, we did not hesitate to remedy by any means.

First consider PCF and the immediate reduction rules over its terms (Definition 1.2, page 5). For any term, there is *at most* one rule that applies. Hence the operational semantics is *deterministic*. Imagine a machine implementing PCF, in the middle of a computation, reducing a term $C[M_1, \ldots, M_i, \ldots, M_k]$. If M_i is the subterm being worked on at the moment, there is no way the process will jump to another subterm M_j $(j \neq i)$ unless the computation on M_i is finished off with

M_i being evaluated to a constant. We try to formulate a property that captures this intuition and call a language satisfying this property *sequential*.

Sequentiality can be studied both syntactically and semantically. Plotkin's *activity lemma* [Plo77] is an example of a syntactic formulation, Berry's *syntactic sequentiality theorem* [Ber79] [5] is another, which for the reader can serve as a good motivation for a semantic investigation of sequentiality first introduced by Vuillemin [Vui74], though in reality Vuillemin's work preceded that of Berry's.

Here in this paper we pursue Vuillemin's semantic approach originally defined for functions over *flat cpo's*:

Definition 1.25 [flat cpo] Given any nonempty set X, $(X_\perp, \sqsubseteq)$ defined by:

$$X_\perp = X \cup \{\perp\} \qquad (\text{where } \perp \notin X)$$

and

$$\forall x, y \in X_\perp : x \sqsubseteq y \Leftrightarrow (x = \perp \ \vee \ x = y)$$

is a cpo. We call cpo's of this shape **flat cpo's**.

Examples of flat cpo's we use are $\mathbb{B}_\perp$ and $\mathbb{N}_\perp$ (Example 1.13, page 7) and $\mathbb{2}$ (Definition 3.2, page 31).

Definition 1.26 [sequential function (Vuillemin)] Let $D, D_1, \ldots, D_n$ be flat cpo's, and let $f : D_1 \times \cdots \times D_n \to D$ be continuous. Let $\overrightarrow{x} = (x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n$, and suppose that $f(\overrightarrow{x}) = \perp$. We say that f is sequential at $\overrightarrow{x}$ if either $f(\overrightarrow{z}) = \perp$ for all $\overrightarrow{z} \sqsupseteq \overrightarrow{x}$, or there exists i such that $x_i = \perp$ and:

$$\forall \overrightarrow{y} = (y_1, \ldots, y_n) : (\overrightarrow{y} \sqsupset \overrightarrow{x} \text{ and } f(\overrightarrow{y}) \neq \perp) \Rightarrow y_i \neq \perp$$

We say then that i is a sequentiality index for f at $\overrightarrow{x}$.
f is sequential if it is sequential at all $\overrightarrow{x}$ in its domain.

Note that Vuillemin-sequentiality is defined *only for first order functions over flat cpo's*. In fact the definition as it is cannot be generalized to higher types because a function like $\widehat{\exists}$ which is intuitively of an infinite parallel nature would become Vuillemin-sequential. But at first order the definition works well as it can be shown that for **compact** first order functions f in $\cup\{D^\sigma\}$, Vuillemin-sequentiality coincides with PCF-definability (Theorem 1.31, page 17).

Also of interest is the Vuillemin-sequentiality of **all** unary first order functions over flat cpo's. This can be used to show that although all first order PCF-definable functions in $\cup\{D^\sigma\}$ are Vuillemin-sequential [AC98, Exercise 6.5.5, page 137], the converse is not true as there are uncountably many elements in $D^{\mathsf{nat}\to\mathsf{nat}}$, all of which are Vuillemin-sequential, while there are only countably many PCF-definable elements in there.

Equally interesting, at least for our purposes, is Sieber's approach [Sie92] which proves to have a tight relation to Vuillemin's definition, as we shall see in Theorem

[5] see [AC98, section 2.4, page 41] for an English version.

1.31, page 17. But before that, we need to take a look at an important tool called *logical relations*.

1.2 Logical Relations

Notation 1.27 *Throughout this paper, by $\Lambda(C)$ we mean the extension of the simply-typed λ-calculus with a set of constants C.*

Definition 1.28 [Logical Relations] Let $\mathcal{M}_i$, $(1 \leq i \leq k)$ be k models of $\Lambda(C)$, and $D_i^\sigma = [\![\sigma]\!]_{\mathcal{M}_i}$ (where σ ranges over types), and for any ground type o, $R_k^o \subseteq D_1^o \times \cdots \times D_k^o$. Then a k-ary logical relation R_k between $\mathcal{M}_1, \ldots, \mathcal{M}_k$ can be built up from R_k^o's by the following definition for function type cases:

for any $f_1, \ldots, f_k \in D^{\sigma\to\tau}$:

$$(f_1, \ldots, f_k) \in R_k^{\sigma\to\tau} \Leftrightarrow$$
$$\forall (x_1, \ldots, x_k) \in R_k^\sigma : (f_1(x_1), \ldots, f_k(x_k)) \in R_k^\tau$$

$f \in D^\sigma$ is said to *preserve* the logical relation R_k if and only if:

$$(f, \ldots, f) \in R_k^\sigma$$

R is said to be a C-logical relation if for any $c : \sigma$ in C:

$$([\![c]\!]_{\mathcal{M}_1}, \ldots, [\![c]\!]_{\mathcal{M}_k}) \in R_k^\sigma$$

Logical relations are useful especially when it comes to establishing links between syntax and semantics as in Jung-Tiuryn's theorem on lambda-definability [JT92]. Of course Jung and Tiuryn used a more powerful class of logical relations called *Kripke logical relations* which unlike our definition, have varying arities.

Anyway Definition 1.28 is enough for our purposes. The important part of defining a logical relation is over the ground types, as that is the part over which we have control. Then having defined a suitable logical relation, we make extensive use of the following important lemma:

Lemma 1.29 (Fundamental lemma of logical relations) *Let R be a k-ary C-logical relation, C a set of constants, between k models of $\Lambda(C)$, namely $\mathcal{M}_1, \ldots, \mathcal{M}_k$. Then for any closed $\Lambda(C)$-term M of type σ we have:*

$$([\![M]\!]_{\mathcal{M}_1}, \ldots, [\![M]\!]_{\mathcal{M}_k}) \in R^\sigma$$

Note 1 *To find out more about logical relations as presented here, including a proof of Lemma 1.29, see [AC98, chapter 4, section 5].*

For an example of logical relations, let us mention an important class of logical relations known as *Sieber-sequential relations*:

Definition 1.30 [Sieber-sequential relations] Let $A \subseteq B \subseteq \{1, \ldots, n\}$ and consider the following n-ary relations $\mathcal{S}_{A,B}^n$ over ground types $o \in \{\mathsf{nat}, \mathsf{bool}\}$:

$$(d_1, \ldots, d_n) \in \mathcal{S}^n_{A,B} \Leftrightarrow (\exists i \in A\colon d_i = \bot) \text{ or } (\forall i, j \in B\colon d_i = d_j)$$

A **Sieber-sequential** relation is an n-ary logical relation $\mathcal{S}$ such that $\mathcal{S}^o$ is an intersection of relations of the form $\mathcal{S}^n_{A,B}$.

It can be shown that C-logical relations, where C is the set of all PCF constants C_{A} (see page 3) are exactly the Sieber-sequential relations of Definition 1.30 (see [AC98, Exercise 6.5.3, page 136]).

Now let us take some special cases of Sieber-sequential relations, namely $S_{k+1} \coloneqq \mathcal{S}^{k+1}_{\{1,\ldots,k\},\{1,\ldots,k+1\}}$, $(k \geq 1)$:

$$(x_1, \ldots, x_{k+1}) \in S_{k+1} \Leftrightarrow (\exists j \leq k\colon x_j = \bot) \text{ or } (x_1 = \cdots = x_{k+1} \neq \bot)$$

which help us demonstrate the relation between Sieber's and Vuillemin's approaches to sequentiality:

Theorem 1.31 *For a compact first order function f in $\cup\{D^\sigma\}$, the following are equivalent:*

(1) f is Vuillemin-sequential.
(2) f is definable in PCF.
(3) f is invariant under all $k+1$-ary relations S_{k+1}.

For a proof of this theorem the reader can refer to [AC98, Theorem 6.5.4, page 137].

1.3 Computation on Real Numbers: Real-PCF

PCF-programs are meant to output constants of the ground types bool or nat. Of course it is obvious that there are many more collection of objects over which we like to do computation. One such important case is the set of *real numbers*.

