An Abstract Data Type for Real Numbers. [⋆]

Pietro Di Gianantonio

Dipartimento di Matematica e Informatica,
Università di Udine
via delle Scienze 206 I-33100 Udine Italy
e-mail: digianantonio@dimi.uniud.it

Abstract

We present a calculus having real numbers as a basic data type. The calculus is defined by its denotational semantics. We prove the universality of the calculus. We show how the definition of an operational semantics is problematic. We discuss this problem and present a possible solution.

Key words: real number computability, domain theory, denotational and operational semantics, abstract data types.

1 Introduction

The aim of this work is to attempt a connection between two different approaches to computability on real numbers: a practical approach based on programming languages, and a more theoretical one based on domain theory.

Several implementations of exact computations on real numbers have been proposed so far ([4,17,26,19]). In these works, real numbers are represented by programs generating sequences of discrete elements, e.g. digits. Conversely, a variety of theoretical work on computability on real numbers are based on domain theory: [14,15,10,9]. In all these works domains of approximations for real numbers are considered. A point in these domains represents either a real number or the approximation of a real number. Approximated reals are normally described by intervals of the real line.

[⋆] The research on which this work is based was supported by EPSRC grant "Techniques of Real Number Computation", HCM contract "Lambda Calcul Typé", and MURST 40% grant.

The connection between the two approaches is described here in several steps. First we present a domain of approximations which is directly derived from a representation for real numbers used in some implementations of exact real number computation ([4,17]). From this domain of approximations we derive a calculus for real numbers. The calculus we present is an extension of PCF having real numbers as a ground type. We call it $\mathcal{L}_r$. We define $\mathcal{L}_r$ by giving its denotational semantics.

The next obvious step would be to give an operational semantics to the calculus, possibly using the representation for real numbers we had employed at the start. If this were to be possible, we would be able to establish a close connection between the domain of approximations for real numbers and the implementations of real number computations. We would have a calculus which, for many aspects, is similar to the calculi used in the implementations and whose terms could be directly interpreted in the approximation domain. Unfortunately we prove that it is impossible to define the operational semantics in this way. To define an operational semantics for $\mathcal{L}_r$ it is necessary to introduce a new kind of representation for real numbers. This new representation is radically different from all classical ones, since real numbers are here represented also by sequences of digits undefined on some elements. In order to compute with this representation it is absolutely necessary to use parallel operators. In the final section of this paper, we discuss whether parallel computation is necessary in all faithful calculi for real numbers.

2 Basic Definitions

2.1 Real number notation

We consider the following representation for real numbers:

Definition 1 *A real number x is represented by a computable sequence of integers $\langle s_0, \ldots, s_i, \ldots \rangle$ such that:*

(*i*) $\forall n \,.\, 2s_n - 1 \leq s_{n+1} \leq 2s_n + 1$

(*ii*) $x = \bigcap_{n \in \mathbb{N}} \left[\frac{s_n - 1}{2^n}, \frac{s_n + 1}{2^n} \right]$

In this representation, a sequence of integers is used to describe a sequence of rational intervals. Each interval in the sequence is contained into the previous one. For practical purposes, this representation is very convenient, as it allows to reduce exact real number computation to computation on integers. In this way it is possible to exploit the implementation of integer arithmetic already available. In [5] and [16,17] a similar representation has been used to develop quite efficient algorithms for the arithmetic operations.

2.2 The language PCF

From here on, the language PCF will be used as a formalism for expressing and studying computable functions on reals. PCF is a typed lambda calculus having a call-by-name strategy of evaluation, which we have chosen because it is both simple and has been thoroughly investigated. Moreover, the lambda calculus is a paradigm for functional programming languages, hence many problems which are typical of programming languages can be discussed in this setting. For completeness, we give here the basic definition of PCF [18].

2.2.1 Syntax

The set T of type expressions of PCF is defined by the grammar:

$$\sigma ::= \iota \mid o \mid \sigma \rightarrow \tau$$

where σ, τ are metavariables ranging over the set of types, ι and o are the type constants for integers and booleans respectively. Types ι and o are called ground types. Standard PCF has natural numbers as its basic type; for convenience, in this work we substitute natural numbers with integers. PCF does not have a product type constructor. Instead, functions of several arguments can be viewed as functions of a single argument by currying.

The set $\mathcal{L}$ of expressions of PCF is defined by the grammar:

$$M \ ::= \ x^\sigma \mid c^\sigma \mid M^{\sigma\rightarrow\tau}(M^\sigma) \mid (\lambda x^\sigma . M^\tau)$$

where x^σ is a metavariable over a countable set of variables Var_σ of type σ, and c^σ is a metavariable over the set of constants C. In addition the symbols y, α are used here as metavariables over the set of the variables. When no confusion arises, the type superscript σ in the term M^σ will be omitted. Typing rules, free variables, bound variables and closed terms are defined as usual. The application of terms $M(N)$ is understood to be associative to the left. $[N/x]M$ denotes the result of substituting the term N in all free occurrences of x in the term M.

The constants are:

$$\underline{0}\ \ldots\ \underline{i}\ \ldots : \iota \qquad \underline{-1}\ \ldots\ \underline{-i}\ \ldots : \iota$$

$$\mathsf{tt}, \mathsf{ff} : o$$

$$\mathsf{pred}, \mathsf{succ} : \iota \to \iota, \qquad \mathsf{Z} : \iota \to o$$

$$\mathsf{if}_\iota : o \to \iota \to \iota \to \iota, \qquad \mathsf{if}_o : o \to o \to o \to o,$$

$$Y_\sigma : (\sigma \to \sigma) \to \sigma$$

Type assignments and type constraints are defined as usual.

The languages $\mathcal{L}_{PA}$ and $\mathcal{L}_{PA+\exists}$ will also be considered in this paper. $\mathcal{L}_{PA}$ is the language $\mathcal{L}$ extended with the constants $\mathsf{pif}_\iota : o \to \iota \to \iota \to \iota$ and $\mathsf{pif}_o : o \to o \to o \to o$. $\mathcal{L}_{PA+\exists}$ is the language $\mathcal{L}_{PA}$ extended with the constant $\exists : (\iota \to o) \to o$. We will denote with $\mathcal{L}^\sigma$ ($\mathcal{L}^\sigma_{PA}$, $\mathcal{L}^\sigma_{PA+\exists}$) the set of terms of $\mathcal{L}$ ($\mathcal{L}_{PA}$, $\mathcal{L}_{PA+\exists}$) having type σ.

2.2.2 Operational Semantics

The operational semantics is given by an *immediate reduction relation*, $\to$, between terms. It is defined by the following set of reduction rules,
constants:

$$\mathsf{succ}(\underline{i}) \to \underline{i+1} \qquad \mathsf{pred}(\underline{i}) \to \underline{i-1}$$

$$\mathsf{Z}(\underline{i}) \to \mathsf{tt} \text{ if } i \leq 0$$

$$\mathsf{Z}(\underline{i}) \to \mathsf{ff} \text{ if } i > 0$$

conditional:

$$\mathsf{if}_\tau(\mathsf{tt})(M)(N) \to M \qquad \mathsf{if}_\tau(\mathsf{ff})(M)(N) \to N$$

fixed point:

$$Y_\sigma(M) \to M((Y_\sigma)(M))$$

application:

$$(\lambda x.M)(N) \to [N/x]M$$
$$\frac{M \to M'}{M(N) \to M'(N)}$$
$$\frac{N \to N'}{M(N) \to M(N')} \text{ for } M \in \{\mathsf{succ}, \mathsf{pred}, \mathsf{if}_\iota, \mathsf{if}_o, \mathsf{Z}\}$$

parallel test:

$$\frac{M \to M'}{\mathsf{pif}_\tau(P)(M)(N) \to \mathsf{pif}_\tau(P)(M')(N)} \qquad \frac{N \to N}{\mathsf{pif}_\tau(P)(M)(N) \to \mathsf{pif}_\tau(P)(M)(N')}$$

$$\mathsf{pif}_\tau(P)(c)(c) \to c \qquad \mathsf{pif}_\tau(\mathsf{tt})(M)(N) \to M \qquad \mathsf{pif}_\tau(\mathsf{ff})(M)(N) \to N$$

existential :

$$\frac{M(\Omega_\sigma) \overset{*}{\to} \mathsf{ff}}{\exists M \to \mathsf{ff}} \qquad \frac{M(\underline{n}) \overset{*}{\to} \mathsf{tt}}{\exists M \to \mathsf{tt}}$$

where $\overset{*}{\to}$ is the transitive closure of the relation $\to$, and $\Omega_\sigma = Y_\sigma(\lambda\alpha^\sigma.\alpha^\sigma)$ is a term defining a diverging computation.

We define the partial function Eval on programs (closed terms having ground type) as: $\text{Eval}(M) = c$ if $M \overset{*}{\to} c$ for some constant c.

2.2.3 Denotational Semantics

The denotational semantics for $\mathcal{L}$ is given using the set of Scott-domains:

$$UD = \{D_\sigma \mid \sigma \in T\}$$

where $D_\iota = \mathbb{Z}_\perp$, $D_o = \{\mathsf{tt}, \mathsf{ff}\}_\perp$ and $D_{\sigma\to\tau} = [D_\sigma \to D_\tau]$.

The semantic interpretation function $\mathcal{E}$ has the form:

$$\mathcal{E} : \mathcal{L} \to Env \to UD$$

where Env is the set of environments. An environment is a function ρ from Var to UD satisfying the condition $\rho(x^\sigma) \in D_\sigma$.

The definition of $\mathcal{E}$ is given by structural induction,

$$\begin{aligned}
\mathcal{E}[\![c]\!]_\rho &= \mathcal{B}[\![c]\!]\\
\mathcal{E}[\![x^\sigma]\!]_\rho &= \rho(x^\sigma)\\
\mathcal{E}[\![M^{\sigma\to\tau}M^\sigma]\!]_\rho &= \mathcal{E}[\![M^{\sigma\to\tau}]\!]_\rho(\mathcal{E}[\![M^\sigma]\!]_\rho)\\
\mathcal{E}[\![\lambda x^\sigma.M^\tau]\!]_\rho &= \lambda d \in D_\sigma.\mathcal{E}[\![M^\tau]\!]_{(\rho[d/x])}
\end{aligned}$$

The function $\mathcal{B}$ for the interpretation of constants is defined as:

$$\mathcal{B}[\![\underline{n}]\!] = n$$

$$\mathcal{B}[\![\mathsf{succ}]\!](n) = \begin{cases} n+1 & \text{if } n \in \mathbb{Z}\\ \bot & \text{if } n = \bot \end{cases}$$

$$\mathcal{B}[\![\mathsf{pred}]\!](n) = \begin{cases} n-1 & \text{if } n \in \mathbb{Z}\\ \bot & \text{if } n = \bot \end{cases}$$

$$\mathcal{B}[\![\mathsf{Z}]\!](n) = \begin{cases} \mathsf{tt} & \text{if } n \le 0\\ \mathsf{ff} & \text{if } n > 0\\ \bot & \text{if } n = \bot \end{cases}$$

$$\mathcal{B}[\![\mathsf{if}_\tau]\!](b)(x)(y) = \begin{cases} x & \text{if } \; b = \mathsf{tt}\\ y & \text{if } \; b = \mathsf{ff}\\ \bot & \text{if } \; b = \bot \end{cases}$$

$$\mathcal{B}[\![Y_\tau]\!](f) = \bigsqcup_{n \in N} \{f^n \bot_\tau\}$$

$$\mathcal{B}[\![\mathsf{pif}_\tau]\!](b)(x)(y) = \begin{cases} x & \text{if } \; b = \mathsf{tt}\\ y & \text{if } \; b = \mathsf{ff}\\ x \sqcap y & \text{if } \; b = \bot \end{cases}$$

$$\mathcal{B}[\![\exists]\!](g) = \begin{cases} \mathsf{ff} & \text{if } \; f(\bot) = \mathsf{ff}\\ \mathsf{tt} & \text{if } \; \exists n.f(n) = \mathsf{tt}\\ \bot & \quad\text{otherwise} \end{cases}$$

The operational and denotational semantics are related by the following proposition.

Proposition 2 (Adequacy) *For every closed term of ground type M:*

(1) $\mathcal{E}[\![M]\!]_\rho = \mathcal{B}[\![\mathrm{Eval}(M)]\!]$ *if* $\mathrm{Eval}(M)$ *is defined;*
(2) $\mathcal{E}[\![M]\!]_\rho = \bot$ *otherwise.*

Proof See [18].

3 Real number computation in PCF

In order to represent real numbers in PCF it is sufficient to implement in PCF the representation in Definition 1. In what follows, given a type σ, $\mathcal{L}^\sigma_{PA+\exists}$ indicates the set of closed terms in $\mathcal{L}_{PA+\exists}$ having type σ.

Definition 3 *A partial representation function* $\mathrm{Eval}_\mathbb{R} : \mathcal{L}^{\iota\to\iota}_{PA+\exists} \rightharpoonup \mathbb{R}$ *is defined by:* $\mathrm{Eval}_\mathbb{R}(M) = x$ *if there exists a sequence of integers s such that:*

(1) $\forall n \in \mathbb{N}.\mathrm{Eval}(M(\underline{n})) = \underline{s_n}$;
(2) $\forall n\,.\,2s_n - 1 \leq s_{n+1} \leq 2s_n + 1$
(3) $x = \bigcap_{n\in\mathbb{N}}[\frac{s_n-1}{2^n}, \frac{s_n+1}{2^n}]$.

A real number x is said to be computable *if it belongs to the image of the* $\mathrm{Eval}_\mathbb{R}$.

