A Denotational Semantics for Total Correctness of Sequential Exact Real Programs

Thomas Anberrée

Dr. Martín Hötzel Escardó
Supervisor

Dr. Alex Simpson
External Examiner

Dr. Steve Vickers
Internal Examiner

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
The University of Birmingham
October 2007

Abstract

We provide a denotational model for a functional programming language for exact real number computation. A well known difficulty in real number computation is that the tests $x = y$ and $x \leq y$ are undecidable and hence cannot be used to control the execution flow of programs. One solution, proposed by Boehm and Cartwright, is to use a non-deterministic test. For any two rational numbers $p < q$ and any real number x, at least one of the relations $p < x$ or $x < q$ can be determined to hold; thus, an operator rtest is used, whose evaluation never diverges when x is a real number:

1. $\mathrm{rtest}_{p,q}(x)$ evaluates to true or to false,
2. $\mathrm{rtest}_{p,q}(x)$ may evaluate to true iff $x < q$ and
3. $\mathrm{rtest}_{p,q}(x)$ may evaluate to false iff $p < x$.

Since a program can in general produce different results in different runs, Escardó and Marcial-Romero took the view in previous work that programs of real-number type denote sets of real numbers, and the question arose as to which power domains would be suitable for modelling the behaviour of rtest. It was shown that, among the known power domains, only the Hoare power domains are suitable. However, a semantics based on Hoare power domains only accounts for partial correctness of programs.

We argue that, although Smyth power domains cannot faithfully model the $\mathrm{rtest}_{p,q}$ operator, they can nevertheless provide an approximation which is sufficiently precise to define many common first-order functions, and has the further advantage over other choices that total correctness of programs is expressed in the model. Writing $[\![M]\!]$ and $[M]$ for the denotational and operational meaning of a program M, respectively, we show that, in general, one has $[\![M]\!] \sqsubseteq [M]$, but not necessarily $[\![M]\!] = [M]$. However, if $[\![M]\!]$ represents a real number, then $[\![M]\!]$ is maximal, and hence $[\![M]\!] = [M]$. To illustrate the usefulness of the approximate semantics in practice, we show that many programs of interest have a total denotation, which allows to establish their total correctness without resorting to operational methods. Moreover, we show that all computable first-order functions on the reals are definable in the language with respect to our denotational semantics.

A Julie et Auguste.

Acknowledgements

At the end of my Master at the University of Paris VII, my supervisor Chantale Berline introduced me to the work of Martín Escardó. At the time, John Longley was there as a visitor and we had valuable discussions together. John and Chantale provided me with encouragements and references to do my PhD under the supervision of Martín Escardó, at the University of Birmingham. I thank them both for this opportunity, as well as Martín and Achim Jung who so amicably welcomed me at the School of Computer Science in the summer 2003. Be they all reassured that I have no regrets about this move in my life!

Martín Escardó has been a wonderful supervisor who gave me much support and dedicated a lot of time to give me advice and never refused to answer my questions. I benefited a lot from our discussions and his patience when teaching me mathematics. Martín has obviously a vast knowledge, a clear and very focused mind, and a rare ability to convey deep ideas in a simple manner. Learning from him has been a great pleasure.

I am extremely thankful to Achim Jung for his support and help, not only as a mentor in my research but also for the help at a more personal level that he and his family provided to my wife and me.

I have worked among the Theory Group at the School of Computer Science. I thank all its members and in particular Paul Levy for his unstinting help.

Two former members of the Theory Group were Weng Kin Ho and Jose Raymundo Marcial Romero, who also obtained their PhD under the supervision of Mártin Escardó. Although I didn't get a chance to know Jose Raymundo well, I am very grateful to him for the work he has done, on which the present work is based. I got to know Weng Kin much better as we shared the same office for three years. I have the happiest memories of his presence and the discussions we had.

Finally, to my two examiners, Alex Simpson and Steve Vickers: many thanks for accepting to examine this work at such short notice and many thanks for your very helpful comments.

P.S. To Dan Ghica: thank you for the pictures.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We provide a domain-based denotational semantics for a sequential language for exact real number computation with a non-deterministic test operator. The semantics is only an approximate one, as it may not faithfully describe the behaviour of all programs. However, for each first-order computable total function $f\colon \mathbb{R}^n \to \mathbb{R}$, there exists a program for f whose denotation does faithfully describe the behaviour of the program.