Traditionally we are used to computation on real numbers via *floating point approximations* which is satisfactory for everyday business but can prove to be extremely unreliable in special circumstances. Floating point computation carries the problem of *round-off errors* with it, which we try to ignore in everyday life applications for a variety of reasons. In [MM96] this subject is explored together with two interesting examples demonstrating the *unreliability of floating point approach*. Accordingly the idea of *exact real number computation* has been put forward which is, as opposed to floating point computation, reliable, i.e. the output produced is guaranteed to be correct. Moreover the results can be computed *effectively* (e.g. as opposed to BSS approach [BSS89]) to within any desired degree of accuracy.

Exact real number computation itself can be approached in two ways. At first people focused on *representation* while neglecting the issue of *datatypes* for real numbers, among which [BCOR86] is considered seminal. On the other hand, perhaps [Gia93] is among the earliest works where there is a clear distinction between a representation-dependent operational semantics and a representation-independent

denotational semantics. Di Gianantonio added to PCF a ground type which is interpreted as a domain of real numbers. This domain turns out to be algebraic (see Definition 1.10, page 7) and therefore *cannot have the real line as its subspace of maximal elements*. This creates the possibility of defining functions not extensional over real numbers ([Gia93, page 62] [6]).

Martín Escardó introduced Real-PCF following similar ideas [Esc97]. He added to PCF a ground type for real numbers interpreted as the so-called *unit interval domain* (see Definition 1.34, page 19) which has the interval $[0, 1]$ as its subspace of maximal elements. Also the problem of non-extensionality with Di Gianantonio's approach is avoided in Real-PCF. Of course there is much more to both Di Gianantonio's and Escardó's works. Here we present an overview of Real-PCF as it is the necessary background to the rest of the paper.

Definition 1.32 [Real-PCF] The set of Real-PCF types is generated by the grammar:

$$\sigma ::= \mathsf{bool} \mid \mathsf{nat} \mid I \mid \sigma \to \sigma$$

and the set of Real-PCF constants $\mathcal{RC}_{\mathrm{A}}$ is the extension of C_{A} (see page 3) with the following constants:

$$\begin{aligned} \mathsf{cons}_a &: I \to I \\ \mathsf{tail}_a &: I \to I \\ \mathsf{head}_r &: I \to \mathsf{bool} \\ \mathsf{pif}_I &: \mathsf{bool} \to I \to I \to I \end{aligned}$$

The aim is to take the ground type I as the type of *real numbers in the unit interval, i.e.* $[0, 1]$ and use the constants introduced in the definition for computation over them. In the above definition a ranges over intervals with rational end-points in $[0, 1]$, i.e.

$$a \in \{[p, q] \mid p, q \in \mathbb{Q} \cap [0, 1], p \leq q\}$$

These end-points must be distinct when a is a subscript for tail, i.e:

$$p < q$$

and

$$r \in \mathbb{Q} \cap (0, 1)$$

Notation 1.33 *We freely use the abbreviation RPCF for Real-PCF.*

The definition of the following terms for RPCF setting should be straightforward now and we omit them here. Moreover we abuse the notation where there is no confusion and use these terms for the meanings mentioned below:

1. $\mathcal{T}$: the set of *RPCF-terms*.
2. *Var* : the set of *RPCF-variables*.

[6] Of course on the same page, Di Gianantonio himself claims to have fixed the problem, but the author still believes that is not the case.

3. FV(M) : the set of the *free variables of the RPCF-term M*.

1.3.1 Denotational Semantics: Interval Domain Model

Let us begin with the ground type I whose denotation D^I we take to be:

Definition 1.34 [unit interval domain] The cpo $(\mathcal{I}, \sqsubseteq_{\mathcal{I}})$ defined by:

$$\mathcal{I} = \{[r, s] \mid r, s \in \mathbb{R}, 0 \leq r \leq s \leq 1\}$$

and

$$\forall a, b \in \mathcal{I} : a \sqsubseteq_{\mathcal{I}} b \Leftrightarrow a \supseteq b$$

is called the **unit interval domain**.

For simplicity, we denote $[r, r]$ by r, and in the other direction as well we talk about an element like $r \in [0, 1]$ where we really mean $[r, r] \in \mathcal{I}$.

We want the elements of $[0, 1]$ to be *maximal*, and use the *rational intervals* to approximate them. To make a proper distinction, we refer to the maximal elements as *total* real numbers whereas we call the others *partial* real numbers. As a smaller interval is a better approximation to a number than a bigger one, we want to have the *superset* relation to be the order on the intervals, hence the definition of $\sqsubseteq_{\mathcal{I}}$.

It is not difficult to show that $(\mathcal{I}, \sqsubseteq_{\mathcal{I}})$ is a cpo where supremum operation is simply defined to be *the set-theoretic intersection*, i.e.

$$\forall X \subseteq_{dir} \mathcal{I} : \sqcup X = \cap X$$

In fact $(\mathcal{I}, \sqsubseteq_{\mathcal{I}})$ is bounded complete as well:

$$\forall X \subseteq \mathcal{I} : X \text{ bounded } \Rightarrow l.u.b.\ X = \cap X$$

Moreover the countable set:

$$\mathcal{I}^o := \{[r, s] \in \mathcal{I} \mid r, s \in \mathbb{Q}\}$$

forms a basis (Definition 1.9, page 6) for $\mathcal{I}$ and makes it an *ω-continuous cpo* (Definition 1.11, page 7).

We let $\mathcal{I}$ denote the datatype I, hence:

Definition 1.35 [collection of domains for RPCF] $\{D_\sigma\}$ is *a collection of domains for RPCF* if:

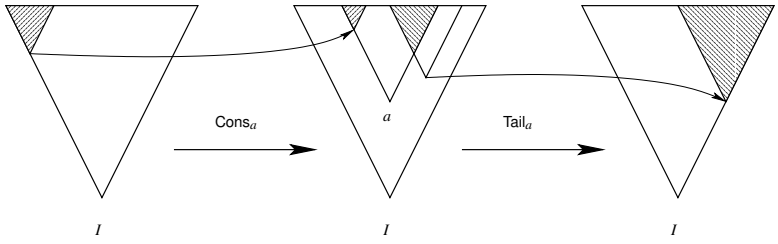
$$\begin{aligned} D^{\mathsf{bool}} &= \mathbb{B}_\perp \\ D^{\mathsf{nat}} &= \mathbb{N}_\perp \\ D^I &= \mathcal{I} \\ D^{\sigma\to\tau} &= [D^\sigma \to D^\tau] \end{aligned}$$

As we did for PCF (Part 1.1.2) we first try to interpret the constants via a function $\mathcal{RA}\colon \mathcal{RC}_{\mathrm{A}} \to \cup\{D_\sigma\}$ and then extend it in a natural way to a denotational semantics. For any constant $c \in C_{\mathrm{A}}$, we simply define $\mathcal{RA}(c) := \mathcal{A}(c)$, where $\mathcal{A}$ is defined as in Definition 1.14, page 8.

Remark 1.36 With each RPCF-type σ which is not a PCF-type, a new fix-point combinator Y_σ is added to the language, often without our notice!. Anyway, the interpretation is formulated as in Definition 1.14, page 8 for PCF-types, i.e.

$$\mathcal{RA}(\mathsf{Y}_\sigma)(f) = \sqcup\{f^n(\bot) \mid n \geq 0\} \qquad (f \in D^{\sigma\to\sigma})$$

For the proper RPCF-constants, let us present their denotations in a more intuitive fashion. From a geometric point of view, it does not matter if we choose two other numbers $r < s$, rather than 0 and 1, and build our interval domain upon it. Let us suppose $0 \leq r \leq s \leq 1$ and denote the interval $[r, s]$ by a, and the interval domain built upon it by $a\mathcal{I}$, bearing in mind that $a\mathcal{I}$ is the singleton domain in the case $r = s$. If $r < s$ then $\mathcal{RA}(\mathsf{cons}_a)$ is defined to be the scaling isomorphism — denoted by $Cons_a$ — from $\mathcal{I}$ to $a\mathcal{I}$, otherwise (i.e. $r = s$) it is simply the unique constant function available. We can simply consider the codomain to be the whole of $\mathcal{I}$, hence say $\mathcal{RA}(\mathsf{cons}_a)\colon \mathcal{I} \to \mathcal{I}$. There is apparently another scaling isomorphism from $a\mathcal{I}$ back to $\mathcal{I}$ (if $r \neq s$) that we call $Tail_a\colon a\mathcal{I} \to \mathcal{I}$. We can extend the domain of the function from $a\mathcal{I}$ to $\mathcal{I}$ so that it remains a morphism in the category **CPO**. In fact we consider the maximal extension (under the order relation on $[\mathcal{I} \to \mathcal{I}]$) and for simplicity denote it by the same name $Tail_a$, hence $Tail_a : \mathcal{I} \to \mathcal{I}$, and we put $\mathcal{RA}(\mathsf{tail}_a) = Tail_a$. The following figure may give a better intuition:



Now perhaps the following formulae are easier to follow:

$$\mathcal{RA}(\mathsf{cons}_a)(b) = [r + (s-r)x, r + (s-r)y] \tag{2}$$

$$\mathcal{RA}(\mathsf{tail}_a)(b) = [(x' - r)/(s-r), 1 - (s - y')/(s-r)] \tag{3}$$

where:

$$a = [r, s]$$

$$b = [x, y]$$

$$\begin{cases} x' = \min(\max(r, x), s) \\ y' = \max(\min(s, y), r) \end{cases}$$

The denotation of the constants head_r and pif_I can be easily described explicitly while their corresponding immediate reduction rule (see Definition 1.37, page 22)

tells all about their expected behaviour. Let r be a rational number $0 < r < 1$:

$$\forall x \in \mathcal{I}: [\mathcal{RA}(\mathsf{head}_r)](x) = \begin{cases} tt & \text{if} \quad (x = [x_1, x_2] \wedge x_2 < r) \\ ff & \text{if} \quad (x = [x_1, x_2] \wedge r < x_1) \\ \bot \text{ otherwise i.e } r \in x \end{cases}$$

$$\forall p \in \mathbb{B}_\bot, \forall x, y \in \mathcal{I}: [\mathcal{RA}(\mathsf{pif}_I)](p)(x)(y) = \begin{cases} x & \text{if } p = tt \\ y & \text{if } p = ff \\ x \sqcap y & \text{if } p = \bot \end{cases}$$

Having defined $\mathcal{RA}$ it is straightforward to define a denotational semantics

$$[\![.]\!]: \mathcal{T} \to (\text{Env} \to \cup\{D^\sigma\})$$

for RPCF, following the same style as we did for PCF in Definition 1.15, page 9. Note that we have not modified any symbol from PCF to RPCF (except $\mathcal{A}$ to $\mathcal{RA}$) as we are only dealing with RPCF throughout the rest of the paper, hence there should be no confusion.

Consider any two intervals $a, b \in \mathcal{I}$, where $a = [a_1, a_2]$ and $b = [b_1, b_2]$ and define:

$$ab = [a_1 + (a_2 - a_1)b_1, a_1 + (a_2 - a_1)b_2]$$

It is easy to verify that:

$$\mathcal{RA}(\mathsf{cons}_{ab}) = \mathcal{RA}(\mathsf{cons}_a).\mathcal{RA}(\mathsf{cons}_b)$$

where . is just functional composition.

Also, consider $a, b \in \mathcal{I}$, where $a = [a_1, a_2]$ and $b = [b_1, b_2]$ and this time subject to the conditions:

$$a_1 \neq a_2 \qquad \text{and} \qquad a \sqsubseteq b$$

then there exists a unique $c \in \mathcal{I}$ such that $ac = b$, which we denote by $b \setminus a$. In fact:

$$c = [(b_1 - a_1)/(a_2 - a_1), (b_2 - a_1)/(a_2 - a_1)]$$

Now it seems reasonable to use cons_a's ($a \in \mathcal{I}$, with rational end-points) as digits in order to represent real numbers in the interval $[0, 1]$. The idea is to represent any *shrinking* sequence of intervals with rational end-points by a sequence of the form:

$$(\mathsf{cons}_{a_1},\ \mathsf{cons}_{a_1}\mathsf{cons}_{a_2}, ..., \mathsf{cons}_{a_1}...\mathsf{cons}_{a_n}, ...)$$

This sequence of intervals converges to an element $a \in \mathcal{I}$ which can be partial or total depending on the nature of the sequence. We simply represent this element a by:

$$\mathsf{cons}_{a_1}\mathsf{cons}_{a_2}...\mathsf{cons}_{a_n}\ldots$$

1.3.2 Operational Semantics

We extend the immediate reduction relation for PCF (Definition 1.2, page 5) to one for RPCF and still denote it by $\to$. The aim is to reduce any Real-PCF program M of type I to some $\mathsf{cons}_a M'$ — where a has rational end-points — and then continue reducing M' to $\mathsf{cons}_{a'} M''$, and so on. This way we produce a stream of digits.

Before presenting the definition, take note of the following:

(i) There are RPCF-types σ that are not PCF-types, correspondingly there are new *fix-point combinators* $\mathsf{Y}_\sigma : (\sigma \to \sigma) \to \sigma$ that are not PCF-constants. But still the reduction rule is the same as clause (v) of Definition 1.2, page 5, i.e.

$$\mathsf{Y}_\sigma M = M(\mathsf{Y}_\sigma M)$$

in particular:

$$\mathsf{Y}_I\ \mathsf{cons}_{[0,1/2]} = 0, \qquad \mathsf{Y}_I\ \mathsf{cons}_{[1/2,1]} = 1$$

(ii) For intervals $a, b \in \mathcal{I}$ where $a = [a_1, a_2]$ and $b = [b_1, b_2]$ we define:

$$a \leq b \coloneqq a_2 \leq b_1$$

(iii) Similarly for real number r and the interval $a = [a_1, a_2]$ we define:

$$a < r \coloneqq a_2 < r$$

Definition 1.37 [immediate reduction relation for RPCF] The immediate reduction relation $\to$ is the extension of the corresponding relation from PCF (Definition 1.2,

page 5) to Real-PCF by the following rules:

1. $\mathsf{cons}_a(\mathsf{cons}_b M) \;\to\; \mathsf{cons}_{ab} M$
2. $\mathsf{tail}_a(\mathsf{cons}_b M) \;\to\; \mathsf{Y}_I\, \mathsf{cons}_{[0,1/2]} \qquad (\text{if } b \le a)$
3. $\mathsf{tail}_a(\mathsf{cons}_b M) \;\to\; \mathsf{Y}_I\, \mathsf{cons}_{[1/2,1]} \qquad (\text{if } a \le b)$
4. $\mathsf{tail}_a(\mathsf{cons}_b M) \;\to\; \mathsf{cons}_{b\setminus a} M \qquad (\text{if } a \sqsubseteq b \text{ and } a \neq b)$
5. $\mathsf{tail}_a(\mathsf{cons}_b M) \;\to\; \mathsf{cons}_{(a\sqcup b)\setminus a}(\mathsf{tail}_{(a\sqcup b)\setminus b} M)$

   $(\text{if } a \uparrow b,\ a \not\sqsubseteq b,\ b \not\sqsubseteq a,\ a \not\leq b,\ b \not\leq a)$
6. $\mathsf{head}_r(\mathsf{cons}_a M) \;\to\; \mathsf{true} \qquad (\text{if } a < r)$
7. $\mathsf{head}_r(\mathsf{cons}_a M) \;\to\; \mathsf{false} \qquad (\text{if } a > r)$
8. $\mathsf{pif}\ \mathsf{true}\ M\ N \;\to\; M, \qquad \mathsf{pif}\ \mathsf{false}\ M\ N \;\to\; N$
9. $\mathsf{pif}\ L\ (\mathsf{cons}_a M)\ (\mathsf{cons}_b N) \;\to$

   $\mathsf{cons}_{a\sqcap b}(\mathsf{pif}\ L\ (\mathsf{cons}_{a\setminus(a\sqcap b)} M)\ (\mathsf{cons}_{b\setminus(a\sqcap b)} N))$

   $(\text{if } a\sqcap b \neq \bot)$
10. $\dfrac{N \to N'}{MN \to MN'} \qquad (M \in \{\mathsf{cons}_a,\ \mathsf{tail}_a,\ \mathsf{head}_r, \mathsf{pif}_I\})$
11. $\dfrac{M \to M'}{\mathsf{pif}_I\ L\ M \;\to\; \mathsf{pif}_I\ L\ M'} \qquad \dfrac{N \to N'}{\mathsf{pif}_I\ L\ M\ N \;\to\; \mathsf{pif}_I\ L\ M\ N'}$

We denote the reflexive and transitive closure of $\to$ by $\to^*$.

Definition 1.38 [operational semantics for RPCF] The map Eval (Definition 1.3, page 5) can be extended to a partial map over RPCF-programs (which we still denote by Eval) by the following case for programs M of type I:

$$\mathrm{Eval}(M) := \{a \in \mathcal{I} \mid M \to^* \mathsf{cons}_a M',\ \text{for some } M'\}$$

Note 2 *For a more precise and also comprehensive treatment of Real-PCF, see [Esc97].*

2 Sequentiality and Parallelism in RPCF

Unlike PCF, RPCF has a parallel operator as a constant, i.e. pif_I. By the time Martín Escardó put forward RPCF [Esc97], it was already speculated that representation-independent real number computation needs parallel operators ([BCOR86,Gia93]). Therefore pif_I was included in RPCF right from the beginning. In [Esc97] Escardó shows that all computable elements of type at most 1 in the interval domain model are definable in RPCF. Of course, like PCF, there are higher order objects such as $\widehat{\exists}$ (Definition 1.23, page 13) not definable in RPCF. But by adding a constant $\exists$ for existential quantification, the language becomes *universal* for the model, i.e.

all computable objects of any order in the interval domain model are definable ([Esc96]).