Notation. We indicate with $\mathbb{R}_l$ the set of the computable real numbers.

This definition of computable real number coincides with other definitions in literature, such as [1], [13], [15], [20], [25].

The definition of computability can be extended to functions on real numbers.

Definition 4 *For each natural number n, let τ_n be the type inductively defined as:* $\tau_0 = (\iota \to \iota)$ *and* $\tau_{n+1} = \tau_0 \to \tau_n$. *The function* $\mathrm{Eval}^n_\mathbb{R} : \mathcal{L}^{\tau_n}_{PA+\exists} \to (\mathbb{R}_l)^n \to \mathbb{R}_l)$ *is defined by:*

$$\mathrm{Eval}^n_\mathbb{R}(M) = f \quad \textit{iff} \quad \forall x_1,\ldots,x_n \in \mathbb{R}_l\,.\,\forall N_1,\ldots,N_n \in \mathcal{L}^{\iota\to\iota}_{PA+\exists}\,.$$
$$(\forall i \leq n\,.\,\mathrm{Eval}_\mathbb{R}(N_i) = x_i) \;\Rightarrow\; \mathrm{Eval}_\mathbb{R}(M(N_1)\ldots(N_n)) = f(x_1,\ldots,x_n).$$

*A function $f : (\mathbb{R}_l)^n \to \mathbb{R}_l$ is said to be $\mathcal{L}$-*computable *if it belongs to the image of* $\mathrm{Eval}^{\tau_n}_\mathbb{R}$.

Notation. We indicate with $\mathbb{F}_l^n$ the set of $\mathcal{L}$-computable functions with n arguments.

It is interesting to observe that the parallel operators pif_ι, pif_o and $\exists$ are not necessary in order to define computable functions on reals. This fact can be proved by observing that the two sets of total functions $(\mathbb{Z} \to \mathbb{Z}) \to (\mathbb{Z} \to \mathbb{Z})$ definable in $\mathcal{L}$ and in $\mathcal{L}_{PA+\exists}$ coincide.

The form of computation on real numbers implied by the above definition is similar to the one used in the implementations of exact real number computation which has been described in [4] and in [17].

The above definition of computability is equivalent to the one presented in [6], and it is characterised by the fact that the domain of definition of the computable function is restricted to the computable reals.

In the next section we will present a second definition of computable functions where the domain of definition is the whole real line. The two definitions lead to quite different classes of computable functions. The restriction of the domain of definition of functions to computable reals has some curious consequences. It is a well known result that every computable function on reals is continuous on its domain of definition (w.r.t the Euclidean topology). Now, there exists a computable function defined on all the computable elements of the interval $[0, 1]$, that is continuous and unbounded. This function cannot be continuously extended to a total continuous function [24, p. 309]. To avoid this peculiarity some authors ([3]) give a stronger definition of computability: they require computable functions to be uniformly continuous, with a computable modulus of uniformity.

It is not difficult to extend the notion of computability to arbitrary higher order functions on reals. Here we consider the extension to second order functionals.

Definition 5 *For each n-tuple of natural numbers $\overline{m} = \langle m_1, \dots m_n \rangle$ let $\tau_{\overline{m}}$ be the type $\tau_{m_1} \to \dots \to \tau_{m_n} \to \tau_0$. The function $\mathrm{Eval}_{\mathbb{R}}^{\overline{m}} : \mathcal{L}^{\tau_{\overline{m}}} \to (\mathbb{F}_l^{m_1} \to \dots \to \mathbb{F}_l^{m_n} \to \mathbb{R}_l)$ is defined by:*

$$\mathrm{Eval}_{\mathbb{R}}^{\overline{m}}(M) = F \quad \textit{iff} \quad \forall f_1 \in \mathbb{F}_l^{m_1}, \dots, f_n \in \mathbb{F}_l^{m_n}. \forall N_1 \in \mathcal{L}^{m_1}, \dots, N_n \in \mathcal{L}^{m_n}.$$
$$\forall i \leq n \,.\, \mathrm{Eval}_{\mathbb{R}}^{m_i}(N_i) = f_i \;\Rightarrow\; \mathrm{Eval}_{\mathbb{R}}(M(N_1)\dots(N_n)) = F(f_1, \dots, f_n).$$

A functional $f : \mathbb{F}_l^{m_1} \to \dots \to \mathbb{F}_l^{m_n} \to \mathbb{R}_l$ is $\mathcal{L}$-computable *(*$\mathcal{L}_{PA}$-computable $\mathcal{L}_{PA+\exists}$-computable*) if it belongs to the image, via $\mathrm{Eval}_{\mathbb{R}}^{\tau_n}$, of the set $\mathcal{L}^{\tau_n}$ ($\mathcal{L}_{PA}^{\tau_n}$, $\mathcal{L}_{PA+\exists}^{\tau_n}$)*

It is an open problem whether these three notions of computability coincide. This open problem is connected to a more fundamental open problem for PCF. It is still unknown whether $\mathcal{L}$ and $\mathcal{L}_{PA+\exists}$ define the same set of total functionals on $\mathbb{N}$ ([7]). However, it is possible to show that several, apparently parallel, functionals on real numbers, like integration, are definable in the sequential language $\mathcal{L}$ ([22]).

The definitions given in this section are relative to one particular representation for real numbers. It is not difficult to prove that many other representations for real numbers induce equivalent definitions of computability ([9]). In general, two different representations for real numbers induce the same notion of computability if it is possible to transform, in an effective and uniform way, one representation into the other.

In this section, we have only considered computable functions which are defined on computable reals. By using domain theory, it is possible to consider functions which are defined on the whole real line. This approach is discussed in the following section.

4 A domain of approximations for real numbers

In the literature, several approaches to computability on real numbers can be found which use domain theory. Early works in this ambit are [14], [15], and [21]. In all these approaches the real line is embedded in a space of approximations where a notion of computability can be defined in a natural way. Many results concerning the computability theory on real numbers are given in these contexts. These spaces of approximations are countably based continuous cpos. We are now going to present a space of approximations that is similar in many respects to those mentioned above, but has some important differences. First, we base our construction on the integer sequence representations of Definition 1. As a result, our space has less approximation points and is more closely related to the kind of computation used in some implementations of exact real number arithmetic. A second important difference consists in the fact that our space of approximations is a Scott-domain. The other approaches use spaces of approximations that are continuous but not algebraic cpos. The space of approximations presented here has been extensively studied in [9]. The main results are summed up here; although the proofs are not given.

The domain of approximations defined next is called Reals Domain (RD). We construct RD starting with the integer sequence representation for real numbers. Let $\langle s_i \rangle_{i \in \mathbb{N}}$ be a sequence of integers defining a real number x according to Definition 1 and let $\langle s_i \rangle_{i<n}$ be an initial subsequence. $\langle s_i \rangle_{i<n}$ gives partial information about the value x. By examining $\langle s_i \rangle_{i<n}$, we can deduce that the value x is contained in an interval of real numbers. This observation leads to the definition of a function from finite sequences of integers to intervals in the real line. To any finite sequence $\langle s_i \rangle_{i<n}$ we associate the interval $[a, b]$ containing the real numbers that can be represented by sequences having as initial subsequences $\langle s_i \rangle_{i<n}$. The interval $[a, b]$ represents the information contained in the sequence $\langle s_i \rangle_{i<n}$.

Definition 6 *Let S be the set of sequences of integers defined by:*

$$S = \{\langle s_i \rangle_{i<n} \mid n \in \mathbb{N}, \forall i < n - 1 \,.\, 2s_n - 1 \leq s_{n+1} \leq 2s_n + 1\}.$$

Let RI denote the set of rational intervals. The function ϕ from the $S \to RI$ is defined by,

$$\phi(\langle s_0, s_1, \ldots, s_n \rangle) = [\frac{s_i - 1}{2^i}, \frac{s_i + 1}{2^i}]$$

Let $(DI, \sqsubseteq)$ denote the partial order formed by the rational intervals in the image of ϕ. The order relation $\sqsubseteq$ on DI is the superset relation, that is $[a, b] \sqsubseteq [a', b']$ if and only if $[a', b'] \subseteq [a, b]$ (if and only if $[a'b']$ is a more precise approximation of a real number that $[a, b]$). Let RD denote the cpo obtained by the ideal completion of $(DI, \sqsubseteq)$.

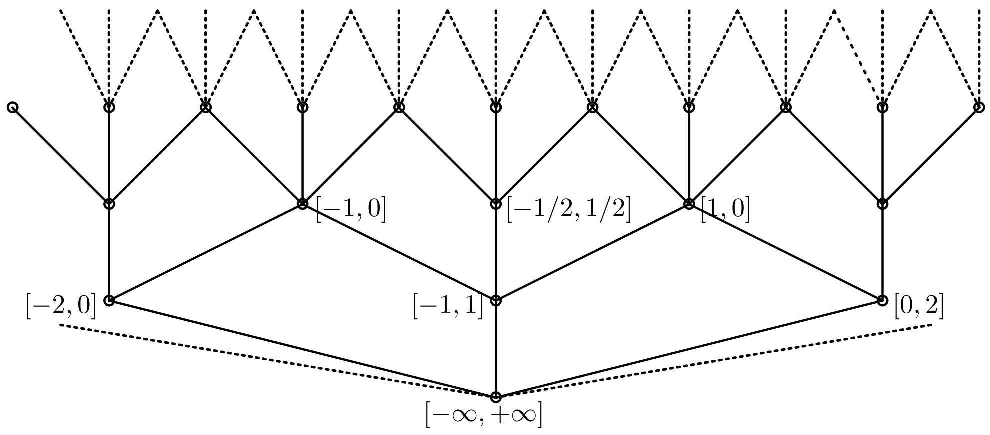


Fig. 1. The diagram representing DI.

Notation. Given a partial order $\langle D, \sqsubseteq \rangle$ and an element $d \in D$ we denote the set $\{d' \mid d' \sqsubseteq d\}$ by $\downarrow d$. Obviously, $\downarrow d$ is a (principal) ideal.

Proposition 7 *RD is a consistently complete ω-algebraic cpo (Scott-domain). RD is an effective Scott-domain when we consider the following enumeration* *of finite elements:*

$$\begin{aligned}\epsilon_r(0) &= \bot \\ \epsilon_r(\langle\langle n_1, n_2\rangle, n_3\rangle + 1) &= \downarrow [(n_1 - n_2 - 1)/2^{n_3}, (n_1 - n_2 + 1)/2^{n_3}]\end{aligned}$$

where $\langle\ \rangle$ is an effective coding function for pairs of natural numbers.

The elements of RD can be thought as equivalence classes of (partial) sequences of integers. Each equivalence class is composed by sequences containing identical information about the real value they approximate.

The relationship between the real line and the infinite elements of RD can be clarified by means of the following functions:

Definition 8 *A function $q_{\mathcal{P}} : RD \to \mathcal{P}(\mathbb{R})$ is defined by:*

$$q_{\mathcal{P}}(d) = \bigcap_{[a,b] \in d} [a, b]$$

Conversely, three functions $e, e^-, e^+ : \mathbb{R} \to RD$ are defined by:

$$\begin{aligned}e(x) &= \{[a, b] \in DI \mid x \in (a, b)\} \\ e^-(x) &= \{[a, b] \in DI \mid x \in (a, b]\} \\ e^+(x) &= \{[a, b] \in DI \mid x \in [a, b)\}\end{aligned}$$

where (a, b) denotes the open interval from a to b and $(a, b]$ and $[a, b)$ indicate the obvious half open, half closed intervals.

Recall that a dyadic number is a rational number in the form $z/2^n$ with $n \in \mathbb{N}$, $z \in \mathbb{Z}$.

Proposition 9 *The following statements hold:*

(1) for every infinite element $d \in RD$ there exists a real number x such that $q_{\mathcal{P}}(d) = \{x\}$
(2) for every real number x, $\{x\} = q_{\mathcal{P}} \circ e(x) = q_{\mathcal{P}} \circ e^-(x) = q_{\mathcal{P}} \circ e^-(x)$,
(3) for every non-dyadic number x, $e(x) = e^-(x) = e^+(x)$,
(4) for every dyadic number x, $e(x) \sqsubset e^-(x)$, $e(x) \sqsubset e^+(x)$ and $e^-(x)$ is not consistent with $e^+(x)$,
(5) $e(\mathbb{R}) \cup e^-(\mathbb{R}) \cup e^+(\mathbb{R})$ is equal to the set of infinite elements of RD.

We can observe that the infinite elements of RD are a close representation of the real line, and that the set of infinite elements in RD is similar to the real line except that each dyadic number is tripled.

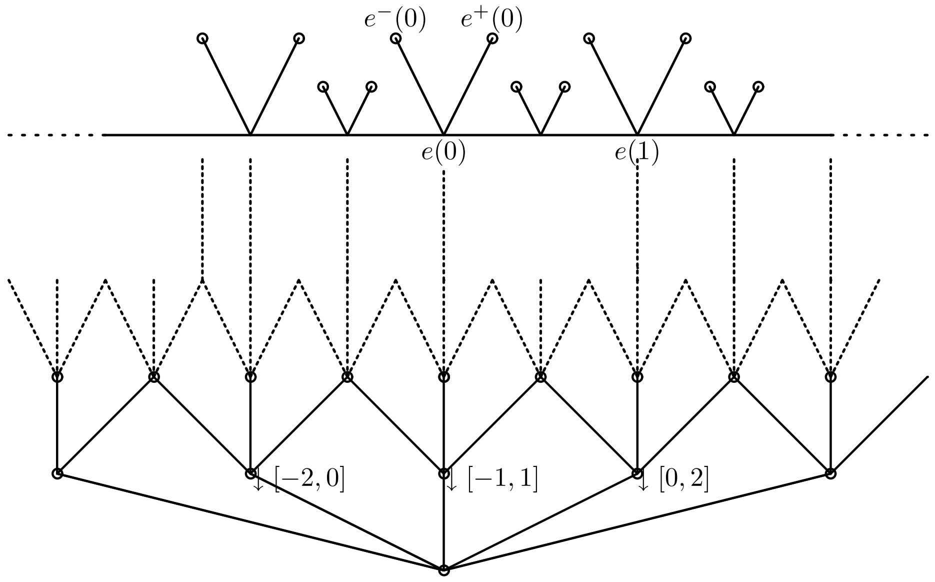


Fig. 2. The diagram representing RD.