A long known problem when computing with real numbers is that the test $0 < x$ is not decidable. If the input x is 0 and passed to a device in the form of the infinite sequence of digits $0.00000000000\ldots$ then the device has to read infinitely many digits in order to detect that its input represents 0. This phenomenon occurs with all reasonable representations of the real numbers [28]. Hence, one cannot for instance define the absolute value as

$$\mathrm{abs}\,(x) = \mathtt{if}\; x < 0 \;\mathtt{then}\; -x \;\mathtt{else}\; x,$$

because it is undefined for $x = 0$:

$$\mathrm{abs}\,(0) = \mathtt{if}\; \bot \;\mathtt{then}\; -0 \;\mathtt{else}\; 0 = \bot.$$

One way around this difficulty is to use a parallel conditional $\mathtt{pif}\ldots\mathtt{then}\ldots\mathtt{else}\ldots$ instead of the usual, sequential $\mathtt{if}\ldots\mathtt{then}\ldots\mathtt{else}\ldots$, and write

$$\mathrm{abs}\,(x) = \mathtt{pif}\; x < 0 \;\mathtt{then}\; -x \;\mathtt{else}\; x.$$

This works because

$$\mathtt{pif}\; \bot \;\mathtt{then}\; x \;\mathtt{else}\; x = x,$$

and hence

$$\mathrm{abs}\,(0) = \mathtt{pif}\; \bot \;\mathtt{then}\; -0 \;\mathtt{else}\; 0 = 0.$$

This approach was proposed by Boehm and Cartwright [2] and investigated in [11, 7]. But this is an unsatisfactory solution as it requires parallel evaluation of the condition and the branches and causes an exponential growth of parallel processes in recursive definitions.

A more efficient solution, also proposed by Boehm and Cartwright [2], and investigated in [3, 4, 16], is based on the following observation. For any p and q in $\mathbb{R}$ such that $p < q$ and any $x \in \mathbb{R}$, at least one of the two inequalities $p < x$ and $x < q$ can be determined to hold. Tests on real numbers can thus be performed by *redundant test* operators, written $\texttt{rtest}_{p,q}$, which behave as follows:

1. $\texttt{rtest}_{p,q}(x)$ evaluates to true or to false,
2. $\texttt{rtest}_{p,q}(x)$ may evaluate to true iff $x < q$, and
3. $\texttt{rtest}_{p,q}(x)$ may evaluate to false iff $p < x$.

In this work we consider a functional programming language which uses sequences of nested intervals as representations of real numbers [24, 6] and is essentially the language LRT described by Marcial-Romero in his thesis [16]. A computation of a real number is a sequence of reductions of the form

$$\ldots \to \texttt{bound}_{[p_0,q_0]}\,(M_0) \to \texttt{bound}_{[p_1,q_1]}\,(M_1) \to \ldots \to \texttt{bound}_{[p_i,q_i]}\,(M_i) \to \ldots$$

where $[p_i, q_i]$ are nested intervals with rational endpoints, converging to the real number x being evaluated. At any step, the operational semantics can easily check whether the current interval $[p_i, q_i]$ satisfies $q_i < q$, which entails $x < q$, or satisfies $p < p_i$, which entails $p < x$.

In contrast with the partial test $x < 0$ discussed above, the tests $\texttt{rtest}_{p,q}(x)$ are defined at all real numbers. Furthermore, expressions of the form

$$\texttt{if}\,\texttt{rtest}_{p,q}(x)\,\texttt{then}\,M_0\,\texttt{else}\,M_1$$

are evaluated sequentially:

1. First evaluate x, at until either $p < x$ or $x < q$ is detected,
2. then evaluate $\texttt{rtest}_{p,q}(x)$ to either true or false (using the outcome of 1),
3. then evaluate the unique branch given by step 2, among M_0 and M_1.

It is convenient to write

$$\begin{array}{rrcl} \texttt{cases} & x < q & \to & M_0 \\ & p < x & \to & M_1 \end{array}$$

rather than $\texttt{if}\,\texttt{rtest}_{p,q}(x)\,\texttt{then}\,M_0\,\texttt{else}\,M_1$. One has to bear in mind that the conditions of this cases construct are overlapping in general and that a non-deterministic choice is made by the operational semantics as to which branch to follow. In practice, implementations of the operational semantics can resolve the non-determinism by *e.g.* giving

priority to one of the branches. In any case, however, which of the conditions $p < x$ or $x < q$ is found first also depends on the particular sequence of intervals that implements x.