Though adding parallel operators to the language solves the definability problems, they come at a heavy cost, i.e. the issue of efficiency in practice. Therefore we would rather have a more efficient substitute for pif_I. We need to analyze the language and its model more carefully to have a better view of our choices. One might think of getting rid of any kind of parallelism in the language. This idea was ruled out by Escardó, Hofmann and Streicher in [EHS98] where they proved that even if functions as basic as any continuous extension of mediation $\oplus\colon [0,1] \to [0,1] \to [0,1]$ defined by:

$$\forall x, y \in [0,1]\colon x \oplus y := \frac{x+y}{2} \tag{4}$$

to the whole interval domain $\mathcal{I}$, are going to be definable in the language, then the existence of some parallel mechanism is necessary. Of course this result crucially depends on some specific conditions, some of which need not necessarily hold in an ideal setting. For example, in the interval domain model — the model in which this result was studied — all functions on real numbers are *extensional* at both partial and total real numbers, i.e. for any RPCF-type σ:

$$\forall f \in D_{I\to\sigma}, x \in \mathcal{I}, y_1, y_2 \in D_\sigma\colon f(x) = y_1 \wedge f(x) = y_2 \Rightarrow y_1 =_{D_\sigma} y_2$$

where $=_{D_\sigma}$ is the equality on D_σ. In practice, we generally do not care whether such a function is extensional at partial real numbers or not.

While there is a search for more efficient substitutes for pif_I one needs to be assured of the segment of RPCF without pif_I being sequential. Let $w\mathcal{RC}_{\mathrm{A}}$ denote the set of RPCF-constants with pif_I removed:

Definition 2.1 [$w\mathcal{RC}_{\mathrm{A}}$]

$$w\mathcal{RC}_{\mathrm{A}} := \mathcal{RC}_{\mathrm{A}} \setminus \{\mathsf{pif}_I\}$$

where $\mathcal{RC}_{\mathrm{A}}$ is as in Definition 1.32, page 18.

Definition 2.2 [weak-RPCF (wRPCF)] By **wRPCF** we mean the segment of RPCF built upon the set of constants $w\mathcal{RC}_{\mathrm{A}}$ (Definition 2.1). In other words wRPCF is RPCF without pif_I.

Remark 2.3 Note that the set of wRPCF-types is the same as the set of RPCF-types.

First we need to fix a criterion for sequentiality and test wRPCF against it. The one we consider in this paper is a generalization of Vuillemin-sequentiality (Definition 1.26, page 15) to functions over the interval domain. This way we will show that the first order wRPCF-definable functions are sequential. [7]

Definition 2.4 [generalized Vuillemin-sequentiality] Suppose

$$\forall j \in \{0, \ldots, n\}\colon D_j \in \{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\}$$

[7] In [Far] we considered another criteria, i.e. *conservativity over PCF*.

then

$$f\colon D_1 \times \cdots \times D_n \to D_0$$

is said to be **Vuillemin-sequential** (or simply **sequential**) at $\overrightarrow{x} = (x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n$ if either:

1. $f(\overrightarrow{z}) = f(\overrightarrow{x})$ for all $\overrightarrow{z} \sqsupseteq \overrightarrow{x}$

or otherwise

2. $\exists i \in \{1, \ldots, n\}\colon \forall \overrightarrow{z} = (z_1, \ldots, z_n) \in D_1 \times \cdots \times D_n$:

$$\text{if } \overrightarrow{z} \sqsupset \overrightarrow{x} \text{ and } f(\overrightarrow{z}) \sqsupset f(\overrightarrow{x}) \text{ then } z_i \sqsupset x_i$$

This i is called a **sequentiality index** for f at $\overrightarrow{x}$.
f is **sequential** if it is sequential at all $\overrightarrow{x}$ in its domain.

Remark 2.5 Note that by the above definition, all unary first order functions of the interval domain model are trivially sequential.

Although as discussed before (page 15) Vuillemin-sequentiality cannot be freely generalized to any domain, our generalization can be made legitimate on the following accounts:

Intuition : Think of a first-order function f of k arguments as a black-box with k channels of input. The intuition behind sequentiality is that we want f to be called sequential if at any time and any stage of computation process, f is "looking at" only one of its arguments. If this argument is the i-th one we like to call i the index of sequentiality (at this stage in the process). Now if the information from any other channel is increased, it cannot improve the output of f as f is focusing only on the i-th argument.

Matching the expectations : As we shall see (from Remark 2.6 on page 26 and Lemma 2.8 on page 27) any function with intuitive parallel behaviour is not Vuillemin-sequential. In particular, constants like pif_I or por are not Vuillemin-sequential.

We prove that any first order wRPCF-definable function is sequential. Our proof is a generalization of the proof of Theorem 1.31 (page 17) as presented in [AC98, Theorem 6.5.4, page 137], which can give a much better idea as to why we define the logical relation S_{k+1} as in equation (5), page 25. Also, it is worth mentioning that both proofs are crucially based on the so-called *fundamental lemma of logical relations* (Lemma 1.29, page 16).

Let us suppose that for any RPCF-type σ, D^σ is the interpretation of type σ in the interval domain model (see Definition 1.35, page 19). Now for any $k \geq 1$, we define a relation S^o_{k+1} of arity $k+1$ over the elements of D^o, where o is a ground type, i.e bool, nat or I:

$$(x_1, \ldots, x_k, x_{k+1}) \in S^o_{k+1} \Leftrightarrow \exists j \leq k\colon (\forall i \leq k+1\colon x_j \sqsubseteq x_i) \tag{5}$$

We build up logical relations S_{k+1} for each $k \geq 1$ over these basic cases, as in Definition 1.28, page 16. It is pretty easy to show that each S_{k+1} is in fact a $w\mathcal{RC}_{\mathrm{A}}$-

logical relation. If c is a unary constant in $w\mathcal{RC}_{\mathrm{A}}$, then the S_{k+1}'s are preserved as a result of $\mathcal{RA}(c)$ being monotone (in fact continuous). If $c \in \{\mathsf{if}_{\mathsf{bool}}, \mathsf{if}_{\mathsf{nat}}\}$, then it can be verified by case analysis over the boolean argument.

Remark 2.6 It is also worth mentioning that the so-called parallel operators do not preserve all S_{k+1}'s. As an example, take $\widehat{por}$ (see Definition 1.19, page 11) and $k = 2$. The following figure can give a better picture of why this is the case. The first two left columns are elements of S_3^{bool} whereas the rightmost column, which is the result of applying $\widehat{por}$ over the elements of the first two, is not in S_3^{bool}:

$$\begin{array}{cccc} \top & \bot & \overset{\widehat{por}}{\longrightarrow} & \top \\ \bot & \top & \overset{\widehat{por}}{\longrightarrow} & \top \\ \bot & \bot & \overset{\widehat{por}}{\longrightarrow} & \bot \end{array}$$

The only non-trivial case may be the so-called *fix-point operators* Y_σ. Remember (Remark 1.36, page 20) that for each wRPCF-type σ there is a wRPCF-constant $\mathsf{Y}_\sigma : (\sigma \to \sigma) \to \sigma$, with the denotation $[\![\mathsf{Y}_\sigma]\!] \in D^{(\sigma\to\sigma)\to\sigma}$ defined by:

$$\forall f \in D^{\sigma\to\sigma} : [\![\mathsf{Y}_\sigma]\!](f) = \sqcup\{f^{(n)}(\bot_{D^\sigma}) \mid n \geq 0\} \tag{6}$$

So, to prove that Y_σ preserves all S_{k+1}'s, we first show that the set of S_{k+1} invariant elements of any D^σ forms a so-called *inclusive predicate*. That is, we have to verify two properties at each type σ:

(i) $(\bot_{D_1^\sigma}, \ldots, \bot_{D_{k+1}^\sigma}) \in S_{k+1}^\sigma$, where D_i^σ is the denotation of the type σ in the model $\mathcal{M}_i$, for each $i \in \{1, \ldots, k+1\}$.

(ii) If $\{x^i = (x_1^i, \ldots, x_{k+1}^i) \in S_{k+1}^\sigma \mid i \in \mathbb{N}\}$ is an ascending chain in S_{k+1}^σ, then $\sqcup\{x^i\} \in S_{k+1}^\sigma$.