A closer connection between the infinite elements of RD and the real line can be established by using topological notions. Let $RD^\dagger$ denote the subspace of RD consisting of the infinite elements with the subspace Scott-topology on RD.

Proposition 10 *The real line is a retract of* $RD^\dagger$ *via a pair of continuous functions* $q : RD^\dagger \to \mathbb{R}$ *and* $e : \mathbb{R} \to RD^\dagger$ *with*

$$q(d) = x \;\; \textit{iff} \;\; q_{\mathcal{P}}(d) = \{x\} \; (= \bigcap_{[a,b]\in d} [a,b])$$
$$e(x) = \{[a,b] \in DI \mid x \in (a,b)\}$$

The function q associates to each element of $RD^\dagger$ the corresponding real number. We can interpret e as the function which picks a canonical representative for each real number. Using q it is possible to give a definition of a computable real number:

Definition 11 *A real number* x *is* computable *if there is a computable element* $d \in RD$ *such that* $x = q(d)$.

The above definition is equivalent to Definition 3. Using e and q it is possible to associate to each Scott-continuous function $f : RD \to RD$ a partial real function $\overline{f} : \mathbb{R} \to \mathbb{R}$ defined by $\overline{f} = q \circ f \circ e$.

This construction can be extended to functions with several arguments. The relation existing between functions on R and functions on RD can be stated in terms of the retraction.

Definition 12 *For each natural number n,*

(1) the topological space $\mathbb{F}_n$ is defined by:

$$\mathbb{F}_n = \{ f : \mathbb{R}^n \to \mathbb{R} \mid f \textit{ total continuous function } \}$$

where $\mathbb{R}^n$ denotes the usual topological product of $\mathbb{R}$.
The topology on $\mathbb{F}_n$ is the compact-open topology.

(2) FD_n is the effective Scott-domain of the Scott-continuous functions $[RD^n \to RD]$.

(3) $FD_n^\dagger$ is the subspace of FD_n defined by:

$$\{ g \in FD_n \mid g((RD^\dagger)^n) \subseteq RD^\dagger \}$$

The topology on $FD_n^\dagger$ is the subspace topology of the Scott-topology.

Observe that FD_0 is homeomorphic to RD.

Not every element in RD denotes a real number: some elements are just finite approximations of real numbers. Similarly, not every function in FD_n represents a function in $\mathbb{F}_n$. For this reason, we have chosen to define the subspaces $FD_n^\dagger$; within $FD_n^\dagger$ every element denotes an element in $\mathbb{F}_n$.

Notation. In this paper, an n-tuple $\langle y_0, \ldots, y_{n-1} \rangle$ is denoted also by $\overline{y}$. If f is a function on the elements of a tuple $\overline{y}$, $\overline{f(y)}$ denotes its pointwise application $\langle f(y_1), \ldots, f(y_{n-1}) \rangle$. The symbols $[a,b]$, $[a',b']$, $[a_i,b_i]$ ... are reserved for intervals in DI. An interval $[a_i,b_i]$ is denoted also by $[a,b]_i$ and, finally, if $\overline{[a,b]}$ is an n-tuple of dyadic intervals, $\prod_{i<n}[a,b]_i$ denotes the obvious subset of $\mathbb{R}^n$.

Proposition 13 *For each natural number n, $\mathbb{F}_n$ is a retract of $FD_n^\dagger$. The pair of retract functions $q_n : FD_n^\dagger \to \mathbb{F}_n$ and $e_n : \mathbb{F}_n \to FD_n^\dagger$ are defined as follows:*

$$q_n(g)(\overline{x}) = q(g(\overline{e(x)}))$$

$$e_n(f)(\overline{d}) = \begin{cases} \bot & \textit{if } \exists i < n \,.\, d_i = \bot \\ \bigsqcup \{ [a',b'] \mid \exists \overline{\downarrow [a,b]} \sqsubseteq \overline{d} \,.\, f(\prod_{i<n}[a,b]_i) \subseteq (a',b') \} & \textit{otherwise} \end{cases}$$

The functions e_n and q_n defined above are the natural generalisation of the functions e and q. In fact, q_n associates to each element of $FD_n^\dagger$ the element of $\mathbb{F}_n$ which is represented by it. And e_n chooses, for each element in $\mathbb{F}_n$, a canonical representation in $FD_n^\dagger$. We can also say that the function q_n partitions $FD_n^\dagger$ into equivalence classes. All the elements contained in a single equivalence class represent the same element in $\mathbb{F}_n$. The function e_n defines a canonical representation for each class.

Definition 14 *For each natural number n, a function $f \in \mathbb{F}_n$ is* domain-computable *if there is a computable element $g \in FD_n^\dagger$ such that $f = q_n(g)$.*

It follows that every computable function on real numbers is continuous w.r.t. the Euclidean topology.

We now consider second order functionals. In this case however it is only possible to state a set theoretic relation between functionals on RD and functionals of R.

Definition 15 *For each n-tuple of natural numbers $\overline{m} = \langle m_1, \ldots m_n \rangle$*

(1) the set of functionals on reals $I\!F_{\overline{m}}$ is defined by:

$$I\!F_{\overline{m}} = \{f : (\mathbb{F}_{m_1} \times \ldots \times \mathbb{F}_{m_n}) \to \mathbb{R}) \mid f \textit{ total continuous function } \}$$

(2) $FD_{\overline{m}}$ is the effective Scott-domain of the Scott-continuous functions $[(FD_{m_1} \times \ldots \times FD_{m_n}) \to RD]$
(3) $FD_{\overline{m}}^\dagger = \{g \in FD_{\overline{m}} \mid g(FD_{m_1}^\dagger \times \ldots \times FD_{m_n}^\dagger) \subseteq RD^\dagger\}$

Using the retract constructions for the first-order functions it is possible to associate to each second order functional in $FD_{\overline{m}}^\dagger$ the functional on the reals represented by it.

Definition 16 *(1) For every type tuple of natural numbers $\overline{m}$ let $q_{\overline{m}}$ be the function from $FD_{\overline{m}}^\dagger$ to $I\!F_{\overline{m}}$ defined by:*

$$q_{\overline{m}}(G)(f_1, \ldots, f_n) = q(G(e_{m_1}(f_1), \ldots, e_{m_n}(f_n)))$$

(2) A functional on real numbers $F \in I\!F_{\overline{m}}$ is computable *if there exists a computable element $G \in FD_{\overline{m}}$ such that $F = q_{\overline{m}}(G)$.*

5 PCF extended with real numbers

In this section we employ the domain RD to define an extension of the language PCF having a ground data type for the real numbers. We call this extension $\mathcal{L}_r$, and denote with r the type for real numbers. In $\mathcal{L}_r$, expressions having type r represent elements in RD. We want to prove that any computable function on RD is definable by a suitable expression in $\mathcal{L}_r$. A programming language very similar to $\mathcal{L}_r$ has first been introduced in [8]. An extension of PCF based on a different domain of approximation for real numbers has also been presented in [12].

Compared with the real computation described in Section 3, the real computation in $\mathcal{L}_r$ has several advantages. Given a closed term $M \in \mathcal{L}^{(\iota\to\iota)\to(\iota\to\iota)}$, the value $\mathrm{Eval}_{\mathbb{R}}(M)^1$ may prove to be undefined; for example:
(i) there may be a term N representing a real number such that the sequence $M(N)(\underline{0}), \ldots, M(N)(\underline{n}), \ldots$ does not define a real number.
(ii) there may be two terms N_1 and N_2 defining the same real number and such that $M(N_1)$ and $M(N_2)$ define different real numbers.

The language $\mathcal{L}_r$ is free from these inadequacies. Terms of type r in $\mathcal{L}_r$ can always be interpreted as (approximated) reals; more importantly, terms of type $r \to r$ preserve the equivalence between different representations of the same real number. We can say, therefore, that $\mathcal{L}_r$ defines an abstract data type for real numbers, i.e., it defines a collection of primitive functions on reals which generate any other computable function.

In this section, we give a denotational semantics to $\mathcal{L}_r$. The attribution of an operational semantics to $\mathcal{L}_r$ presents a numbers of problems, which will be discussed in the next sections.

5.1 Syntax

The types of $\mathcal{L}_r$ are the PCF types extended with a new ground r. The set T of type expressions is defined by the grammar:

$$\sigma ::= \iota \mid o \mid r \mid \sigma_1 \to \sigma_2$$

The terms of $\mathcal{L}_r$ are the terms of $\mathcal{L}_{PA+\exists}$ extended with the new constants:

$$(-1),\ (+1),\ (\times 2),\ (\div 2),\ \mathsf{PR}\ :\ r \to r,$$
$$(\leq 0)\ :\ r \to o$$
$$\mathsf{pif}_r\ :\ o \to r \to r \to r,$$
$$Y_\sigma\ :\ (\sigma \to \sigma) \to \sigma \qquad \text{for each new type } \sigma$$

5.2 Semantics

The denotational semantics for $\mathcal{L}_r$ is given using the set of Scott-domains $UD := \{D_\sigma \mid \sigma \in T\}$ where $D_r = RD$, $D_\iota = \mathbb{Z}_\perp$, $D_o = \{\mathsf{tt}, \mathsf{ff}\}_\perp$ and $D_{\sigma\to\tau} = [D_\sigma \to D_\tau]$.

The denotation of the new constants is:

The constants $(+1), (-1), (\times 2), (\div 2)$ realize the corresponding functions on reals.

$$\begin{aligned}
\mathcal{B}[\![(+1)]\!](d) &= \{[a+1, b+1] \mid [a,b] \in d\} \\
\mathcal{B}[\![(-1)]\!](d) &= \{[a-1, b-1] \mid [a,b] \in d\} \\
\mathcal{B}[\![(\times 2)]\!](d) &= \{[a \times 2, b \times 2] \mid [a,b] \in d \ \wedge \ [a \times 2, b \times 2] \in DI\} \\
\mathcal{B}[\![(\div 2)]\!](d) &= \textstyle\bigcup_{[a,b]\in d} \downarrow [a \div 2, b \div 2]
\end{aligned}$$

The constant (≤ 0) tests if a number is smaller or larger than 0.

$$\mathcal{B}[\![(\leq 0)]\!](d) = \begin{cases} \mathsf{tt} & \text{if it exists } [a,b] \in d, \ b \leq 0 \\ \mathsf{ff} & \text{if it exists } [a,b] \in d, \ 0 \leq a \\ \bot & \text{otherwise} \end{cases}$$

The constant PR defines a kind of projection on the interval $[-1, 1]$.

$$\mathcal{B}[\![\mathsf{PR}]\!](d) = \begin{cases} d \sqcup \downarrow [-1,1] & \text{if } d \text{ is consistent with } \downarrow [-1,1] \\ e^{+}(-1) & \text{if } \exists [a,b] \in d . b \leq -1 \\ e^{-}(1) & \text{if } \exists [a,b] \in d . a \geq 1 \end{cases}$$

The constant pif_r defines a parallel test.

$$\mathcal{B}[\![\mathsf{pif}_r]\!](b)(d)(d') := \begin{cases} d & \text{if } b = \mathsf{tt} \\ d' & \text{if } b = \mathsf{ff} \\ d \sqcap d' & \text{if } b = \bot \end{cases}$$

If the boolean argument is undefined, the function $\mathcal{B}[\![\mathsf{pif}_r]\!]$ gives as an output the most precise approximation of the second and third argument.

The constants Y_σ are the usual fixed point operators.

It is not difficult to prove that:

Proposition 17 *For every closed expression M^σ and environment ρ, $\mathcal{E}[\![M^\sigma]\!]_\rho$ is a computable element of D_σ.*

Next we prove the universality of $\mathcal{L}_r$, i.e., we prove that every computable function on RD is definable by a suitable term in $\mathcal{L}_r$. In order to do this we present a generalisation of the universality theorem for PCF [18, Theorem 5.1]. An equivalent generalisation had already been given in [23], although there are some important differences. The proof we give here follows the line of the original proof in [18]; this however can only be applied to extensions of PCF where ground types are denoted by flat domains. We have modified some parts of that proof to make it applicable to extensions of PCF where ground types are denoted by coherent domains. The proof given in [23] is more abstract but it uses, as a lemma, the theorem in [18], and therefore it is not independent from it.

Some definitions and lemmata are necessary here.

Definition 18 *A subset A of a p.o. P is* coherent *if any pair of elements has an upper bound. A* coherent domain *is a Scott-domain for which any coherent subset has an upper bound.*

Coherent domains are closed under many of the semantics functors. In particular:

Proposition 19 *(1) If D_1 and D_2 are coherent domains then $D_1 \to D_2$ is a coherent domain.*
(2) RD is a coherent domain.