One program may evaluate to inconsistent results upon different runs, even if all computations lead to total results (that results denoting real numbers). For instance, consider the program

$$\begin{array}{lrcl} \texttt{cases} & \frac{1}{2} < 1 & \to & 4 \\ & 0 < \frac{1}{2} & \to & 5, \end{array}$$

whose possible outputs are 4 and 5.

However, programs with consistent, total outputs along all reduction paths can be defined. For example, the following expression is a program for the absolute value function at real numbers:

$$\begin{array}{lrcl} \operatorname{abs}(x) = \texttt{cases} & x < 0 & \to & -x \\ & -1 < x & \to & \texttt{cases} \quad x < 1 \quad \to \quad \texttt{bound}_{[-1,1]}\left(\frac{1}{2}\operatorname{abs}(2x)\right) \\ & & & \qquad\quad 0 < x \quad \to \quad x. \end{array} \tag{1.0.1}$$

If $x \leq -1$ or $x \geq 1$, the evaluation of $\operatorname{abs}(x)$ is deterministic and immediately leads to $|x|$. If $x \in (-1, 0) \cup (0, 1)$, different reduction paths may be followed but they all lead to the correct result (this is proved in Chapter 5). If $x = 0$ then only one branch can be followed in each cases construct and we have that

$$\operatorname{abs}(0) = \texttt{bound}_{[-1,1]}\left(\frac{1}{2}\operatorname{abs}(0)\right).$$

We will see that this leads to $\operatorname{abs}(0) = 0$, as desired.

A very active and successful trend in computer science is the search of semantic models for programming languages in certain categories of domains. Seminal work in this area is that of Scott [23, 22, 24]. In order to provide a model to the non-deterministic language they considered, Marcial-Romero and Escardó took the view that expressions such as $\texttt{rtest}_{0,1}(x)$ denote sets of results and the question arose as to which power domains are suitable to model the behaviour of $\texttt{rtest}_{0,1}$ [16, 17]. One of their results is that, among the known power domains, only the Hoare power domain allows a continuous interpretation of the `rtest` construct. Although one would like to see the expression $\texttt{rtest}_{0,1}\left(\frac{1}{2}\right)$ as denoting the set $\{\text{true}, \text{false}\}$, this set is not part of the Hoare semantics, because this semantics doesn't distinguish between programs that may diverge and programs that must converge. Indeed, the denotation of $\texttt{rtest}_{0,1}\left(\frac{1}{2}\right)$ is the set $\{\bot, \text{true}, \text{false}\}$, where the element $\bot$ denotes a divergent computation. For this reason, the Hoare semantics cannot be used to show that a program converges. It can however account for *partial* correctness of programs: the set of total values that a program may compute is uniquely determined by the denotation of the program in the Hoare semantics. For example, we can deduce from the denotation of the program $\texttt{rtest}_{0,1}\left(\frac{1}{2}\right)$ that it may reduce to true and to false.

In order to have a model that accounts for total correctness, we consider Smyth power domains. In these domains, it is impossible to give a continuous interpretation of `rtest` that perfectly describes its behaviour at real numbers, as shown by Marcial-Romero and Escardó. However, we define a continuous function, $\mathrm{ttest}_{0,1}$, that approximates the behaviour of the $\mathrm{rtest}_{0,1}$ operator at real numbers. For all $r \in \mathbb{R} \setminus \{0, 1\}$, we have that

$$\mathrm{ttest}_{0,1}(r) = \mathrm{rtest}_{0,1}(r)$$

but

$$\mathrm{ttest}_{0,1}(0) = \mathrm{ttest}_{0,1}(1) = \{\mathrm{true}, \mathrm{false}\}$$

whereas, operationally,

$$\texttt{rtest}_{0,1}(0) \text{ may only reduce to } \mathrm{true}$$

and

$$\texttt{rtest}_{0,1}(1) \text{ may only reduce to } \mathrm{false}\,.$$

Interpreting $\texttt{rtest}_{0,1}$ as $\mathrm{ttest}_{0,1}$, we obtain a denotational semantics with the following properties:

1. If a program of ground type has a maximal denotation, then the program computes its denotation, and

2. a large class of computable partial functions $\mathbb{R}^n \to \mathbb{R}$, including all computable total functions, can be defined by programs that have total denotations.