We prove this two-fold fact by induction over the type σ:

- If σ is a ground type then (i) holds by the definition of S_{k+1}^σ at ground types. To prove (ii), suppose the sequence $\{x^i\}$ is given. For each $i \geq 0$, there exists an index $j_i \leq k$, such that $\forall m \leq k+1 : x_{j_i}^i \sqsubseteq x_m^i$, because $x^i \in S_{k+1}^\sigma$ and σ is ground. So, in particular, there must be an index $l \leq k$ for which there are infinitely many i's — e.g. elements of an infinite set $A \subseteq \mathbb{N}$ — with $\forall m \leq k+1 : x_l^i \sqsubseteq x_m^i$. Hence we have:

$$\forall m \leq k+1 : \sqcup\{x_l^i \mid i \in \mathbb{N}\} = \sqcup\{x_l^i \mid i \in A\} \sqsubseteq \sqcup\{x_m^i \mid i \in A\} = \sqcup\{x_m^i \mid i \in \mathbb{N}\}$$

  which means $\sqcup\{x^i \mid i \in \mathbb{N}\} \in S_{k+1}^\sigma$.
- If $\sigma = \sigma_1 \to \sigma_2$ then both (i) and (ii) hold by induction hypothesis on σ_2.

Proposition 2.7 *For any $k \geq 1$, S_{k+1} (as defined in (5), page 25) is a C-logical relation, where C is the set of wRPCF-constants $w\mathcal{RC}_{\mathrm{A}}$ (Definition 2.1, page 24).*

Proof. It remains to show the proof for the constants $\mathsf{Y}_\sigma : (\sigma \to \sigma) \to \sigma$. To prove that Y_σ preserves S^σ_{k+1}, by (6) (page 26) it suffices to show three properties at each type σ:

(i) $(\bot_{D^\sigma_1}, \ldots, \bot_{D^\sigma_{k+1}}) \in S^\sigma_{k+1}$, where D^σ_i is the denotation of the type σ in the model $\mathcal{M}_i$, for each $i \in \{1, \ldots, k+1\}$.

(ii) If $(x_1, \ldots, x_{k+1}) \in S^\sigma_{k+1}$ and $(F_1, \ldots, F_{k+1}) \in S^{\sigma\to\sigma}_{k+1}$, then $(F_1 x_1, \ldots, F_{k+1} x_{k+1}) \in S^\sigma_{k+1}$.

(iii) If $\{x^i = (x^i_1, \ldots, x^i_{k+1}) \in S^\sigma_{k+1} \mid i \in \mathbb{N}\}$ is an ascending chain in S^σ_{k+1}, then $\sqcup\{x^i\} \in S^\sigma_{k+1}$.

We have already shown that (i) and (iii) hold at each type σ, and (ii) holds by definition of logical relations. □

It is possible to embark on proving the sequentiality of first order wRPCF-definable function of the interval domain model right now. But perhaps presenting the relation between S_{k+1}'s and sequentiality as a separate result would give a better understanding of why S_{k+1}'s were chosen in the first place:

Lemma 2.8 *Let*

$$f : D_1 \times \cdots \times D_n \to D_0$$

where

$$\forall j \in \{0, \ldots, n\} : D_j \in \{\mathbb{B}_\bot, \mathbb{N}_\bot, \mathcal{I}\}$$

be a first order function in the interval domain model of RPCF. Then f is Vuillemin-sequential if and only if it preserves all S_{k+1}'s, $(k \geq 1)$.

Proof.

($\Leftarrow$) : Suppose f is as in the statement of the theorem, and preserves all S_{k+1}'s, $(k \geq 1)$. We prove that f is Vuillemin-sequential by contradiction:

Suppose f is *not Vuillemin-sequential* at a point $x = (x_1, \ldots, x_n)$ in its domain. Define:

$$A = \{j \mid 1 \leq j \leq n, x_j \text{ is not maximal}\}$$

As $A \neq \emptyset$ [8], without loss of generality let us suppose $A = \{1, \ldots, k\}$, $k \leq n$. For any $j \in A$, there must exist an $x^j = (x^j_1, \ldots, x^j_n)$ such that:

(i) $x^j_j = x_j$
(ii) $x^j_i = x_i$, for all $i > k$, if any [9].
(iii) $x^j \sqsupset x$
(iv) $f(x^j) \sqsupset f(x)$

Now consider the $(k+1) \times n$ matrix X defined by:

[8] In fact A has at least two elements, otherwise f would be vacuously sequential at x.
[9] Notice that these are the maximal elements.

$$X_{i,j} = \begin{cases} x_j^i \text{ if } i \leq k \\ x_j \text{ if } i = k+1 \end{cases}$$

$$\begin{bmatrix} x_1^1 & x_2^1 & \dots & x_k^1 & x_{k+1}^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_k^2 & x_{k+1}^2 & \dots & x_n^2 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ x_1^k & x_2^k & \dots & x_k^k & x_{k+1}^k & \dots & x_n^k \\ x_1 & x_2 & \dots & x_k & x_{k+1} & \dots & x_n \end{bmatrix}$$

It is easy to see that for any $j \leq n$, the j-th column of X is an element of $S_{k+1}^{\sigma_j}$, because:

- If $j \leq k$ then $\forall m \leq k+1 : X_{j,j} = x_j^j = x_j \sqsubseteq x_j^m = X_{m,j}$
- If $j > k$, then $X_{1,j} = X_{2,j} = \cdots = X_{k+1,j}$

Applying f to all the **rows** of X results in the vector :

$$\begin{bmatrix} f(x^1) \\ \vdots \\ f(x^k) \\ f(x) \end{bmatrix}$$

As f is supposed to preserve S_{k+1}, for an index $i_0 \leq k$, $f(x^{i_0})$ is the minimum element of the above vector (under $\sqsubseteq$). In particular $f(x^{i_0}) \sqsubseteq f(x)$. On the other hand, by (iv) above, we have $f(x^{i_0}) \sqsupset f(x)$, a contradiction.

($\Rightarrow$) : Assume f is Vuillemin-sequential, and for any $i \in \{1, \dots, n\}$, a $k+1$-dimensional vector $x^i = (x_1^i, \dots, x_{k+1}^i)$ is given such that:

$$\exists j \leq k : \forall m \leq k+1 : x_j^i \sqsubseteq x_m^i$$

We denote the least such j as $j(i)$. Take $i_0 \in \{1, \dots, n\}$ to be an index of sequentiality for f at $(x_{j(1)}^1, \dots, x_{j(n)}^n)$. Then we have:

$$f(x_{j(1)}^1, \dots, x_{j(i_0)}^{i_0}, \dots, x_{j(n)}^n) = f(x_{j(i_0)}^1, \dots, x_{j(i_0)}^{i_0}, \dots, x_{j(i_0)}^n)$$

because

$$(x_{j(1)}^1, \dots, x_{j(i_0)}^{i_0}, \dots, x_{j(n)}^n) \sqsubseteq (x_{j(i_0)}^1, \dots, x_{j(i_0)}^{i_0}, \dots, x_{j(i_0)}^n)$$

and the two vectors agree on their i_0-th components. On the other hand for any $i \neq i_0$ we have:

$$f(x_i^1, \dots, x_i^n) \sqsupseteq f(x_{j(1)}^1, \dots, x_{j(n)}^n)$$

because

$$(x_i^1, \dots, x_i^n) \sqsupseteq (x_{j(1)}^1, \dots, x_{j(n)}^n)$$

therefore, for all $i \leq k+1$:

$$f(x^1_{j(i_0)}, \ldots, x^n_{j(i_0)}) \sqsubseteq f(x^1_i, \ldots, x^n_i)$$

which shows that f preserves the logical relation S_{k+1} (with $j(i_0)$ being the required index of the minimum element).

□

Combining proposition 2.7 (page 26) and Lemma 2.8 (page 27), we obtain the main result of the paper:

Theorem 2.9 *Let*

$$f\colon D_1 \times \cdots \times D_n \to D_0$$

where

$$\forall j \in \{0, \ldots, n\}\colon D_j \in \{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\}$$

be a first order ***wRPCF-definable*** *function in the interval domain model of RPCF. Then f is Vuillemin-sequential.*

Therefore, by virtue of this theorem, functions like *parallel-or* are ruled out from being definable in wRPCF (see Remark 2.6, page 26).