Proof. (i) This is a standard proposition of domain theory.
(ii) This follows from the fact that for any finite set of intervals C the intersection of the intervals in C is empty if and only if there exist two intervals in C having an empty intersection. $\square$

A fundamental step in the proof of universality consists in showing that for every type σ it is possible to define three functions, namely c_σ, p_σ and $\#_\sigma$. Here c_σ and p_σ are respectively a test and a projection function for the types σ, and $\#_\sigma(n)(d)$ checks if the element d is inconsistent with the finite element $\epsilon_\sigma(n)$ (where ϵ_σ is the effective enumeration of the finite elements of the domain D_σ ([18, page 249])). Formally:

Definition 20 *A partial function $f : D_{\sigma_1} \to \ldots D_{\sigma_n} \rightharpoonup D_\sigma$ is* definable *in $\mathcal{L}_r$ if there exists a closed term M such that for all $d_1 \in D_{\sigma_1} \ldots d_n \in D_{\sigma_n}$ if $f(d_1)\ldots(d_n)$ is defined then $\mathcal{E}[\![M]\!]_\rho(d_1)\ldots(d_n) = f(d_1)\ldots(d_n)$.*

Definition 21 *Given a coherent-domain D_σ, the function $c_\sigma : B_\perp \to D_\sigma \to D_\sigma \to D_\sigma$, and the partial functions $\#_\sigma : \mathbb{Z}_\perp \to D_\sigma \rightharpoonup B_\perp$, $p_\sigma : \mathbb{Z}_\perp \to D_\sigma \rightharpoonup$*

D_σ are defined by,

$$c_\sigma(b)(d_1)(d_2) = \begin{cases} d_1 & if\ b = \mathsf{tt} \\ d_2 & if\ b = \mathsf{ff} \\ d_1 \sqcap d_2 & if\ b = \bot \end{cases}$$

$$\#_\sigma(n)(d) = \begin{cases} \mathsf{ff} & \text{if } n \in \mathbb{N}, \epsilon_\sigma(n) \sqsubseteq d \\ \mathsf{tt} & \text{if } n \in \mathbb{N} \text{ and the elements } \epsilon_\sigma(n), d \text{ are inconsistent} \\ \text{undefined} & \text{if } n \text{ is a negative number} \\ \bot & \text{otherwise} \end{cases}$$

$$p_\sigma(n)(d) = \begin{cases} d \sqcup \epsilon_\sigma(n) & \text{if } n \in \mathbb{N} \text{ and the elements } \epsilon_\sigma(n), d \text{ are consistent} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Since the domain D_σ is coherent, the function p_σ can be extended to the whole domain. The functions p_σ and $\#_\sigma$ are defined as a partial function for convenience. In the proof it is shown that for every type σ there exist terms P_σ and T_σ whose denotations behave like p_σ and $\#_\sigma$ on their domains of definition. There is no interest in the behaviour of P_σ and T_σ outside these domains.

Lemma 22 *If, in a language extending $\mathcal{L}_{PA+\exists}$, for every ground type τ the function $c_\tau, p_\tau, \#_\tau$ are definable by some terms $\mathsf{pif}_\tau, P_\tau, T_\tau$, then for any other type σ the functions $c_\sigma, p_\sigma, t_\sigma$ are definable by some suitable terms $\mathsf{pif}_\sigma, P_\sigma, T_\sigma$.*

Proof. By structural induction on the type σ. The basic step is true by hypothesis. Let $\sigma = \sigma_1 \to \sigma_2$. The terms pif_σ and P_σ can be defined as follows,

$\mathsf{pif}_\sigma = \lambda x.\lambda\alpha_1^\sigma.\lambda\alpha_2^\sigma.\lambda\beta^{\sigma_1}.\mathsf{pif}_{\sigma_2} x\ \mathsf{then}\ \alpha_1^\sigma\beta^{\sigma_1}\ \mathsf{else}\ \alpha_2^\sigma\beta^{\sigma_1}$

```
$P_\sigma = \lambda m\,.\,\lambda\alpha^\sigma\,.\,\lambda\beta^{\sigma_1}\,.$
        $Y_{\iota\to\sigma_2}(\lambda\gamma^{\iota\to\sigma_2}\,.\,\lambda n\,.$
              $\mathsf{pif}_{\sigma_2}\mathsf{Z}n$
                  $\mathsf{then}\ \alpha^\sigma\beta^{\sigma_1}$
                  $\mathsf{else}\ \mathsf{pif}_{\sigma_2}T_{\sigma_1}(\mathsf{FIRST}_\sigma\ m\,n)\beta^{\sigma_1}$
                          $\mathsf{then}\ \gamma^{\iota\to\sigma_2}(\mathsf{pred}(n))$
                          $\mathsf{else}\ P_{\sigma_2}(\mathsf{SECOND}_\sigma\ m\,n)(\gamma^{\iota\to\sigma_2}(\mathsf{pred}(n))))$
        $(\mathsf{SIZE}_\sigma\ m)$
```

where FIRST_σ, SECOND_σ and SIZE_σ define respectively three primitive recursive functions f, g and h such that for each natural number m:

$$\epsilon_\sigma(m) = \bigsqcup\{\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n)) \mid 0 < n \leq h(m)\}$$

here we denote with $\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n))$ the step-function in D_σ defined by: $(\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n)))(d) = \epsilon_{\sigma_2}(g(m,n))$ if $\epsilon_{\sigma_1}(f(m,n)) \sqsubseteq d$ and $(\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n)))(d) = \perp$ otherwise.

The idea behind the definition of P_σ is the following, the projection of a function α^σ on the function $d = \bigsqcup\{\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n)) \mid 0 < n \leq l\}$ is calculated pointwise. Given an argument β^{σ_1}, it is checked if β^{σ_1} is consistent with $\epsilon_{\sigma_1}(f(m,h))$. If β^{σ_1} is inconsistent with $\epsilon_{\sigma_1}(f(m,h))$ then the result is the projection, d_p, of $\alpha^\sigma(\beta^{\sigma_1})$ on $d'(\beta^{\sigma_1})$ where $d' = \bigsqcup\{\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n)) \mid 0 < n \leq l-1\}$. If $\beta^{\sigma_1} \sqsupseteq \epsilon_{\sigma_1}(f(m,h))$ then the result is the projection of d_p on the element $\epsilon_{\sigma_2}(g(m,n))$. If the consistency cannot be decided then the result is the g.l.b. of the results given in the two previous cases.

The term T_σ is defined by:

```
$T_\sigma = \lambda m.\lambda\alpha^\sigma.Y_{\iota\to o}(\lambda\gamma^{\iota\to o}.\lambda n.$
      $\mathsf{if}_o \mathsf{Z} n$
        $\mathsf{then\ ff}$
        $\mathsf{else\ pif}_o \exists(\lambda l.T_{\sigma^2}(\mathsf{SECOND}_\sigma\, m\, n)(\alpha^\sigma(P_{\sigma_1}(\mathsf{FIRST}_\sigma\, m\, n)(P_{\sigma_1}\, l\, \Omega_{\sigma_1}))))$
              $\mathsf{then\ tt}$
              $\mathsf{else}\ \gamma^{\iota\to o}(\mathsf{pred}(n)))$
      $(\mathsf{SIZE}_\sigma\, m)$
```

In order to check if a function α^σ is inconsistent with a function $d = \bigsqcup\{\epsilon_{\sigma_1}(f(m,n)) \Rightarrow \epsilon_{\sigma_2}(g(m,n)) \mid 0 < n \leq l\}$, it is sufficient to check if there exists $n \leq l$, and a finite element β, $\beta \sqsupseteq \epsilon_{\sigma_1}(f(m,n))$, such that $\alpha(\beta)$ is inconsistent with $\epsilon_{\sigma_2}(g(m,n))$. Observe that $\mathcal{E}[\![P_\sigma \underline{n}(P_\sigma\, \Omega_\iota\, \Omega_\sigma)]\!] = \epsilon_\sigma(n)$ and that the function $\mathcal{E}[\![\lambda l.P_\sigma \underline{n}(P_\sigma\, l\, \Omega_\sigma)]\!]$ enumerates the finite elements above $\epsilon_\sigma(n)$. $\square$

Lemma 23 *If, in an extension of the language $\mathcal{L}$, for a type σ the function p_σ is definable, then every computable element in D_σ is definable.*

Proof. Given a computable element d in D_σ, let f be a primitive recursive function such that

$$x = \bigsqcup\{\epsilon_\sigma(f(i)) \mid i \in N\}$$

and let F be a term defining the function f and P_σ the term defining the function p_σ. We have:

$$d = \mathcal{E}[\![Y_{\iota\to\sigma}(\lambda\alpha^{\iota\to\sigma}\,.\,\lambda n\,.\,P_\sigma(Fn)(\alpha^{\iota\to\sigma}(\mathsf{succ}(n))))\underline{0}]\!]_\rho$$

□

Theorem 24 *For every computable element d in D_σ there exists a closed expression M in $\mathcal{L}_r$, such that: $\mathcal{E}[\![M]\!]_\rho = d$.*

Proof. By the two previous lemmas, it is sufficient to prove that the functions: $c_\tau, p_\tau, \#_\tau$ are definable in the language when τ is a ground type. In [18], it has already been shown that these functions are definable when τ is equal to ι and o. Therefore, we only need to prove the definability for the basic type r. The function c_r is defined by the constant pif_r. To simplify the proof we define the terms P_r and T_r recursively by cases. The actual PCF terms can be obtained straightforwardly. The terms P_r and T_r are defined as:

$P_r(n)(x) = x$ for $n \leq 0$

$P_r(\langle\langle n_1+1, n_2+1\rangle, n_3\rangle + 1)(x) = P_r(\langle\langle n_1, n_2\rangle, n_3\rangle + 1)(x)$

$P_r(\langle\langle n_1+2^{n_3}, 0\rangle, n_3\rangle + 1)(x) = (+1)(P_r(\langle\langle n_1, 0\rangle, n_3\rangle + 1)((-1)((x))))$

$P_r(\langle\langle 0, n_2+2^{n_3}\rangle, n_3\rangle + 1)(x) = (-1)(P_r(\langle\langle 0, n_2\rangle, n_3\rangle + 1)((+1)((x))))$

$P_r(\langle\langle n_1, 0\rangle, n_3+1\rangle + 1)(x)$
$= (\div 2)(P_r(\langle\langle n_1, 0\rangle, n_3\rangle + 1)((\times 2)((+1)(\mathsf{PR}((-1)(\mathsf{PR}(x)))))))$
with $0 < n_1 < 2^{n_3}$

$P_r(\langle\langle 0, n_2\rangle, n_3+1\rangle + 1)(x)$
$= (\div 2)(P_r(\langle\langle 0, n_2\rangle, n_3\rangle + 1)((\times 2)((-1)(\mathsf{PR}((+1)(\mathsf{PR}(x)))))))$
with $0 < n_2 < 2^{n_3}$

$P_r(\langle\langle 0, 0\rangle, n_3+1\rangle + 1)(x) = (\div 2)(P_r(\langle\langle 0, 0\rangle, n_3\rangle + 1)((\times 2)(\mathsf{PR}(x))))$

$P_r(\langle\langle 0, 0\rangle, 0\rangle + 1)(x) = \mathsf{PR}(x)$

$T_r(n)(x) = \mathsf{ff}$ with $n \leq 0$

$T_r(\langle\langle n_1+1, n_2+1\rangle, n_3\rangle + 1)(x) = T_r(\langle\langle n_1, n_2\rangle, n_3\rangle + 1)(x)$

$T_r(\langle\langle n_1+2^{n_3}, 0\rangle, n_3\rangle + 1)(x) = T_r(\langle\langle n_1, 0\rangle, n_3\rangle + 1)((-1)((x)))$

$T_r(\langle\langle 0, n_2+2^{n_3}\rangle, n_3\rangle + 1)(x) = T_r(\langle\langle 0, n_2\rangle, n_3\rangle + 1)((+1)((x)))$

$T_r(\langle\langle n_1, 0\rangle, n_3+1\rangle + 1)(x)$
$= \mathsf{pif}_o(\leq 0)(x)$
then tt

```
        else $\mathsf{pif}_o(\leq 0)((-1)(x))$
              then $T_r(\langle\langle n_1, 0\rangle, n_3\rangle + 1)((\times 2)(x))$
              else $\mathsf{tt}$
  with $0 < n_1 < 2^{n_3}$

$T_r(\langle\langle 0, n_2\rangle, n_3 + 1\rangle + 1)(x)$
  $= \mathsf{pif}_o(\leq 0)(x)$
        then $\mathsf{pif}_o(\leq 0)((+1)(x))$
               then $\mathsf{tt}$
               else $T_r(\langle\langle 0, n_2\rangle, n_3\rangle + 1)((\times 2)(x))$
        else $\mathsf{tt}$
  with $0 < n_2 < 2^{n_3}$

$T_r(\langle\langle 0, 0\rangle, n_3 + 1\rangle + 1)(x)$
  $= \mathsf{pif}_o(\leq 0)((+1)(x))$
        then $\mathsf{tt}$
        else $\mathsf{pif}_o(\leq 0)((-1)(x))$
              then $T_r(\langle\langle n_1, 0\rangle, n_3\rangle + 1)((\times 2)(x))$
              else $\mathsf{tt}$

$T_r(\langle\langle 0, 0\rangle, 0\rangle + 1)(x)$
  $= \mathsf{pif}_o(\leq 0)(+1)(x)$
        then $\mathsf{tt}$
        else $\mathsf{pif}_o(\leq 0)((-1)(x))$
              then $\mathsf{ff}$
              else $\mathsf{tt}$
```

It is a lengthy but straightforward proof to check that the definitions of P_r and T_r consider all possible cases and that the given results are correct. □

In the domain RD every dyadic rational number has three representations, among which computable functions on RD are able to discriminate. Since the language $\mathcal{L}_r$ is universal w.r.t RD, also the functions definable in $\mathcal{L}_r$ can distinguish among the three representations of a dyadic number. The constant (≤ 0), which discriminates among the three representations of the number 0, is an example to the point. To avoid this behaviour, we present a second language, called $\mathcal{L}_{wr}$, which is expressive enough to define all computable functions on reals, but whose functions do not distinguish among the three different representations of the dyadic numbers. The only difference among $\mathcal{L}_{wr}$ and $\mathcal{L}_r$ is the presence of the constant (≤ 0). In $\mathcal{L}_{wr}$ the constant (≤ 0) is substituted by a new constant (< 0). The denotational semantics of (< 0) is:

$$\mathcal{B}[\![(<0)]\!](d) := \begin{cases} \mathsf{tt} & \text{if there exists } [a,b] \in d,\ b < 0 \\ \mathsf{ff} & \text{if there exists } [a,b] \in d,\ 0 < a \\ \perp & \text{otherwise} \end{cases}$$

The function $\mathcal{B}[\![(<0)]\!]$ is the greatest approximation of the function $\mathcal{B}[\![(\leq 0)]\!]$ (w.r.t. the domain order) which does not distinguish among the three representations of 0 in RD.