For example, the denotation of the program for the absolute value we gave earlier satisfies

$$[\![\mathrm{abs}]\!]\,(\eta\,(r)) = \eta\,(|r|)$$

for all real numbers r, where $\eta\,(r)$ is the element representing r in the model.

An important step in proving that our denotation satisfies such properties is to formulate and prove an adequacy result at total elements. Writing $[\![M]\!]$ and $[M]$ for the denotational and operational meaning of a program M, respectively, we show that, in general, one has $[\![M]\!] \sqsubseteq [M]$, but not necessarily $[\![M]\!] = [M]$. However, if $[\![M]\!]$ represents a real number, then $[\![M]\!]$ is maximal, and hence $[\![M]\!] = [M]$. Because we only have $[\![M]\!] \sqsubseteq [M]$ in general, we refer to our model as an *approximate semantics*.

Of course, our adequacy result at total elements is only useful as long as there are programs with total denotations. Our main contribution is not so much the adequacy result but the realization that our model can be used to define a large class of programs and prove their total correctness.

1.1 Organization

Chapters 7–6 are our own work, as is part of Chapter 3, as discussed below.

Chapter 2

We briefly describe the language PCF and its Scott model, as the language we consider is an extension of this language. The material of this chapter is standard.

Chapter 3

We describe an extension of PCF with a type `real` for real numbers, in particular using an `rtest` operator. The language thus obtained is essentially LRT [16], which in turn is Real PCF [6] with the parallel-conditional removed and `rtest` added. But our presentation is different, based on ideas in lecture notes by Escardó [8] that are equivalent to the ones of LRT but allow a simpler description and treatment of the language. However, the definition of the operational meaning of programs of ground types, via fair reduction paths rather than fair strategies, is part of our contribution.

Chapter 4

We describe an approximate semantics based on Smyth power domains for the language. In particular we define a continuous function ttest to interpret the `rtest` constructor. We then establish adequacy for total elements.

Chapter 5

We define what it means for a first-order continuous function in the model to *represent* a partial function. A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is then said to be *definable* if a program has a denotation that represents the function f. We use the approximate Smyth semantics to define common first-order functions, as well as operators for the limit of certain sequences and to find the root of certain numerical functions.

Chapter 6

Using the results of the previous chapter, we prove that all first-order computable functions $\mathbb{R}^n \to \mathbb{R}$ defined on recursively open subsets of $\mathbb{R}^n$ are definable in the language.

Chapter 7

We extend a result by Escardó, Hofmann and Streicher [9] which shows that continuous domains do not provide a proper environment to model deterministic, real number computation.

Chapter 2

The language PCF and its Scott model

Before considering real number computation in the next chapters, we describe the framework on which we base our approach. All the material presented in this chapter is standard.

PCF is a functional programming language with types for natural numbers, boolean values and functions. The Scott model of PCF is a mathematical structure in which PCF expressions are interpreted. A closed PCF term of ground type is given two meanings: a *denotational meaning*, or denotation, and an *operational meaning,* which describes its output upon evaluation. An important result is the *adequacy* between those two meanings: they are the same. This allows one to reason about programs by considering their denotation, in particular to prove their correctness.

In the next chapters, we will extend PCF and its Scott model to real numbers, with the aim of preserving some properties such as adequacy between the language and its model, expressivity of the language, and sequentiality. However, we will see that these properties must be somewhat relaxed in order to coexist in the new setting. Our notation and presentation mostly borrow from Streicher's book [26], to which the reader is referred for further details.

2.1 PCF

PCF (Programming language for Computable Functionals) was introduced in 1977 by Plotkin [19] and is based on the Logic for Computable Functions (LCF), introduced by Scott [22]. It is a typed lambda calculus with constants, whose types are given by

$$\sigma := \mathtt{nat} \mid \mathtt{bool} \mid (\sigma \to \sigma).$$

Atomic types such as `nat` and `bool` are called *ground types*. Other types are *function types*. We define the order of a type as follows:

$$\begin{aligned}&\mathrm{order}(\gamma) = 0, \text{ for ground types } \gamma \text{ and}\\ &\mathrm{order}(\sigma \to \tau) = \max\big(1 + \mathrm{order}(\sigma), \mathrm{order}(\tau)\big).\end{aligned}$$