Although we will show that not all unary first order functions of the interval domain model are definable in wRPCF (see Lemma 3.9, page 34 and Part 3.1, page 35), they are Vuillemin-sequential (see Remark 2.5, page 25) and adding them to the language does not affect the sequentiality:

Corollary 2.10 *Let $\Gamma = w\mathcal{RC}_A + C$, where $\forall c \in C\colon (\llbracket c \rrbracket\colon D_1 \to D_2$ is a computable unary first order function, i.e : $D_1, D_2 \in \{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\})$, and denote the segment of RPCF built upon the constants in Γ by $wRPCF_\Gamma$. Then any first order $wRPCF_\Gamma$-definable function is Vuillemin-sequential.*

Proof (Sketch) For any $c \in C$, $\llbracket c \rrbracket$ is monotone (because it is computable), hence preserves S_{k+1} for any k. Now Lemma 2.8 (page 27) is applicable. □

Remark 2.11 Generally logical relations are helpful for studying the behaviour of (first order) functions definable in extensions of λ-calculus. As a simple example, one can show that excluding $\mathsf{if}_{\mathsf{bool}}$ and $\mathsf{if}_{\mathsf{nat}}$ from wRPCF, leaves us with a language in which all functions are essentially **unary**. Let $\Delta := w\mathcal{RC}_A \setminus \{\mathsf{if}_{\mathsf{bool}}, \mathsf{if}_{\mathsf{nat}}\}$, and $wRPCF_\Delta$ be the segment of wRPCF built upon the set of constants Δ. For any $k \geq 1$, let T_{k+1} be the $k+1$-ary logical relation defined at the ground types $o \in \{\mathsf{bool}, \mathsf{nat}, I\}$ by:

$$T^o_{k+1} = \{(x_1, \ldots, x_{k+1}) \in (D^o)^{k+1} \mid \exists j \leq k\colon x_j = x_{k+1}\}$$

then similar to the proof of the main theorem, it can be shown that $wRPCF_\Delta$-definable functions preserve T_{k+1} for any $k \geq 1$. The following counter-example shows how $\mathsf{if}_{\mathsf{nat}}$ does not preserve T_3, where the first three columns on the left are

elements of T_3, while the rightmost column — the result of applying $\mathsf{if}_{\mathsf{nat}}$ to the elements of the first three — is not:

$$\begin{array}{lcl} tt\ 0\ 1 & \overset{\mathsf{if}_{\mathsf{nat}}}{\rightarrow} & 0 \\ ff\ 1\ 0 & \overset{\mathsf{if}_{\mathsf{nat}}}{\rightarrow} & 0 \\ tt\ 1\ 0 & \overset{\mathsf{if}_{\mathsf{nat}}}{\rightarrow} & 1 \end{array}$$

Now using this, one can get a model theoretic proof of the following simple fact:

Let $f\colon D_1 \times \cdots \times D_n \to D_0$ (where $D_j \in \{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\}$ for all $0 \leq j \leq n$) be a first order $wRPCF_\Delta$-definable function. Then for some $i \in \{1, \ldots, n\}$, there exists a $wRPCF_\Delta$-definable $f_i\colon D_i \to D_0$ such that :

$$f = f_i \circ \pi_i \ \ (\pi_i\colon D_1 \times \cdots \times D_n \to D_i \text{ projection})$$

In words, first order $wRPCF_\Delta$-definable functions are essentially functions of one argument.

3 Piece-wise affinity

In this section we derive another non-definability result in a segment of RPCF which contains wRPCF (Definition 2.2, page 24) as a sub-segment. For that we need to clarify some terms and definitions:

Definition 3.1 [piece-wise affine] Let $-\infty \leq p \leq q \leq +\infty$ and $f\colon [p,q] \to \mathbb{R}$. We say that f is **affine** if and only if:

$$\exists m, n \in \mathbb{R}\colon\ \forall x \in [p,q]\colon\ f(x) = mx + n \tag{7}$$

A continuous function f is said to be **piece-wise affine** if and only if for some:

$$\{p_0, \ldots, p_i, p_{i+1}, \ldots, p_n\} \subseteq [p,q]$$

such that:

$$p = p_0 \leq \cdots \leq p_i \leq p_{i+1} \leq \cdots \leq p_n = q$$

$f \upharpoonright [p_i, p_{i+1}]$, i.e. the restriction of f to $[p_i, p_{i+1}]$, is affine for all $0 \leq i \leq n-1$.

Now take $a = [r,s] \in \mathcal{I}$ and consider $[\![\mathsf{cons}_\mathsf{a}]\!]$ acting on the maximal elements of $\mathcal{I}$, i.e. $[0,1]$. By equation (2) (page 20) we have:

$$\forall x \in \mathcal{I}\colon\ [\![\mathsf{cons}_\mathsf{a}]\!](x) = (s-r)x + r$$

therefore by taking:

$$m \coloneqq s - r \qquad n \coloneqq r$$

in equation (7), we observe that $[\![\mathsf{cons}_\mathsf{a}]\!]$ acts as an affine (hence trivially piece-wise affine) function over the maximal elements of $\mathcal{I}$.

Assuming $r \neq s$ let us take a look at how $[\![\mathsf{tail_a}]\!]$ acts over $[0,1]$. According to equation (3) (page 20), and by taking:

$$(p_0 := 0) \ \leq \ (p_1 := r) \ \leq \ (p_2 := s) \ \leq \ (p_3 := 1)$$

we see that substituting:

(i)
$$m := 0 \qquad n := 0$$
in equation (7) makes $[\![\mathsf{tail_a}]\!]$ affine over $[p_o, p_1] = [0, r]$.

(ii)
$$m := 1/(s-r) \qquad n := 0$$
in equation (7) makes $[\![\mathsf{tail_a}]\!]$ affine over $[p_1, p_2] = [r, s]$.

(iii)
$$m := 0 \qquad n := 1$$
in equation (7) makes $[\![\mathsf{tail_a}]\!]$ affine over $[p_2, p_3] = [s, 1]$.

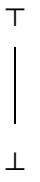
hence $[\![\mathsf{tail_a}]\!]$ is piece-wise affine over $[0,1]$.

Observing the constants $\mathsf{cons_a}$ and $\mathsf{tail_a}$ being interpreted as piece-wise affine functions over $[0,1]$, one might guess this property can be preserved by all wRPCF constructions on the basis that wRPCF is in fact weak when it comes to defining total functions over real types using definition by cases. With a suitable choice of logical relations, we can prove this guess for a language slightly more powerful than wRPCF.

Definition 3.2 [Sierpinski space] We call the flat cpo $(\mathbb{2}, \sqsubseteq)$ where:

$$\mathbb{2} = \{\bot, \top\}$$
$$\forall x, y \in \mathbb{2} \colon \ x \sqsubseteq y \Leftrightarrow (x = y \ \vee \ x = \bot)$$

the **Sierpinski space**. In picture:



Definition 3.3 [$\widehat{wpor}$] $\widehat{wpor} \colon \mathbb{2} \times \mathbb{2} \to \mathbb{2}$ is defined by:

$$\forall a, b \in \mathbb{2} \colon \widehat{wpor}(a,b) = \begin{cases} \top \text{ if a} = \top \text{ or b} = \top \\ \bot \text{ otherwise} \end{cases}$$

Definition 3.4 [weakly-parallel RPCF (wPR)] The set of wPR-types are generated by the grammar:

$$\sigma ::= \mathsf{S} \mid \mathsf{bool} \mid \mathsf{nat} \mid I \mid \sigma \to \sigma$$

$\{D^\sigma\}$ is a collection of domains for wPR if:

$$\begin{aligned} D^{\mathsf{S}} &= \mathbb{2} \\ D^{\mathsf{bool}} &= \mathbb{B}_\perp \\ D^{\mathsf{nat}} &= \mathbb{N}_\perp \\ D^{\mathcal{I}} &= \mathcal{I} \\ D^{\sigma\to\tau} &= [D^\sigma \to D^\tau] \end{aligned}$$

The set $w\mathcal{PRC}_{\mathrm{A}}$ of wPR-constants is defined as:

$$w\mathcal{PRC}_{\mathrm{A}} := w\mathcal{RC}_{\mathrm{A}} \cup \{\star\colon \mathsf{S},\ \mathsf{wpor}\colon \mathsf{S}\to\mathsf{S}\to\mathsf{S}\}$$

where $w\mathcal{RC}_{\mathrm{A}}$ is the set of wRPCF-constants as in Definition 2.1, page 24.

By **wPR** we mean the extension of wRPCF built upon $w\mathcal{PRC}_{\mathrm{A}}$, the immediate reduction rules of which are those of wRPCF extended with the following rules:

$$\begin{cases} \mathsf{wpor}\ \star\ N\ \to\ \top \\ \mathsf{wpor}\ M\ \star\ \to\ \top \end{cases}$$

$$\frac{M \to M'}{\mathsf{wpor}\ M\ \to\ \mathsf{wpor}\ M'}$$

$$\frac{N \to N'}{\mathsf{wpor}\ M\ N\ \to\ \mathsf{wpor}\ M\ N'}$$

and whose denotational semantics is that of wRPCF extended with the following interpretations of the new constants: [10]

$$[\![\mathsf{wpor}]\!] = \widehat{wpor}$$
$$[\![\star]\!] = \top$$

where $\widehat{wpor}$ is the function defined in Definition 3.3, page 31.