In order to prove that terms of $\mathcal{L}_{wr}$ do not distinguish among the different representations of dyadic numbers, we give the following definition.

Definition 25 *A family of partial equivalence relations $\cong_\sigma$ on the domains D_σ are defined by:*

$$\begin{array}{rl} b_1 \cong_o b_2 & \textit{iff} \;\; b_1 = b_2 \\ n_1 \cong_\iota n_2 & \textit{iff} \;\; n_1 = n_2 \\ d_1 \cong_r d_2 & \textit{iff} \;\; q_{\mathcal{P}}(d_1) = q_{\mathcal{P}}(d_2) \\ d_1 \cong_{\sigma\to\sigma'} d_2 & \textit{iff} \;\; \forall d_1', d_2' \in D_\sigma . d_1' \cong_\sigma d_2' \;\Rightarrow d_1(d_1') \cong_{\sigma'} d_2(d_2') \end{array}$$

On basic types $\cong_\sigma$ is the finer equivalence relation that identifies the three representations of each dyadic real number. On higher types the relation $\cong_\sigma$ is defined hereditarily. For each element $d \in D_{\sigma\to\sigma'}$, we have $d \cong_{\sigma\to\sigma'} d$ if and only if d preserves the partial equivalence relation at lower types.

The partial equivalence relations $\cong_\sigma$ can be extended pointwise to environments:

$$\rho_1 \cong \rho_2 \text{ iff } \forall x^\sigma \,.\, \rho_1(x) \cong_\sigma \rho_2(x)$$

Proposition 26 *For every term M^σ in $\mathcal{L}_{wr}$ and environment ρ if $\rho \cong \rho$ then $\mathcal{E}[\![M^\sigma]\!]_\rho \cong_\sigma \mathcal{E}[\![M^\sigma]\!]_\rho$*

Proof. By structural induction on M. The only non-trivial case is represented by constants Y_σ. By an easy structural induction on types, it is possible to prove that for every type σ: $\perp \cong_\sigma \perp$ and that the relation $\cong_\sigma$ is closed by l.u.b. of chains, i.e. it is inductive. It follows that for any element $d \in D_{\sigma\to\sigma}$ such that $d \cong_{\sigma\to\sigma} d$ we have: $\mathcal{B}[\![Y_\sigma]\!](d) = \bigsqcup_{n\in N} d^n(\perp_\sigma) \cong_\sigma \bigsqcup_{n\in N} d^n(\perp_\sigma)$. Therefore: $\mathcal{B}[\![Y_\sigma]\!] \cong_{(\sigma\to\sigma)\to\sigma} \mathcal{B}[\![Y_\sigma]\!]$. $\square$

We need to prove that every computable function on reals is definable in $\mathcal{L}_{wr}$.

Proposition 27 *For each tuple of natural numbers $\overline{m}$ and for every computable functional F in $\mathbb{F}_{\overline{m}}$ there exists a term M in $\mathcal{L}_{wr}$ such that: $q_{\overline{m}}(\mathcal{E}[\![M]\!]) = F$.*

Proof. Given a term M in $\mathcal{L}_r$, let $M^\star$ indicate the term obtained by substituting each occurrence of the constant (≤ 0) in M with the constant (< 0). Let d be a computable element in $RD_{\overline{m}}$ such that $q_{\overline{m}}(d) = F$, and let M be the term in $\mathcal{L}_r$, which has d as its denotation and is constructed according to the proof of Theorem 24. We will prove that $q_{\overline{m}}(\mathcal{E}[\![M]\!]) = q_{\overline{m}}(\mathcal{E}[\![M^\star]\!])$.

We need to introduce the following definitions: for each natural number n, let r_n be the type inductively defined as: $r_0 = r$ and $r_{n+1} = r_0 \to r_n$. For each n-tuple of natural numbers $\overline{m} = \langle m_1, \ldots m_n\rangle$ let $r_{\overline{m}}$ be the type $r_{m_1} \to .. \to r_{m_n} \to r_0$. Let $w : RD \to RD$ be the function defined by:

$$w(d) = \{[a,b] \mid \exists [a',b'] \in d \,.\, [a',b'] \subset (a,b)\}.$$

For each natural number n let $w_n : FD_n \to FD_n$ be defined by:

$$w_n(g) = w \circ g.$$

It is not difficult to prove that:

(i) $\forall x \in \mathbb{R} \,.\, e(x) = w(e(x))$,
(ii) $\forall f \in \mathbb{F}_n \,.\, e_n(f) = w_n(e_n(f))$.

Moreover for each $d \in RD$:

$$\begin{aligned}
\mathcal{B}[\![(<0)]\!](d)) &= \mathcal{B}[\![(\leq 0)]\!](w(d)),\\
w(\mathcal{B}[\![(+1)]\!](d)) &= \mathcal{B}[\![(+1)]\!](w(d)),\\
w(\mathcal{B}[\![(-1)]\!](d)) &= \mathcal{B}[\![(-1)]\!](w(d)),\\
w(\mathcal{B}[\![(\div 2)]\!](d)) &= \mathcal{B}[\![(\div 2)]\!](w(d)),
\end{aligned}$$

The terms $P_r, \mathsf{pif}_\sigma, \mathsf{FIRST}_\sigma, \mathsf{SECOND}_\sigma$ do not contain the constant (≤ 0), therefore: $P_r^\star = P_r$, $\mathsf{pif}_\sigma^\star = \mathsf{pif}_\sigma$, $\mathsf{FIRST}_\sigma^\star = \mathsf{FIRST}_\sigma$ and $\mathsf{SECOND}_\sigma^\star = \mathsf{SECOND}_\sigma$

From the above identities it follows that:

(iii) $\forall d \in RD \,.\, \mathcal{E}[\![T_r^\star]\!]_\rho(d) = \mathcal{E}[\![T_r]\!]_\rho(w(d))$,
(iv) $\forall n \in \mathbb{N} \,.\, \forall g \in FD_n \,.\, \mathcal{E}[\![T_{r_n}^\star]\!]_\rho(f) = \mathcal{E}[\![T_{r_n}]\!]_\rho(w_n(f))$.

Lemma 28 *For each n-tuple of natural numbers $\overline{m}$, for each pair of elements $G, G' \in FD_{\overline{m}}$ and for each $g_1 \in FD_{m_1} \ldots, g_n \in FD_{m_n}$, if*

$$G(w_{m_1}(g_1)) \ldots (w_{m_n}(g_n)) \sqsubseteq G'(g_1) \ldots (g_n)$$

then

$$\forall i \in \mathbb{N} \,.\, \mathcal{E}[\![P_{r_{\overline{m}}}]\!]_\rho(i)(G)(w_{m_1}(g_1)) \ldots (w_{m_n}(g_n)) \sqsubseteq \mathcal{E}[\![P^\star_{r_{\overline{m}}}]\!]_\rho(i)G'(g_1) \ldots (g_n).$$

Proof of the lemma. The proof is by structural induction on the type $r_{\overline{m}}$. The basic step occurs when the sequence $\overline{m}$ is composed by a single element. In this case the proof becomes a simple calculation.

Inductive step. Let $\overline{m} = \langle m_1, \ldots m_n \rangle$, $\overline{m}' = \langle m_2, \ldots m_n \rangle$, we have:

$$\begin{array}{l}
\mathcal{E}[\![P_\sigma]\!]_\rho(i)(G)(w_{m_1}(g_1)) \ldots (w_{m_n}(g_n)) \\
= \mathcal{E}[\![Y_{\iota \to r_{\overline{m}'}}(\lambda \gamma^{\iota \to r_{\overline{m}'}} . \lambda n. \\
\qquad\qquad \mathsf{pif}_{r_{\overline{m}'}} \mathsf{Z} n \\
\qquad\qquad\qquad \mathsf{then}\ \alpha^{r_{\overline{m}}} \beta^{r_{m_1}} \\
\qquad\qquad\qquad \mathsf{else\ pif}_{r_{\overline{m}'}} T_{r_{m_1}}(\mathsf{FIRST}_{r_{\overline{m}}}\ m\, n)\beta^{r_{m_1}} \\
\qquad\qquad\qquad\qquad \mathsf{then}\ \gamma^{\iota \to r_{\overline{m}'}}(\mathsf{pred}(n)) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ P_{r_{\overline{m}'}}(\mathsf{SECOND}_{r_{\overline{m}}}\ m\, n)(\gamma^{\iota \to r_{\overline{m}'}}(\mathsf{pred}(n))) \\
\qquad\quad)(\mathsf{SIZE}_{r_{\overline{m}}}\ m)]\!]_{\rho[i/m][G/\alpha^{r_{\overline{m}}}][(w(g_1)/\beta^{m_1}]}(w_{m_2}(g_2)) \ldots (w_{m_n}(g_n))
\end{array}$$

Using the inductive hypothesis and the identities (iii) and (iv), by a simple calculation it is possible to prove that for each $G_1, G_1' : \mathbb{Z}_\perp \to RD_{\overline{m}'}$, $i \in \mathbb{N}$, $g_2 \in FD_{m_2} \ldots, g_n \in FD_{m_1}$ if $G_1(i)(w_{m_2}(g_2)) \ldots (w_{m_n}(g_n)) \sqsubseteq G_1'(i)(g_2) \ldots (g_n)$ then

$$\begin{array}{l}
\mathcal{E}[\![\lambda n \,.\, \mathsf{pif}_{r_{\overline{m}'}} \mathsf{Z} n \\
\qquad\qquad \mathsf{then}\ \alpha^{r_{\overline{m}}} \beta^{r_{m_1}} \\
\qquad\qquad \mathsf{else\ pif}_{r_{\overline{m}'}} T_{r_{m_1}}(\mathsf{FIRST}_{r_{\overline{m}}}\ m\, n)\beta^{r_{m_1}} \\
\qquad\qquad\qquad \mathsf{then}\ \gamma^{\iota \to r_{\overline{m}'}}(\mathsf{pred}(n)) \\
\qquad\qquad\qquad \mathsf{else}\ P_{r_{\overline{m}'}}(\mathsf{SECOND}_{r_{\overline{m}}}\ m\, n)(\gamma^{\iota \to r_{\overline{m}'}}(\mathsf{pred}(n)) \\
\quad]\!]_{\rho[i/m][G/\alpha^{r_{\overline{m}}}][G_1/\gamma^\sigma][(w(g_1)/\beta^{m_1}]}(w_{m_2}(g_2)) \ldots (w_{m_n}(g_n)) \\
\\
\sqsubseteq \mathcal{E}[\![\lambda n . \mathsf{pif}_{r_{\overline{m}'}} \mathsf{Z} n \\
\qquad\qquad \mathsf{then}\ \alpha^{r_{\overline{m}}} \beta^{r_{m_1}} \\
\qquad\qquad \mathsf{else\ pif}_{r_{\overline{m}'}} T^\star_{r_{m_1}}(\mathsf{FIRST}^\star_{r_{\overline{m}}}\ m\, n)\beta^{r_{m_1}} \\
\qquad\qquad\qquad \mathsf{then}\ \gamma^{\iota \to r_{\overline{m}'}}(\mathsf{pred}(n)) \\
\qquad\qquad\qquad \mathsf{else}\ P^\star_{r_{\overline{m}'}}(\mathsf{SECOND}^\star_{r_{\overline{m}}}\ m\, n)(\gamma^{\iota \to r_{\overline{m}'}}(\mathsf{pred}(n)) \\
\quad]\!]_{\rho[i/m][G'/\alpha^{r_{\overline{m}}}][G_1'/\gamma^\sigma][g_1/\beta^{m_1}]}(g_2) \ldots (g_n)
\end{array}$$

By the definition of $\mathcal{B}[\![Y_{\iota\to r_{\overline{m}'}}]\!]$ it follows that:

$\mathcal{E}[\![P_{r_{\overline{m}}}]\!]_\rho(i)(G)(w_{m_1}(g_1))\dots(w_{m_n}(g_n))$

$$\begin{aligned}
\sqsubseteq\ & \mathcal{E}[\![Y_{\iota\to r_{\overline{m}'}}(\lambda\gamma^{\iota\to r_{\overline{m}'}}.\lambda n.\\
&\qquad \mathsf{pif}_{r_{\overline{m}'}}\,\mathsf{Z}n\\
&\qquad\quad \mathsf{then}\ \alpha^{r_{\overline{m}}}\beta^{r_{m_1}}\\
&\qquad\quad \mathsf{else\ pif}_{r_{\overline{m}'}}\,T^\star_{r_{m_1}}(\mathsf{FIRST}^\star_{r_{\overline{m}}}\ m\,n)\beta^{r_{m_1}}\\
&\qquad\qquad \mathsf{then}\ \gamma^{\iota\to r_{\overline{m}'}}(\mathsf{pred}(n))\\
&\qquad\qquad \mathsf{else}\ P^\star_{r_{\overline{m}'}}(\mathsf{SECOND}^\star_{r_{\overline{m}}}\ m\,n)(\gamma^{\iota\to r_{\overline{m}'}}(\mathsf{pred}(n))))\\
&\qquad (\mathsf{SIZE}_{r_{\overline{m}}}\ m)]\!]_{\rho[i/m][G'/\alpha^{r_{\overline{m}}}][g_1/\beta^{m_1}]}(g_2)\dots(g_n)
\end{aligned}$$

$= \mathcal{E}[\![P^\star_{r_{\overline{m}}}]\!]_\rho(i)(G')(g_1)\dots(g_n)$

The lemma is proved.