The syntax of the language PCF is described in Table 2.1 on the following page by inductive rules defining the set of typing judgements. A *context* is an expression of the form $x_1 \colon \sigma_1, \ldots, x_k \colon \sigma_k$ where the x_i are pairwise distinct variables and the σ_i are types. A *typing judgement* is an expression of the form $\Gamma \vdash M \colon \sigma$, read "term M has type σ in context Γ", where Γ is a context and M is, by definition, a *term*. Note that type σ is uniquely determined by context Γ and term M. A term associated to the empty context is said to be a *closed term*, or *program*. If M is a term, we write $M \colon \sigma$ to indicate that it has type sigma in some context, but notice that if M is a closed term, the type σ of M is uniquely determined. We will often write $M \colon \sigma$ or just M without mentioning the context in which M is typed, in particular when M is a closed term.

The operational machinery that makes PCF a programming language is described by a decidable relation between closed terms. A closed term can be reduced to a closed term of same type according to the *one-step reduction relation* $\to$ defined by the rules given in Table 2.2 on page 10. An expression of the form $M \to M'$ reads "term M reduces (or is reduced) to term M' in one step." Each rule is of the form

$$(\rho)\frac{N \to N'}{M \to M'}C \quad \text{or} \quad (\rho)\frac{}{M \to M'}C$$

where (ρ) is a label to name the rule schema and C is a condition on M for the rule to be applicable (Such conditions are not used for PCF but will be used when we extend the language to a real type, in the next chapter). The expression $N \to N'$ is the *premise* of the rule while $M \to M'$ is its *conclusion*. A rule with no premise is called an *axiom*. A *derivation* (or a *proof*) that a term M reduces in one step to a term M' is a non-empty, finite sequence of (instances of schemata of) rules such that the first one is an axiom, the conclusion of each rule but the last is the premise of the next one and the conclusion of the last one is $M \to M'$. Such a derivation is conveniently represented as follows:

$$(\rho)\cfrac{(\rho_1)\cfrac{(\rho_0)\cfrac{}{M_0 \to M_0'}}{M_1 \to M_1'}}{\vdots}\cfrac{}{M \to M'}.$$

We write $M[N_0/x_0, \ldots, N_k/x_k]$ for the term obtained from M after substitution of the free occurrences of variable x_i by the closed term N_i, for each $i \in \{0, \ldots, k\}$.

Table 2.1: Typing judgements for PCF.

$$\frac{}{\Gamma, x\colon \sigma \vdash x\colon \sigma} \qquad \frac{\Gamma \vdash M\colon \sigma \to \tau \quad \Gamma \vdash N\colon \sigma}{\Gamma \vdash M(N)\colon \tau}$$

$$\frac{\Gamma, x\colon \sigma \vdash M\colon \tau}{\Gamma \vdash (\lambda x\colon \sigma.M)\colon \sigma \to \tau} \qquad \frac{\Gamma \vdash M\colon \sigma \to \sigma}{\Gamma \vdash \mathtt{Y}_\sigma(M)\colon \sigma}$$

$$\frac{}{\Gamma \vdash \mathtt{true}\colon \mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{false}\colon \mathtt{bool}}$$

$$\frac{\Gamma \vdash L\colon \mathtt{nat}}{\Gamma \vdash (0 =)\,(L)\,:\mathtt{bool}}$$

$$\frac{\Gamma \vdash L\colon \mathtt{bool}, \quad \Gamma \vdash M\colon \gamma, \quad \Gamma \vdash N\colon \gamma}{\Gamma \vdash \mathtt{if}_\gamma(L, M, N)\colon \gamma} \quad \text{for } \gamma \in \{\mathtt{bool}, \mathtt{nat}\}$$

$$\frac{}{\Gamma \vdash \underline{n}\colon \mathtt{nat}} \quad \text{for all } n \in \mathbb{N}$$

$$\frac{\Gamma \vdash M\colon \mathtt{nat}}{\Gamma \vdash \mathtt{succ}(M)\colon \mathtt{nat}}$$

$$\frac{\Gamma \vdash M\colon \mathtt{nat}}{\Gamma \vdash \mathtt{pred}(M)\colon \mathtt{nat}}$$

Table 2.2: The reduction relation $\to$ for PCF.