Note 3 *In the previous definition (i.e. Definition 3.4) we mentioned the basics of wPR, so we leave the exact definition to the reader, as we believe with the material presented here, a method similar to that of defining RPCF (Part 1.3) would lead to a complete definition for wPR in a straightforward manner.*

Remark 3.5 wPR as it is cannot be regarded as a segment of RPCF due to the presence of $\mathbb{2}$ and the constants that come with it.

[10] There are new fix-point operators due to the existence of types not-present in wRPCF, but the formula for their interpretation is the same, so we omit it!

For each $k \geq 1$, we define a k-ary logical relation R_k which — to some extent — carries the meaning of piece-wise affinity on first order functions. When $o \in \{\mathsf{S}, \mathsf{bool}, \mathsf{nat}\}$, R^o_k is simply defined as:

$$\forall (x_1, \ldots, x_k) \in (D^o)^k \colon (x_1, \ldots, x_k) \in R^o_k \Leftrightarrow \exists i \leq k \colon \forall j \leq k \colon x_i \sqsubseteq x_j$$

To define R^o_k when $o = I$, we should be more cautious as affinity is essentially a concept used for total functions over reals considering their effect on total real numbers. Bearing that in mind we define R^I_k as follows:

Notation 3.6 *When $x \in \mathcal{I}$, by $\underline{x}$ and $\overline{x}$ we mean the left and right end-points of x respectively. More concisely $x = [\underline{x}, \overline{x}]$.*

For any $\overrightarrow{x} = (x_1, \ldots, x_k) \in \mathcal{I}^k$:

$$\overrightarrow{x} \in R^I_k \Leftrightarrow P_1(\overrightarrow{x}) \vee (P_{2a}(\overrightarrow{x}) \wedge P_{2b}(\overrightarrow{x}))$$

where P_1, P_{2a} and P_{2b} are defined as follows:

$$\begin{aligned}
P_1(x_1, \ldots, x_k) &\Leftrightarrow \exists i \in \{1, \ldots, k\} \colon \forall j \in \{1, \ldots, k\} \colon x_i \sqsubseteq x_j \\
P_{2a}(x_1, \ldots, x_k) &\Leftrightarrow \forall i \in \{1, \ldots, k-1\} \colon \overline{x_i} = \underline{x_{i+1}} \\
P_{2b}(x_1, \ldots, x_k) &\Leftrightarrow \exists \overrightarrow{y} = (y_1, \ldots, y_k) \colon (\overrightarrow{y} \sqsupseteq \overrightarrow{x}) \\
&\quad \wedge \ (\forall i \leq k \colon y_i \text{ is maximal in } \mathcal{I}) \\
&\quad \wedge \ [\exists d > 0 \colon (\forall i \in \{2, \ldots, k-2\} \colon \\
&\qquad \text{If none of } x_{i-1}, x_i, x_{i+1}, x_{i+2} \text{ is maximal in } \mathcal{I} \text{ then} \\
&\qquad y_{i+1} - y_i = d)]
\end{aligned}$$

Definition 3.7 $[R_k]$ R_k is the logical relation generated by the above ground type cases.

The aim is to show that R_k is C-logical, where C is the set $w\mathcal{PRC}_{\mathrm{A}}$. As before the tricky part is to show that the set of R_k invariant elements of the ground type I forms an inclusive predicate.

Lemma 3.8 *If $\{x^i = (x^i_1, \ldots, x^i_k) \in R^I_k \mid i \in \mathbb{N}\}$ is an ascending chain then $\overrightarrow{x} = (\bigsqcup^{\uparrow}_{i \in \mathbb{N}} x^i) \in R^I_k$.*

Proof. There are two cases to consider, which might overlap but nevertheless are exhaustive:

case (a) For an infinite subset $\mathbb{N}_1 \subseteq \mathbb{N}$ of natural numbers, we have $\forall i \in \mathbb{N}_1 \colon P_1(x^i)$. In this case for all $i \in \mathbb{N}_1$ there is an index $j(i) \in \{1, \ldots, k\}$ such that $\forall m \in \{1, \ldots, k\} \colon x^i_{j(i)} \sqsubseteq x^i_m$. This implies that there is an index $l \in \{1, \ldots, k\}$ for which there are infinitely many i's with $j(i) = l$. As $\{x^i \mid i \in \mathbb{N}\}$ is ascending we have:

$$\forall m \in \{1, \ldots, k\} \colon \sqcup^{\uparrow}\{x^i_l \mid i \in \mathbb{N}\} \sqsubseteq \sqcup^{\uparrow}\{x^i_m \mid i \in \mathbb{N}\}$$

So $P_1(\sqcup^{\uparrow}\{x^i \mid i \in \mathbb{N}\})$.

case (b) $\exists n_0 \in \mathbb{N}\colon \forall i \geq n_0\colon P_{2a}(x^i) \wedge P_{2b}(x^i)$. In this case for any $i \geq n_0$, there is a vector $y^i = (y^i_1, \ldots, y^i_k)$ and a real number $d_i > 0$ that make $P_{2b}(x^i)$ true. Now take $\overrightarrow{x} = (x_1, \ldots, x_k) \coloneqq \sqcup\{x^i\}$. As $\{x^i\}$ is ascending and for each $i \geq n_o\colon P_{2a}(x^i)$, we have:

$$\forall j \in \{2, \ldots, k-1\}, \forall i \geq n_0\colon x_j = x^i_j$$

therefore, for arbitrary $y_1 \in x_1$ and $y_k \in x_k$ the vector:

$$(y_1, y^{n_0}_2, \ldots, y^{n_0}_{k-1}, y_k)$$

and d_{n_0} will make $P_{2b}(\overrightarrow{x})$ true. To finish we notice that :

$$\forall j \leq k-1\colon \overline{x_j} = \overline{x^{n_0}_j} = \underline{x^{n_0}_{j+1}} = \underline{x_{j+1}}$$

hence $P_{2a}(\overrightarrow{x})$.

□

Lemma 3.9 *R_k as defined in Definition 3.7 is C-logical, where C is the set of wPR constants $w\mathcal{PRC}_{\mathrm{A}}$.*

Proof. We check out the more interesting constants:

(i) $c \in C$ is the constant $\mathsf{tail}_a\colon I \to I$ for some non-maximal $a \in \mathcal{I}$ and $(x_1, \ldots, x_k) \in R^I_k$. There are two cases:

(a) $P_1(x_1, \ldots, x_k)$: then as tail_a is monotone we have

$$P_1(\mathsf{tail}_a(x_1), \ldots, \mathsf{tail}_a(x_k))$$

(b) $P_{2a}(x_1, \ldots, x_k) \wedge P_{2b}(x_1, \ldots, x_k)$: Assume $a = [\underline{a}, \overline{a}]$ and that $\underline{a} \in x_{i_1}$, $\overline{a} \in x_{i_2}$ for some $1 \leq i_1 \leq i_2 \leq k$ (other cases are more or less similar). In this case we have:

$$\forall j < i_1\colon \mathsf{tail}_a(x_j) = [0, 0]$$
$$\forall j > i_2\colon \mathsf{tail}_a(x_j) = [1, 1]$$

If $\overrightarrow{y} = (y_1, \ldots, y_k)$ and d make $P_{2b}(x_1, \ldots, x_k)$ true, then

$$(0, \ldots, \underset{\uparrow}{0}_{i_1}, \mathsf{tail}_a(y_{i_1+1}), \ldots, \mathsf{tail}_a(y_{i_2-1}), \underset{\uparrow}{1}_{i_2}, \ldots, 1)$$

and $\frac{d}{\overline{a}-\underline{a}}$ will make $P_{2b}(\mathsf{tail}_a(x_1), \ldots, \mathsf{tail}_a(x_k))$ true (see equation (3), page 20). $P_{2a}(\mathsf{tail}_a(x_1), \ldots, \mathsf{tail}_a(x_k))$ holds trivially.