By the lemma above it follows that for each tuple $\overline{m}$, for each $g_1 \in FD_{m_1},\dots,$ $g_n \in FD_{m_1}$,
$\mathcal{E}[\![Y_{\iota\to r_{\overline{m}}}(\lambda\alpha^{\iota\to r_{\overline{m}}}.\lambda n.P_{\sigma_{\overline{m}}}(Fn)(\alpha^{\iota\to r_{\overline{m}}}(\mathsf{succ}(n))))\underline{0}]\!]_\rho(w_{m_1}(g_1))\dots(w_{m_n}(g_n))$
$\sqsubseteq \mathcal{E}[\![Y_{\iota\to r_{\overline{m}}}(\lambda\alpha^{\iota\to r_{\overline{m}}}.\lambda n.P^\star_{\sigma_{\overline{m}}}(Fn)(\alpha^{\iota\to r_{\overline{m}}}(\mathsf{succ}(n))))\underline{0}]\!]_\rho(g_1)\dots(g_n)$.

The proposition follows immediately from the above inequality, from the identities (i) and (ii) and from the definition of the function $q_{\overline{m}}$. $\square$

6 A first attempt at an operational semantics

In this section we discuss the problem of defining an operational semantics for $\mathcal{L}_r$. In Section 4 the elements of RD are constructed as equivalence classes of partial sequences of integers. It is an obvious observation that a function having type $[\mathbb{Z}_\perp \to \mathbb{Z}_\perp]$ can be used to represent a sequence of integers and, as a consequence, an element in RD. Following this approach, higher order functions on $\mathbb{Z}_\perp$ can be employed to represent functions on RD. The construction is the following:

- let S' be the subset of $[\mathbb{Z}_\perp \to \mathbb{Z}_\perp]$ defined by,

$$\begin{aligned}
S' = \{s \mid \forall i \in \mathbb{N}\,.\ (\ & s(i+1) \neq \perp \Rightarrow\\
&\qquad (\ s(i) \neq \perp\ \wedge\ 2s(i) - 1 \leq s(i+1) \leq 2(i) + 1\ ))\}
\end{aligned}$$

  the elements of S' define the partial sequences of digits representing elements in RD.

- let $\phi' : S' \to RD$ be the function,

$$\phi'(s) = \{\downarrow [\frac{s(i)-1}{2^i}, \frac{s(i)+1}{2^i}] \mid i \in \mathbb{N}, s(i) \neq \bot\}$$

Give a function $g : RD \to RD \to RD$, we say that g is represented by a function $f : [\mathbb{Z}_\bot \to \mathbb{Z}_\bot] \to [\mathbb{Z}_\bot \to \mathbb{Z}_\bot] \to [\mathbb{Z}_\bot \to \mathbb{Z}_\bot]$ if for all $s_1, s_2 \in S'$, $g(\phi'(s_1))(\phi'(s_2)) = \phi'(f(s_1)(s_2))$.

The above representation for functions on RD suggests the following approach to operational semantics: for any new constant c in $\mathcal{L}_r$ it is required that we find a function f_c on $[\mathbb{Z}_\bot \to \mathbb{Z}_\bot]$ representing the function $\mathcal{B}[\![c]\!]$. If the functions f_c existed then a set of closed $\mathcal{L}_{PA+\exists}$-terms M_c, such that $\mathcal{E}[\![M_c]\!]_\rho = f_c$, could be used to define an operational semantics for $\mathcal{L}_r$. More precisely, the operational semantics would be given by the reduction rules $c \to M_c$. These rules are justified since the operational behaviour of the hypothetical term M_c would be in accord with the denotational semantics of c. In fact, we prove that this approach cannot be taken.

Notation. Given a function $s : \mathbb{Z}_\bot \to \mathbb{Z}_\bot$ and a natural number n we denote with $s\,|_n$ the restriction of the function s to the elements smaller that n: $s|_n\,(m) = s(m)$ if $m \leq n$, $s|_n\,(m) = \bot$ otherwise.

Proposition 29 *There is no function $g : \{\mathsf{tt}, \mathsf{ff}\}_\bot \to [\mathbb{Z}_\bot \to \mathbb{Z}_\bot] \to [\mathbb{Z}_\bot \to \mathbb{Z}_\bot] \to [\mathbb{Z}_\bot \to \mathbb{Z}_\bot]$ representing the function $\mathcal{B}[\![\mathsf{pif}_r]\!]$.*

Proof. By contradiction. Suppose there exists a continuous function g representing the function $\mathcal{B}[\![\mathsf{pif}_r]\!]$. Let x be a real number and s, t be two representations of x.

For every $i \in \mathbb{N}$ we have:

$$\begin{aligned} g(\bot)(s)(s)(i) &= g(\mathsf{tt})(s)(s)(i) = g(\mathsf{tt})(s)(\bot)(i) = g(\mathsf{tt})(s)(t)(i) \\ &= g(\bot)(s)(t)(i) = g(\mathsf{ff})(s)(t)(i) = g(\mathsf{ff})(\bot)(t)(i) \\ &= g(\mathsf{ff})(t)(t)(i) = g(\bot)(t)(t)(i) \end{aligned}$$

In fact, all pairs of elements in the above equations are order related, and since they are all different from $\bot$ they must be equal. The function $\lambda s \,.\, g(\bot)(s)(s) : [\mathbb{Z}_\bot \to \mathbb{Z}_\bot] \to [\mathbb{Z}_\bot \to \mathbb{Z}_\bot]$ is therefore a continuous function that, for each real number, selects a canonical representation.

We now prove that this function cannot exist. Let $A = \{q(\phi'(s)) \mid g(\bot)(s)(s)(0) = 1\}$, let x be a real number in the boundary of A and let t be a representation of x such that for each $n \in \mathbb{N}$ all the intervals in $\phi'(t_n)$ are neighbourhoods of x. It is not difficult to verify that the representation t exists.

The value $g(\bot)(t)(t)(0)$ must be different from 1. In fact, by continuity of g, there exists a natural number n such that $g(\bot)(t|_n)(t|_n)(0) = g(\bot)(t)(t)(0)$. Let t' be a representation of a real number such that $t'|_n = t|_n$ and $q(\phi'(t')) \notin A$. We have $g(\bot)(t)(t)(0) = g(\bot)(t|_n)(t|_n)(0) = g(\bot)(t')(t')(0) \neq 1$.

By similar arguments it is possible to prove that $g(\bot)(t)(t)(0) = 1$. We obtain a contradiction. $\square$

It is possible to give a stronger result and prove that there exists no representation for a function that behaves like a "parallel if" on the infinite elements (no matter how the function is extended to partial elements). Moreover, this negative result can be extended to a large class of different representations for real numbers. In almost all the representations considered in the literature, a real number is represented by a sequence of elements of a countable set C. For example, C can be a set of digits, the set of integers, the set of p-adic rational numbers, the set of rational numbers, the set of rational intervals.

Definition 30 *A* sequence representation $\langle C, S, v\rangle$ *is given by a countable set C, a subset S of $\mathbb{N} \to C$ and a representation function $v : S \to \mathbb{R}$. The set S is the subset of sequences defining real numbers.*

By repeating the construction of Section 4, we maps finite sequences to subsets of reals.

Definition 31 *Given a sequence representation $\langle C, S, v\rangle$, the extension of the representation $v : S \to \mathbb{R}$, to partial sequences, $\overline{v} : [\mathbb{N} \to C_\bot] \to \mathcal{P}(\mathbb{R})$, is defined by:*

$$\overline{v}(s) = \{v(t) \mid t \in S, s \sqsubseteq t\}.$$

The notion of admissible representation for real numbers has first been introduced in [27, pages 479–482]. That definition can be reformulated as follows.

Definition 32 *A sequence representation $\langle C, S, v\rangle$ is* admissible *if it satisfies the following conditions,*

(i) $\forall s \in S, \epsilon \in \mathbb{R} \,.\, \exists n \in \mathbb{N} \,.\, \overline{v}(s|_n)$ *is contained in an interval having width ϵ,*

(ii) For each real number x there exists a sequence s such that for each natural n, x is contained in the interior of $\overline{v}(s|_n)$.

(iii) Real numbers are represented only by totally defined sequences.

Condition (i) states that the function $v : S \to \mathbb{R}$ is continuous, w.r.t. the Cantor topology on S and the Euclidean topology on $\mathbb{R}$. Condition (ii) implies that the Euclidean topology is the finer topology on the real line for which the function v is continuous. Almost all the representation functions used in computable analysis are admissible. There exist representations that are commonly used and not admissible (e.g. the decimal representation). These commonly used representations are not suitable for computable analysis. It is a well known result that the arithmetic operations on reals are not computable when the decimal representation is used ([27])). In Section 7 we present a new representation for real numbers. The new representation is a non admissible representation but it can be used for computable analysis.

Proposition 33 *For any admissible representation v there is no continuous functional $g : \{\mathsf{tt}, \mathsf{ff}\}_\perp \to [\mathbb{N} \to C_\perp] \to [\mathbb{N} \to C_\perp] \to [\mathbb{N} \to C_\perp]$ representing a parallel test, that is, for all $b \in \{\mathsf{tt}, \mathsf{ff}\}_\perp$, s, t in S,*

$$\begin{aligned} v(g(b)(s)(t)) &= v(s) \ \textit{if}\ b = \mathsf{tt}, \\ v(g(b)(s)(t)) &= v(t) \ \textit{if}\ b = \mathsf{ff}, \\ v(g(b)(s)(t)) &= v(s) \ \textit{if}\ v(s) = v(t) \end{aligned}$$

The proof of Proposition 29 can be easily modified to obtain a proof for this proposition.

There are two possible solutions to the problem of defining a parallel test for reals. The first one consists in introducing non deterministic or intensional operators in the language. The second one consists in using a different representation for real numbers. The first approach has been followed in [12], where an operational semantics of a language similar to $\mathcal{L}_r$ is given using a non deterministic operator. The second approach we will be followed here.

7 An operational semantics

In the literature, real numbers are represented by sequences that are completely defined. We maintain that it is possible to represent real numbers using sequences that are undefined on some index. An example is the following.

Definition 34 *A real number x in the interval $[-1, 1]$ is represented by a sequence s of digits $-1, 1$ such that: $x = \sum_{i \in N} \prod_{0 \leq j \leq i} s_j / 2$*

This notation is similar to the binary digit notation. The main differences consist in the use of the digit -1 instead of the digit 0 and in the fact that in this notation the value of a digit affects the weights of all consecutive digits. In this notation, the real number 0 has two representations: the sequence $\langle -1, -1, 1, 1, 1 \ldots \rangle$ and the sequence $\langle 1, -1, 1, 1, 1 \ldots \rangle$. The two representations differ just for the first digit. Therefore, 0 can also be represented by the sequence $\langle \perp, -1, 1, 1, 1 \ldots \rangle$ undefined on the first element. Moreover, by examining the finite initial parts of the incomplete sequence, it is possible to determine the number it represents with arbitrary precision. Similar considerations hold for any other dyadic rational number: every dyadic rational number has two representations, which differ for just one element. Therefore, every dyadic rational number can be represented also by a partial sequence diverging on one element. Every real number that is not rational dyadic has exactly one representation. If we allow that a sequence undefined on one element may be a possible representation for a real number then, we obtain a representation which is suitable for real number computation.

In order to represent the whole real line we consider the following notation.

Definition 35 *A representation function $v : (\mathbb{N} \to \{-1, 1\}) \to \mathbb{R}$ is defined by:*

$$v(s) = s(0) \times (k + \sum_{i \geq k} \prod_{0 \leq j \leq i} s(j)/2)$$

where $k = min\{i \mid i > 0,\, s(i) = -1\}$

This is a sort "sign, integer part, mantissa" notation for the real numbers. The first digit gives the sign, the next consecutive positive digits determine the integer part, the remaining part of the sequence is the mantissa. Also, in this case, every dyadic rational number is represented by two sequences that differ just for one element and every real number that is not rational dyadic has exactly one representation.