(1) $\dfrac{}{\mathtt{Y}_\sigma(M) \to M(\mathtt{Y}_\sigma(M))}$ (2) $\dfrac{}{(\lambda x\colon \sigma.M)(N) \to M[N/x]}$

(3) $\dfrac{M \to M'}{M(N) \to M'(N)}$

(4) $\dfrac{}{\mathtt{if}(\mathtt{true}, M, N) \to M}$ (5) $\dfrac{}{\mathtt{if}(\mathtt{false}, M, N) \to N}$

(6) $\dfrac{B \to B'}{\mathtt{if}(B, M, N) \to \mathtt{if}(B', M, N)}$

(7) $\dfrac{}{(0 =)(\underline{0}) \to \mathtt{true}}$ (8) $\dfrac{}{(0 =)(\underline{n+1}) \to \mathtt{false}}$ (9) $\dfrac{L \to L'}{(0 =)(L) \to (0 =)(L')}$

(10) $\dfrac{}{\mathtt{succ}(\underline{n}) \to \underline{n+1}}$ (11) $\dfrac{M \to M'}{\mathtt{succ}(M) \to \mathtt{succ}(M')}$

(12) $\dfrac{}{\mathtt{pred}(\underline{0}) \to \underline{0}}$ (13) $\dfrac{}{\mathtt{pred}(\underline{n+1}) \to \underline{n}}$ (14) $\dfrac{M \to M'}{\mathtt{pred}(M) \to \mathtt{pred}(M')}$

2.1.1 Sequentiality

Sequentiality is a hard concept to make mathematically precise but is quite an intuitive one. PCF is seen as intrinsically sequential, because each reduction step depends only on the reduction of a uniquely determined sub-term of the term to be reduced. Another way of putting this is that reducing a term only requires one thread of computation. A typical example of a programming language that is not sequential is PCF extended with the weak parallel operator `wpor`, which takes two arguments of type `bool`. A program of the form $\texttt{wpor}(M, N)$ evaluates to `true` if and only if either of its argument, M or N, evaluates to `true`. The reduction rules for `wpor` are

$$\frac{M \to \texttt{true}}{\texttt{wpor}\,(M, N) \to \texttt{true}} \text{ and } \frac{N \to \texttt{true}}{\texttt{wpor}\,(M, N) \to \texttt{true}} \tag{2.1.1}$$

along with the congruence rules

$$\frac{M \to M'}{\texttt{wpor}\,(M, N) \to \texttt{wpor}\,(M', N)} \text{ and } \frac{N \to N'}{\texttt{wpor}\,(M, N) \to \texttt{wpor}\,(M, N')}. \tag{2.1.2}$$

To reduce $\texttt{wpor}(M, N)$, one can't make a choice as to which of M or N should be reduced first, as it could be that one diverges—*i.e.* doesn't reduce to `true` nor `false`—whereas the other one evaluates to `true`. It is known that the corresponding wpor defined by

$$\begin{aligned} &\mathrm{wpor}\colon \mathbb{B}_\perp \times \mathbb{B}_\perp \to \mathbb{B}_\perp \\ &\mathrm{wpor}(x, y) = \begin{cases} \mathrm{true} & \text{if } x = \mathrm{true} \text{ or } y = \mathrm{true} \\ \perp & \text{otherwise} \end{cases} \end{aligned} \tag{2.1.3}$$

in the Scott model (Section 2.2) is not definable in PCF (see [26] for a proof).

2.2 The Scott model of PCF

Domains are particular ordered sets that are suitable to give a semantics to terms of some programming languages with recursive operators [23, 24]. The order can be seen as a way of comparing the information carried by terms. The motivating property satisfied by all domains is that least fixed point operators exists at all orders, allowing to interpret the recursion operators, and that divergent computations are also interpreted, by least elements. We only very briefly recall some definitions and main results and refer the reader to references [1, 20, 26, 12] for details. In this section, we describe the Scott model of PCF in term of pointed dcpo's. However, the properties mentioned here will carry through when we consider continuous dcpo's instead in the next chapters. All dcpo's considered in this thesis are continuous.