(ii) $c \in C$ is the constant $\mathsf{head}_r\colon I \to \mathsf{bool}$ for some $r \in \mathbb{Q} \cap (0, 1)$:
Let $\overrightarrow{x} = (x_1, \ldots, x_k) \in R^I_k$ and consider the two cases:

(a) $P_1(x_1, \ldots, x_k)$: then as head_r is monotone we have

$$P_1(\mathsf{head}_r(x_1), \ldots, \mathsf{head}_r(x_k))$$

(b) $P_{2a}(x_1, \ldots, x_k) \wedge P_{2b}(x_1, \ldots, x_k)$: There are three cases:

- $\overline{x_k} < r$: in this case

$$(\mathsf{head}_r(x_1), \ldots, \mathsf{head}_r(x_k)) = (tt, \ldots, tt) \in R_k^{\mathsf{bool}}$$

- $r < \underline{x_1}$: we have

$$(\mathsf{head}_r(x_1), \ldots, \mathsf{head}_r(x_k)) = (ff, \ldots, ff) \in R_k^{\mathsf{bool}}$$

- $\exists i \leq k\colon r \in x_i$: in this case $\mathsf{head}_r(x_i) = \bot$ so $\forall j \leq k\colon \mathsf{head}_r(x_i) \sqsubseteq \mathsf{head}_r(x_j)$ which implies

$$(\mathsf{head}_r(x_1), \ldots, \mathsf{head}_r(x_k)) \in R_k^{\mathsf{bool}}$$

(iii) $c \in C$ is a fix point constant Y_σ : Using Lemma 3.8, page 33 the proof in this case is by induction over σ as was done before in proposition 2.7, page 26 for S_{k+1}'s.

(iv) $c \in C$ is any other constant : These are straightforward and left to the reader.

□

3.1 Discussion

There are certain issues to be addressed regarding Lemma 3.9. The logical relations R_k do not by any means characterize piecewise affinity, as functions such as $\mathrm{neg}\colon \mathcal{I} \to \mathcal{I}$:

$$\mathrm{neg}([r, s]) \coloneqq [1 - s, 1 - r]$$

which are obviously affine do not preserve all R_k's. On the other hand, non-affine functions are highly unlikely to preserve all R_k's. Take $f\colon \mathcal{I} \to \mathcal{I}$ defined by:

$$f([r, s]) \coloneqq [r^2, s^2]$$

and consider:

$$\overrightarrow{a} = (a_1, \ldots, a_{20}) \in R_{20}^I$$

where

$$\forall 1 \leq i \leq 20\colon a_i = [(i-1)/20, i/20]$$

Under f, this element of R_{20}^I is sent to:

$$\overrightarrow{b} = (b_1, \ldots, b_{20})$$

where

$$\forall 1 \leq i \leq 20 \colon b_i = [(i-1)^2/400, i^2/400]$$

Claim 3.10 $\overrightarrow{b} = (b_1, \ldots, b_{20}) \notin R^I_{20}$

Proof. Take any arbitrary $\overrightarrow{y} = (y_1, \ldots, y_{20})$ such that:

(i) y_i's are maximal in $\mathcal{I}$ $(1 \leq i \leq 20)$

(ii) $\overrightarrow{b} \leq \overrightarrow{y}$

Then we have:

(i) $y_3 - y_2 \leq 8/400$

(ii) $\max\{y_{19} - y_{18}, y_{18} - y_{17}\} > 17/400$

therefore it is impossible to find any $\overrightarrow{y}$ and $d > 0$ such that $P_{2b}(\overrightarrow{b})$. □

Although we picked a special case, it suggests a uniform approach to showing non-affine functions **not preserving** some R_k, which is part of the future work (see Part 4. page 37).

Remark 3.11 None of the following constants:

$$\begin{aligned}
\mathsf{por} &: \mathsf{bool} \times \mathsf{bool} \to \mathsf{bool} \\
\mathsf{pif}_{\mathsf{bool}} &: \mathsf{bool} \times \mathsf{bool} \times \mathsf{bool} \to \mathsf{bool} \\
\mathsf{pif}_{\mathsf{nat}} &: \mathsf{bool} \times \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat} \\
\mathsf{pif}_I &: \mathsf{bool} \times I \times I \to I
\end{aligned}$$

preserves all the logical relations R_k. Take pif_I for example. In the following figure, the left three columns are elements of R_3, whereas the rightmost column — the result of applying pif_I over the elements of the first three — is not:

$$\begin{array}{cccccc}
tt & [0,1/3] & [0,1/4] & \overset{\mathsf{pif}_I}{\to} & [0,1/3] \\
\bot & [1/3,2/3] & [1/4,3/4] & \overset{\mathsf{pif}_I}{\to} & [1/4,3/4] \\
tt & [2/3,1] & [3/4,1] & \overset{\mathsf{pif}_I}{\to} & [2/3,1]
\end{array}$$

hence none of them is definable in wPR. As a minor result, we have another confirmation of the fact that wpor is strictly weaker than por.

Remark 3.12 Take wRPCF$^+$, the extension of wRPCF with a constant $+ \colon I \to I \to I$, interpreted as the maximal continuous extension of the mediation operator (equation (4), page 24). Escardó, Hofmann and Streicher in [EHS98] show that wpor is wRPCF$^+$-definable. On the other hand, it is easy to show that wRPCF$^+$-definable functions preserve all R_k's, which proves that none of the four *parallel* operations mentioned in Remark 3.11 is wRPCF$^+$-definable.

4 Summary of the results and possible future investigations

The results of this paper have a general non-definability flavour, in the sense that we have presented criteria against which functions can be tested to see if they are not definable in certain segments of RPCF. Theorem 2.9 (page 29) assures us that pif_I is the only source of parallelism in RPCF. The logical relations S_k presented in equation (5), page 25 might give an inspiration as to how to characterize the logical relations preserved by wRPCF terms, i.e. a result similar to that of Sieber's for PCF (see [AC98, Exercise 6.5.3, page 136]).

Also we have shown that wRPCF-definable functions are piecewise affine (Definition 3.1, page 30), though our result is slightly more powerful. Of course the language we studied is really weak so it does not come as a surprise that the functions definable in that language are so limited, one witness being their piecewise affinity. The logical relations R_k we presented (Definition 3.7, page 33) by themselves do not characterize piecewise affinity. But as we discussed in Part 3.1, page 35, the whole framework could give an inspiration for further studies, specially of the following problem:

Problem 4.1 *Having R_k's as defined in Definition 3.7, page 33, how can we characterize piecewise affinity?*

Acknowledgement

Achim Jung — my supervisor — suggested the problem of sequentiality of wRPCF and the definition of the generalized Vuillemin-sequentiality, i.e. Definition 2.4, page 24. I like to thank him for his invaluable help throughout all the stages of preparing the paper. Also my thanks go to Martín Escardó for his comments and suggestions, especially on Corollary 2.10, page 29.

References

[AC98] R. Amadio and P.-L. Curien. *Domains and Lambda Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.

[AJ94] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994.

[BCOR86] H.-J. Böhm, R. Cartwright, M. J. O'Donnell, and M. Riggle. Exact real arithmetic: A case study in higher order programming. In *ACM Symposium on Lisp and Functional Programming*. Association of Computing Machinery, 1986.

[Ber79] G. Berry. Modèles Complèment Adéquats et Stables des Lambda-calculs typés, 1979. Thèse de Doctorat d'Etat, Université Paris VII.

[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers. *Bulletin of the American Mathematical Society*, 21:1–46, 1989.

[EHS98] M. H. Escardó, M. Hofmann, and T. Streicher. Mediation is inherently parallel, 1998. University of Edinburgh, Laboratory for Foundations of Computer Science, EPSRC report for project GR/M64840.

[Esc96] Martín Hötzel Escardó. Real pcf extended with ∃ is universal. In A. Edalat, S. Jourdan, and G. McCusker, editors, *Advances in Theory and Formal Methods of Computing: Proceedings of the Third Imperial College Workshop*, pages 13–24, Christ Church, Oxford, April 1996. IC Press.

[Esc97] Martín Hötzel Escardó. *PCF extended with real numbers: a domain-theoretic approach to higher-order exact real number computation*. PhD thesis, The University of Edinburgh, Department of Computer Science, November 1997.

[Far] Amin Farjudian. Conservativity of wrpcf over pcf. Presented in BCTCS 19, Leicester, England, April 7–9, 2003.

[Gia93] Pietro Di Gianantonio. *A Functional Approach to Computability on Real Numbers*. PhD thesis, Università di Pisa-Genova-Udine, 1993.

[JT92] A. Jung and J. Tiuryn. A new characterization of lambda definability. Technical Report 1498, Technische Hochschule Darmstadt, 1992.

[MM96] Valérie Ménissier-Morain. Arbitrary precision real arithmetic: design and algorithms. Submitted to the Journal of Symbolic Computation, September 1996.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Sie92] K. Sieber. Reasoning about sequential functions via logical relations. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Proc. LMS Symposium on Applications of Categories in Computer Science, Durham 1991*, volume 177 of *LMS Lecture Note Series*, pages 258–269. Cambridge University Press, 1992.

[Sto91] A. Stoughton. Interdefinability of parallel operations in PCF. *Theoretical Computer Science*, 79:357–358, 1991.

[Vui74] J. Vuillemin. Syntaxe, sèmantique et axiomatique d'un langage de programmation simple, 1974. Thèse de Doctorat d'Etat, Université Paris VII.