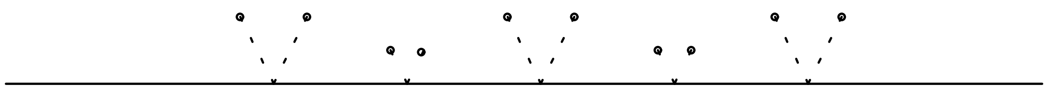
Definition 36 *The extension of v to partial sequences is the function $\overline{v} : (\mathbb{N} \to \{-1, 1\}_\perp) \to \mathcal{P}(\mathbb{R})$ defined by:*

$$\overline{v}(s) = \{v(t) \mid t : \mathbb{N} \to \{-1, 1\},\, s \sqsubseteq t\}.$$

Proposition 37 *The set $\overline{v}(s)$ is an interval if and only if*

$$\begin{aligned} \forall n \,.\, (s(n)\uparrow \wedge s(n+1)\downarrow) \Rightarrow\ & \forall m < n \,.\, s(m)\downarrow \\ & \wedge\ s(n+1) = -1 \\ & \wedge\ \forall m > n+1 \,.\, (s(m)\uparrow \vee s(m) = 1). \end{aligned}$$

Let S^∞ denote the set of partial sequences s such that $\overline{v}(s)$ is an interval. S^∞ is a complete partial order when the subsequence order is considered. If we repeat the construction of Section 4, with the representation v and the set S^∞ of partial elements, we obtain a new domain for real numbers. We call the new domain RD'. In this case no pair of elements in S^∞ contains the same information. It follows that S^∞ and RD' are isomorphic.



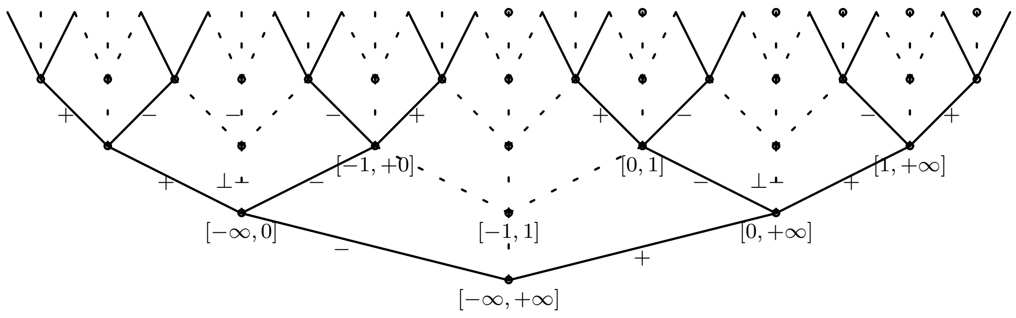


Fig. 3. The diagram representing RD'.

The structures of RD and RD' are quite similar. The main difference consists in the fact that for each natural number n, RD' contains the intervals $[-\infty, -n]$ and $[n, +\infty]$ and, as a consequence, the infinite points $-\infty$ and $+\infty$. It is possible to define a representation for real numbers similar to the one given at Definition 35, in such a way the approximation domain obtained from it is exactly the domain RD. This alternative notation needs to be more complex. In order to have a simpler operational semantics we prefer to use the domain RD' and slightly modify the calculus. Namely, we change one constant in $\mathcal{L}_r$, substituting the constant (-1) with the constants abs and compl, which represent the functions "absolute value" and "complementation". Their denotational semantics is:

$$\begin{aligned}
\mathcal{B}[\![(\mathsf{abs})]\!](d) &= \{[a,b] \mid [a,b] \in d, 0 \le a\} \cup \{[-b,-a] \mid [a,b] \in d, b \le 0\} \\
&\quad \cup \{[0,b] \mid [a,b] \in d, a < 0 < b\} \\
\mathcal{B}[\![(\mathsf{comp})]\!](d) &= \{[-b,-a] \mid [a,b] \in d\}
\end{aligned}$$

We call $\mathcal{L}_{r'}$ this new extension of PCF with real numbers. The proof of Theorem 24 can be trivially modified to prove that $\mathcal{L}_{r'}$ is universal.

Using S^∞ as a representation for RD', it is not difficult to define an operational semantics for $\mathcal{L}_{r'}$. The method we employ is the one described in the first part of Section 6.

For convenience, instead of sequences of $-1, 1$, we use sequences of boolean values: ff stands for -1 and tt stands for 1. Sequences of boolean values are represented by terms having type $\iota \to o$. For each constant c in the set $\{(+1), \mathsf{abs}, \mathsf{comp}, (\times 2), (\div 2), \mathsf{PR}, (\le 0), \mathsf{pif}_r\}$, the denotational value of c induces a function g on S^∞ (RD' and S^∞ are isomorphic), g can be extended to a complete and computable function h on the domain $\mathbb{Z}_\perp \to \{\mathsf{tt}, \mathsf{ff}\}_\perp$ (the extension exists because S^∞ is an effective retraction of the space $\mathbb{Z}_\perp \to \{\mathsf{tt}, \mathsf{ff}\}_\perp$) and, by the universality of $\mathcal{L}_{\exists+PA}$, h can be denoted by a suitable term M. The term M defines the operational semantics of the constant c.

The operational semantics of $\mathcal{L}_{r'}$ is given by the set of reduction rules of $\mathcal{L}_{\exists+PA}$, with the following added rules:

```
$(+1) \to \lambda s.\lambda n.\mathsf{pif}_o s(0)$
                    $\mathsf{then\ if}_o \mathsf{Z}(n)\ \mathsf{then\ tt\ else}\ s(\mathsf{pred}(n))$
                    $\mathsf{else\ pif}_o s(1)$
                         $\mathsf{then\ if}_o(n = 1)\mathsf{then\ ff\ else\ NOT}(s(n))$
                         $\mathsf{else\ if}_o \mathsf{Z}(n)\mathsf{then\ ff\ else}\ s(\mathsf{succ}(n))$

$\mathsf{abs} \to \lambda s.\lambda n.\mathsf{if}_o \mathsf{Z}(n)\mathsf{then\ tt\ else}\ s(n)$

$\mathsf{comp} \to \lambda s.\lambda n.\mathsf{if}_o \mathsf{Z}(n)\mathsf{then\ NOT}(s(n))\ \mathsf{else}\ s(n)$

$(\times 2) \to \lambda s.\lambda n.\mathsf{if}_o \mathsf{Z}(n)\mathsf{then}\ s(0)\mathsf{else\ double}(\lambda m.s(\mathsf{succ}(m)))$

where the the term double is defined by:

$\mathsf{double}(s)(n) = \mathsf{pif}_o s(0)$
                    $\mathsf{thenif}_o \mathsf{Z}(\mathsf{succ}(n))$
                         $\mathsf{then\ tt}$
                         $\mathsf{else\ double}(\lambda m.s(\mathsf{succ}(n)))(n-2)$
                    $\mathsf{else\ pif}_o s(1)$
                         $\mathsf{then\ if}_o \mathsf{Z}(n)\mathsf{then\ ff\ else}\ s(n+1)$
                         $\mathsf{else\ if}_o \mathsf{Z}(n)\mathsf{then\ tt\ else}\ s(n)$

$(\div 2) \to \lambda s.\lambda n.\mathsf{if}_o \mathsf{Z}(n)\mathsf{then}\ s(0)\mathsf{else\ half}(\lambda m.s(m+1))$

where the the term half is defined by:

$\mathsf{half}(s)(n) = \mathsf{pif}_o s(0)$
                  $\mathsf{then\ pif}_o s(1)$
                       $\mathsf{then\ if}_o \mathsf{Z}(n)$
                            $\mathsf{then\ tt}$
                            $\mathsf{else\ half}(\lambda m.s(m+2))(\mathsf{pred}(n))$
                       $\mathsf{else\ if}_o(n = 1)\mathsf{then\ ff\ else\ NOT}(s(n))$
                  $\mathsf{else\ if}_o \mathsf{Z}(n)$
                     $\mathsf{then\ ff}$
                     $\mathsf{else\ if}_o \mathsf{Z}(\mathsf{succ}(n))\mathsf{then\ tt\ else}\ s(\mathsf{pred}(n))$
```

```
$PR \rightarrow \lambda s.\lambda n.\mathsf{if}_o \mathsf{Z}(n)$
            then $s(0)$
            else $\mathsf{if}_o \mathsf{Z}(\mathsf{succ}(n))$
                then ff
                else $\mathsf{pif}_o s(1)$
                    then $\mathsf{if}_o \mathsf{Z}(\mathsf{succ}(\mathsf{succ}(n)))$ then ff else tt
                    else $s(n)$
```

$(\leq 0) \rightarrow \lambda s.s(0)$

$\mathsf{if}_r \longrightarrow \lambda b.\lambda s.\lambda t.\lambda n.\mathsf{pif}_o b \,\mathsf{then}\, s(n\,)\mathsf{else}\, t(n)$

Proposition 38 (Adequacy) *For every closed term M having type ι or o,*

$$\mathcal{E}[\![M]\!]_\rho = \begin{cases} Eval(M) & \text{if } Eval(M) \text{ is defined} \\ \perp & \text{otherwise.} \end{cases}$$

For every closed term M having type r:

$$[a,b] \in \mathcal{E}[\![M]\!]_\rho \text{ iff } \exists n\,.\,\overline{v}(\langle Eval(M(0)),\ldots,Eval(M(n))\rangle) \subseteq [a,b]$$

Proof. The standard computability method (see [18]) can be applied to prove the adequacy of $\mathcal{L}_{r'}$. There is, however, a further difficulty. It can be noted that, in $\mathcal{L}_r$, reduction rules do not preserve the denotational semantics of terms. In fact, a constant reduces to its "implementation" and the semantics of the implementation cannot be given using the domain RD'. This difficulty can be easily overcome by slightly modifying the proof technique.

The predicate COMP on $\mathcal{L}_{r'}$-terms is defined by:

- a closed term M having type ι or o has the property COMP (is computable) if:
  $\mathcal{E}[\![M]\!]_\rho =$Eval($M$) if Eval($M$) is defined and
  $\mathcal{E}[\![M]\!]_\rho = \perp$ otherwise.
- a closed term M^r is computable if:
  $[a,b] \in \mathcal{E}[\![M]\!]_\rho \;\Leftrightarrow\; \exists n\,.\,\overline{v}(\langle \text{Eval}(M(0)),\ldots,\text{Eval}(M(n))\rangle) \subseteq [a,b]$
- a closed term having $M^{(\sigma_1\rightarrow\sigma_2)}$ is computable if for every closed computable term N^{σ_1} the term $M^{(\sigma_1\rightarrow\sigma_2)}(N^{\sigma_1})$ is computable
- an open term M with free variables $x_1,\ldots,x_n$ is computable if for every closed computable terms $N_1,\ldots N_n$ the term $[N_1/x_1,\ldots N_n/x_n]M$ is computable.

It is straightforward to prove by structural induction that every $\mathcal{L}_{r'}$ term is computable. $\square$

8 Sequentiality and real number computations

In a calculus for real numbers based on the previous representation, the use of parallel operators cannot be avoided. In fact, the implementation of a total function on reals must contain parallel operators; otherwise, the computation will diverge as soon as it examines an undefined digit. We discuss whether, in general, parallelism is necessary to perform exact computation on real numbers. There is no straight answer to this question.

Parallel computation is not necessary in a calculus whose functions do not need to preserve the equivalence relation which exists among the different representations of the same real number and among the different representations of the same approximated real. The calculus for real numbers presented in Section 3 is an example.

In the case of calculi where the equivalence relation is preserved, it is necessary to consider which kind of representation for real numbers is being used, and what are the approximated reals that can be obtained as a result of computation.

Given a calculus for real numbers based on a sequence representation $\langle C, S, v\rangle$, let S^0 indicate the set of partial sequences that can be generated as a result of a computation.

In the calculi proposed so far in the literature, S^0 is the set of initial sequences. In general, we consider the case where S^0 is a dense subset of the set of initial sequence, i.e., for each $s \in S$ and for each $i \in \mathbb{N}$, there exists $j \geq i$ such that $s|_j \in S^0$. Any sequence representation induces an information order on partial sequences. In this order, s is below t if $\overline{v}(s) \supseteq \overline{v}(t)$. If the denotational semantics of the calculus is based on a domain of approximations then the implementation of the function must preserve the information order on partial sequences of S^0.

We will prove that, in a calculus using an admissible representation, where the set of partial elements S^0 is a dense subset of the initial sequences, the use of parallel operators is unavoidable.

In order to prove this, we need to use the notion of dI-domains and stable functions. We recall here the standard definition and properties. See [2] for a more complete account.

A Scott-domain D is a dI-domain if (i) for every finite elements $d \in D$ the set $\{d' \mid d' \sqsubset d\}$ is finite and (ii) D is distributive, that is, for $d, d', e \in D$, if d, d' have an upper bound then $(d \sqcup d') \sqcap e = (d \sqcap e) \sqcup (d' \sqcap e)$. A function $f : D \to D'$ between dI-domains is stable if it is continuous and for every bounded $d, e \in D$ if d, e have an upper bound then $f(d \sqcap d') = f(d) \sqcap f(e)$. f is below g in the stable order if for all $d, e \in D$ if $d \sqsubset e$ then $f(d) = f(e) \sqcap g(d)$. If D, D' are two dI-domains, then the set of stable functions from D to D' with the stable order is a dI-domain. In particular dI-domains and stable functions form a Cartesian closed category. Stability is a property that is satisfied by the sequential operators but not by the parallel ones.

The dI-domains form a model for the language $\mathcal{L}$ but not for the language $\mathcal{L}_{PA}$. There is no stable function giving an adequate semantics to the constant pif_o.

Theorem 39 *For any admissible representation $\langle C, S, v\rangle$, for any dense subset S^0 of initial sequences of S, and for any function $f : (\mathbb{R} \times \mathbb{R}) \to \mathbb{R}$, if f is not constant in any of the two variables, then there is no stable continuous functional $g : [\mathbb{N}_\perp \to C_\perp] \to [\mathbb{N}_\perp \to C_\perp] \to [\mathbb{N}_\perp \to C_\perp]$ such that:*

(i) g implements f, i.e., for all s, t in S, $f(v(s))(v(t)) = v(g(s)(t))$
(ii) g respects the induced order relation on partial sequences, i.e., for all s, s', t, t' in S^0, $\overline{v}(s) \supseteq \overline{v}(s')$ and $\overline{v}(t) \supseteq \overline{v}(t')$ implies $\overline{v}(g(s)(t)) \supseteq \overline{v}(g(s')(t'))$.