2.2.1 Directed complete partial orders

Given a partially ordered set $(D, \sqsubseteq)$, a subset A is said to be *directed* if it is not empty and, for any $a, b \in A$, possesses an upper bound of a and b, that is some element $c \in A$ such that $a \sqsubseteq c$ and $b \sqsubseteq c$. A *directed-complete partially ordered set* $(D, \sqsubseteq)$, dcpo for short, is a partially ordered set in which every directed subset A has a supremum, written $\bigsqcup A$. A dcpo D with a least element $\bot_D$ is called a pointed dcpo. Given a set X, the pointed dcpo obtained by adjoining an extra element $\bot_X$ to X and defining a partial order $\sqsubseteq_X$ by

$$\forall x, y \in X, \qquad x \sqsubseteq_X y \text{ if and only if } x = \bot_X \text{ or } x = y \tag{2.2.1}$$

is called the *lifting* of X and is written $X_\bot$. Figure 2.1 represents the Hasse diagrams of $\mathbb{N}_\bot$ and $\mathbb{B}_\bot$, where $\mathbb{N}$ is the set of natural numbers and $\mathbb{B}$ the set $\{\text{true}, \text{false}\}$ of boolean values. PCF programs of type `nat` and `bool` are interpreted in $\mathbb{N}_\bot$ and $\mathbb{B}_\bot$, respectively.

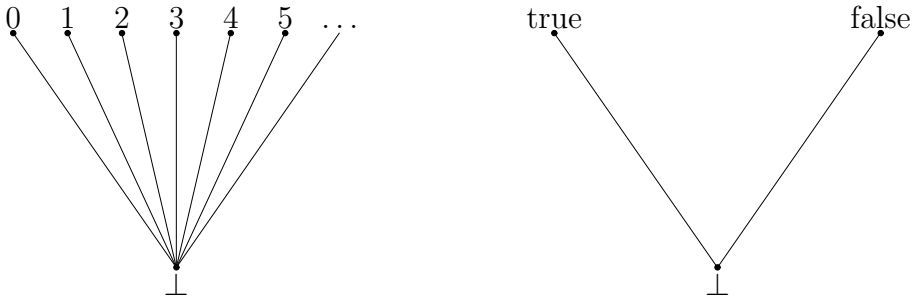


Figure 2.1: Hasse diagrams of $\mathbb{N}_\bot$ and $\mathbb{B}_\bot$.

2.2.2 Scott topology

The Scott closed subsets of a dcpo D are by definition the lower subsets of D that are closed under the formation of directed suprema. Their complements in D are the open sets of the Scott topology on D. For example, the non-trivial open subsets of $\mathbb{B}_\bot$ are $\{\text{true}\}$, $\{\text{false}\}$ and $\{\text{true}, \text{false}\}$. Similarly the non-trivial open subsets of $\mathbb{N}_\bot$ are the subsets of $\mathbb{N}$. The nice parallels between the topological and order-theoretical points of view is a prominent and very satisfying aspect of domain theory. At a basic level, one can retrieve the order on a dcpo from its Scott topology and vice-versa. The following is taken from [1, Section 2.3].

Theorem 2.2.1. *Let D be a dcpo.*

1. *For elements $x, y \in D$, the following are equivalent:*

(a) $x \sqsubseteq y$,

(b) *Every Scott open set which contains* x *also contains* y,

(c) $x \in \mathrm{Cl}(\{y\})$, *where* $\mathrm{Cl}(\{y\})$ *is the closure of* $\{y\}$.

2. *Let* A *be a directed subset of* D. *Then* $x \in D$ *is the supremum of* A *if and only if it is an upper bound for* A *and every Scott neighbourhood of* x *contains an element of* A.

The spaces of natural numbers $\mathbb{N}$ and booleans $\mathbb{B}$ with the discrete topology are embedded in $\mathbb{N}_\perp$ and $\mathbb{B}_\perp$ respectively via the continuous functions

$$\begin{aligned} &\eta_{\mathbb{N}} \colon \mathbb{N} \to \mathbb{N}_\perp \\ &\eta_{\mathbb{N}}\,(n) = n \end{aligned} \tag{2.2.2}$$

and

$$\begin{aligned} &\eta_{\mathbb{B}} \colon \mathbb{B} \to \mathbb{B}_\perp \\ &\eta_{\mathbb{B}}\,(b) = b \end{aligned}$$

and form the subspaces of maximal elements in $\mathbb{N}_\perp$ and $\mathbb{B}_\perp$ respectively.