Proof. By contradiction. Suppose that the function g exists. Since the function f is not constant, g implements f, and g is continuous there exist $s|_i$, $t|_i$ in S°, t' in S, (a, b) rational interval, such that,

(i) $\overline{v}(g(s|_i)(t|_i)) \subseteq (a, b)$,
(ii) $\overline{v}(g(s|_i)(t')) \cap (a, b) = \emptyset$

Given a set of real numbers A, let $\mathsf{Int}(A)$ denote the interior of A, that is the largest open set contained in A. Let B be the set of real numbers defined by

$$B = \bigcup\{\mathsf{Int}(\overline{v}(t''|_j)) \mid t''|_j \in S^\circ, \overline{v}(g(s|_i)(t''|_j)) \subseteq (a, b)\}$$

By conditions (i) and (ii), the set B is a non-empty proper subset of $\mathbb{R}$ so there exists a number x that belongs to the boundary of B. Let u be a representation of x such that, for each natural number i, x is contained in the interior of $\overline{v}(u|_i)$. Since the function f is not constant, the value $g(s|_i, u)$ is a partial element. By continuity of g there exist j, k such that $s|_j, u|_k \in S^0$ and $g(s|_j, u|_k)$ is strictly more defined than $g(s|_i, u)$. By construction, there exists a representation u' such that $u|_k = u'|_k$ and $v(u') \in B$. By definition of B, there exists $l > k$ such that $\overline{v}(g(s|_i, u'|_l)) \subseteq (a, b)$.

It follows that both $g(s|_i, u'|_l)$ and $g(s|_j, u'|_k)$ are initial partial sequences of digits strictly more defined than $g(s|_i, u'|_k)$. Hence,

$$g(s|_i, u'|_l) \sqcap g(s|_j, u'|_k) = \min\{g(s|_i, u'|_l), g(s|_j, u'|_k)\} \neq g(s|_i, u'|_k).$$

Therefore, g cannot be a stable function. $\square$

The previous proposition cannot be generalised to arbitrary representations. With a suitable representation it is possible to define calculi for real numbers, which are universal and whose operational semantics can be given by a sequential and deterministic set of reduction rules.

What we are going to present here is not a sequential calculus for real numbers but just the idea for a possible definition.

Consider the following representations for the integers and for the booleans. Each integer n has an infinite set of different representations namely the set $\{n^*_i : \mathbb{Z} \to \mathbb{Z} \mid i \in \mathbb{N}, \forall j < i \,.\, n^*_i(j) = 1, \forall j \geq i \,.\, n^*_i(j) = 2 \times n\}$. The boolean value true is represented by any function in the set $\{\mathsf{tt}^*_i : \mathbb{Z} \to \mathbb{Z} \mid i \in \mathbb{N}, \forall j < i \,.\, \mathsf{tt}^*_i(j) = 1, \forall j \geq i \,.\, \mathsf{tt}^*_i(j) = 2\}$. The boolean value false is represented by any function in the set $\{\mathsf{tt}^*_i : \mathbb{Z} \to \mathbb{Z} \mid i \in \mathbb{N}, \forall j < i \,.\, \mathsf{ff}^*_i(j) = 1, \forall j \geq i \,.\, \mathsf{ff}^*_i(j) = 0\}$.

This representation can be implemented in PCF by using non-standard types σ^* defined as follows, $\iota^* = \iota \to \iota$, $o^* = \iota \to \iota$ and $(\sigma_1 \to \sigma_2)^* = \sigma^*_1 \to \sigma^*_2$.

To each $\mathcal{L}_{PA+\exists}$ constant c we associate a corresponding term c^*. The constants $\underline{n}^*, \mathsf{pred}^*, \mathsf{succ}^*, \mathsf{tt}^*, \mathsf{ff}^*, Z^*$ are defined pointwise:

$$\begin{array}{ll}
\underline{n}^* = \lambda i \,.\, \underline{2 \times n} & \mathsf{succ}^* = \lambda n i \,.\, \mathsf{if}_o n(i) = \underline{1}\,\mathsf{then}\,\underline{1}\,\mathsf{else}\,\mathsf{succ}(\mathsf{succ}(n(i))) \\
\mathsf{ff}^* = \lambda i \,.\, \underline{0} & \mathsf{pred}^* = \lambda n i \,.\, \mathsf{if}_o n(i) = \underline{1}\,\mathsf{then}\,\underline{1}\,\mathsf{else}\,\mathsf{pred}(\mathsf{pred}(n(i))) \\
\mathsf{tt}^* = \lambda i \,.\, \underline{2} & \mathsf{Z}^* = \lambda n i \,.\, \mathsf{if}_o n(i) = \underline{1}\,\mathsf{then}\,\underline{1}\,\mathsf{else}\,\mathsf{if}_o \mathsf{Z}(n(i))\,\mathsf{then}\,\underline{2}\,\mathsf{else}\,\underline{0}
\end{array}$$

$\mathsf{if}^*_n, \mathsf{pif}^*_n, Y^*_{\iota\to\iota}, \exists^*$ are defined as follows:

$$\mathsf{if}^*_n = \lambda bnmi \,.\, \mathsf{if}_o b(i) = \underline{2}\,\mathsf{then}\,n(i)\,\mathsf{else}\,\mathsf{if}_o b(i) = \underline{0}\,\mathsf{then}\,m(i)\,\mathsf{else}\,\underline{1}$$

$$\begin{aligned}
\mathsf{pif}^*_n = \lambda bnmi \,.\, & \mathsf{if}_o b(i) = \underline{2}\,\mathsf{then}\,n(i) \\
& \mathsf{else}\,\mathsf{if}_o b(i) = \underline{0}\,\mathsf{then}\,m(i) \\
& \mathsf{else}\,\mathsf{if}_o n(i) = m(i)\,\mathsf{then}\,n(i) \\
& \mathsf{else}\,\underline{1}
\end{aligned}$$

$$Y^*_{\sigma_1\to\ldots\to\sigma_n\to\iota} = \lambda f x_1 \ldots x_n i \,.\, f^i(\Omega^*_{\sigma_1\to\ldots\to\sigma_n\to\iota})(x_1)\ldots(x_n)(i)$$

where Ω^*_σ denotes the terms inductively defined by: $\Omega^*_o = \Omega^*_\iota = \lambda i \,.\, \underline{1}$ $\Omega^*_{\sigma_1\to\sigma_2} = \lambda x \,.\, \Omega^*_{\sigma_2}$, and f^i is defined by

$$f^i = \lambda x \,.\, (Y_{\iota\to\sigma_1\to\ldots\to\sigma_n\to\iota}(\lambda y j \,.\, \mathsf{if}_o \mathsf{Z}(j)\mathsf{then}\,x\,\mathsf{else}\,f(y(\mathsf{pred}(j))))(i))$$

```
$\exists^* = \lambda f \,.\, Y_{\iota\to\iota}(\lambda g i \,.\, \mathsf{if}_o i = \underline{0}$
                    $\mathsf{then}\ \underline{1}$
                    $\mathsf{else\ if}_o f(\lambda x.\underline{1})(i) = \underline{0}$
                        $\mathsf{then}\ \underline{0}$
                        $\mathsf{else\ if}_o f(\lambda j \,.\, P_1(i))(P_2(i)) = \underline{2}$
                            $\mathsf{then}\ \underline{2}$
                            $\mathsf{else}\ g(\mathsf{pred}(i)))$
```

where P_1 and P_2 define two primitive recursive functions π_1, π_2 such that for each natural number n, $n = \langle \pi_1(n), \pi_2(n) \rangle$, for a suitable coding function $\langle\ \rangle$.

Observe that all the terms c^* belong to the language $\mathcal{L}$, since they do not contain parallel operators. Given a term M, we denote with M^* the term obtained by substituting each constant with the corresponding non-standard version. It is not too difficult to prove that, for any term M having type ι, $\mathrm{Eval}(M) = n$ if and only if $\exists i \in N \,.\, (\forall j > i \,.\, \mathrm{Eval}(M^*(\underline{j})) = \underline{2 \times n}) \wedge (\forall j \leq i \,.\, \mathrm{Eval}(M^*(\underline{j})) = \underline{1})$; also, $\mathrm{Eval}(M)$ is undefined if and only if for all $i \in \mathbb{N}$ $\mathrm{Eval}(M^*(\underline{i})) = \underline{1}$.

The idea in this representation is to internalise the undefined element. The function $\lambda x \,.\, \underline{1}$ is a total function that represents the undefined computation. By using this representation for the bottom element, it is possible to emulate the parallel computation in a sequential calculus.

The above representation for the natural numbers is certainly unusual. However, there are several representations for real numbers that are commonly used, and where infinite sequences do not necessarily describe totally defined real numbers, but approximations of real numbers ([12,11]). If we use representations in this form, it is possible to define a calculus where all terms representing real numbers generate an infinite sequence of digits. In this way it is possible to perform the computation sequentially.

From a practical point of view, a sequential calculus of this form does not solve the efficiency problem caused by parallel operators. The parallelism in computation is not avoided, it is just emulated sequentially.

Acknowledgements

The research presented in this paper was started at the departement of Computing, Imperial College London, where the authour spent a few months as a research fellow thanks to an EPSRC grant. Discussions with Abbas Edalat and Martin Escardo, as well as the comments of the anonymous referees are

gratefully aknowledged.

References

[1] O. Aberth. *Computable analysis.* MacGraw-Hill, New York, 1980.

[2] G. Berry. Stable models of the typed lambda-calculi. In *Proc. 5th Int. Coll. on Automata Languages and Programming.*, number 62 in LNCS. Springer, 1978.

[3] E. Bishop. *Foundation of constructive analysis.* McGraw-Hill, New York, 1967.

[4] H.-J. Boehm and R. Cartwright. Exact real arithmetic: formulating real numbers as functions. In David Turner, editor, *Research topics in functional programming*, pages 43–64. Addison-Wesley, 1990.

[5] H.-J. Boehm, R. Cartwright, M. Riggle, and M.J. O'Donell. Exact real arithmetic: a case study in higher order programming. In *ACM Symposium on lisp and functional programming*, 1986.

[6] G. S. Ceitin. Algorithmic operators in constructive metric spaces. *Trudy Mat. Inst. Steklov, english translation, Amer. Math. Soc. Transl.*, 64(2):1–80, 1967.

[7] S. Cook. Computability and complexity of higher type functions. In *MSRI Proceedings*, 1990.

[8] P. Di Gianantonio. *A functional approach to real number computation.* PhD thesis, University of Pisa, 1993.

[9] P. Di Gianantonio. Real number computability and domain theory. *Information and Computation*, 127(1):11–25, May 1996.

[10] A. Edalat and M. Escardo. Integration in real PCF. In *IEEE Symposium on Logic in Computer Science*, pages 382–393, 1996.

[11] A. Edalat and P. J. Potts. A new representation for the exact real numbers. In *MFPS 97*, volume 6 of *E.N.T.C.S.* Elsevier Science, 1997.

[12] M. Escardo. PCF extended with real numbers. *Theoret. Comput. Sci*, pages 79–115, July 1996.

[13] A. Grzegorczyk. On the definition of computable real continuous functions. *Fund. Math.*, 44:61–77, 1957.

[14] D. Lacombe. Quelques procédés de définitions en topologie recursif. In *Constructivity in mathematics*, pages 129–158. North-Holland, 1959.

[15] P. Martin-Löf. *Note on Constructive Mathematics.* Almqvist and Wiksell, Stockholm, 1970.

[16] Ménissier-Morain. *Arithmétique exacte, conception, algorithmique et performances d'une implémentation informatique en prcision arbitraire.* Thèse, Université Paris 7, December 1994.

[17] V. Ménissier-Morain. Arbitrary precission real arithmetic: design and algorithms. Submitted to the Journal of Symbolic Computation. Available at http://pauillac.inria.fr/ menissier.

[18] G.D. Plotkin. LCF considered as a programing language. *Theoret. Comput. Sci.*, 5:223–255, 1977.

[19] P. J. Potts, A. Edalat, and M. H. Escardo. Semantcis of exat real arithmetic. In *IEEE Symposium on Logic in Computer Science*, 1997.

[20] H.G. Rice. Recursive real numbers. In *Proc. Amer. Math. Soc 5*, pages 784–791, 1954.

[21] Dana Scott. Outline of the mathematical theory of computation. In *Proc. 4th Princeton Conference on Information Science*, 1970.

[22] A. Simpson. Lazy functional algorithms for exact real functionals. In *MFCS 1998*, LNCS. Springer-Verlag, 1998.

[23] T. Streicher. A universality theorem for pcf with recursive types, parallel-or and $\exists$. *Mathematical Structures for Computing Science*, 4(1):111–115, 1994.

[24] A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics.* North-Holland, Amsterdam, 1988.

[25] A.M. Turing. On computable numbers, with an application to the entscheidungs problem. In *Proc. London Math. Soc. 42*, pages 230–265, 1937.

[26] J. Vuillemin. Exact real computer arithmetic with continued fraction. In *Proc. A.C.M. conference on Lisp and functional Programming*, pages 14–27, 1988.

[27] K. Weihrauch. *Computability.* Springer-Verlag, Berlin, Heidelberg, 1987.