2.2.3 Continuous functions and products

A function $f \colon D \to E$ between dcpo's is Scott continuous if it is monotone and preserves directed suprema, that is $f(\bigsqcup A) = \bigsqcup \{f(a) \mid A\}$, for all directed subsets A of D. The match with the topological counterpart is ensured by the following theorem [1, Section 2.3.1]

Theorem 2.2.2. *For dcpo's* D *and* E, *a function from* D *to* E *is Scott continuous if and only if it is topologically continuous with respect to the Scott topologies on* D *and* E.

If D and E are dcpo's the set of Scott continuous functions, ordered by the pointwise ordering of functions inherited from the order on E, is itself a dcpo, written $[D \to E]$.

Given dcpo's $D_0, \ldots, D_k$, their set-theoretical product $D_0 \times \ldots \times D_k$, ordered coordinatewise, is again a dcpo and the usual set-theoretical projections are Scott continuous. In fact, pointed dcpo's and Scott continuous functions form a cartesian closed category in which the categorical products and function spaces are the same as the ones just described [1].

Theorem 2.2.3. *Let* D, E *and* F *be dcpo's.*

1. *A function* $D \times E \to F$ *is Scott continuous if and only if it is Scott continuous in each variable, separately.*

2. *The application operator* $([D \to E] \times D) \to E$ *that maps* (f, d) *to* $f(d)$ *is continuous.*

3. *The composition operator is a Scott continuous function from* $[D \to E] \times [E \to F]$ *to* $[D \to F]$.

4. *For any* $f \in [D \times E \to F]$, *the function* $\mathrm{curry}\,(f) : D \to [E \to F]$ *defined by* $\mathrm{curry}(f)(d)(e) = f(d, e)$ *is continuous and the* curry *operator thus defined is continuous.*

2.2.4 Least fixed point operators

Let $(D, \sqsubseteq)$ be a dcpo with a least element $\bot_D$ and $f \in [D \to D]$ a Scott continuous function. The supremum

$$\mu_D\,(f) = \bigsqcup f^n(\bot_D) \tag{2.2.3}$$

of all the iterates of f applied to $\bot_D$ exists, because the $\{f^n\,(\bot_D) \mid n \in \mathbb{N}\}$ is directed, by monotonicity of f. It is the least fixed point of f, that is the least element $d \in D$ such that $f(d) = d$.

Theorem 2.2.4. *The operator* $\mu_D \colon [D \to D] \to D$ *is Scott continuous, so belongs to the dcpo* $[[D \to D] \to D]$.

2.2.5 Denotational semantics of PCF

The denotation of types is defined inductively by

$$[\![\mathtt{nat}]\!] = D_{\mathtt{nat}} = \mathbb{N}_\bot, \tag{2.2.4}$$
$$[\![\mathtt{bool}]\!] = D_{\mathtt{bool}} = \mathbb{B}_\bot \text{ and}$$
$$[\![\sigma \to \tau]\!] = D_{\sigma\to\tau} = \big[\,[\![\sigma]\!] \to [\![\tau]\!]\,\big]. \tag{2.2.5}$$

A typing judgement $x_0 \colon \sigma_0, \ldots, x_k \colon \sigma_k \vdash M \colon \sigma$ is then interpreted as an element of the dcpo $[D_{\sigma_0} \times \ldots \times D_{\sigma_k} \to D_\sigma]$. The denotation of typing judgement is defined inductively by the rules in Table 2.3 on the following page. The first four definitions in this table are justified by Theorems 2.2.4 and 2.2.3.

Constructors `succ`, `pred` and $(0 =)$ are interpreted using the following functions that

Table 2.3: Denotation of typing judgements for PCF.

$$[\![x_1\colon \sigma_1, \ldots, x_k\colon \sigma_k \vdash x_i\colon \sigma_i]\!]\,(\overrightarrow{d}\,) = d_i$$

$$[\![\Gamma \vdash (\lambda x\colon \sigma.M)\colon \sigma \to \tau]\!]\,(\overrightarrow{d}\,)(e) = [\![\Gamma, x\colon \sigma \vdash M\colon \tau]\!]\,(\overrightarrow{d}\,, e)$$

$$[\![\Gamma \vdash \mathtt{Y}_\sigma(M)\colon \sigma]\!]\,(\overrightarrow{d}\,) = \mu_{D_\sigma}\Big([\![\Gamma \vdash M\colon \sigma \to \sigma]\!]\,(\overrightarrow{d}\,)\Big)$$

$$[\![\Gamma \vdash M(N)\colon \tau]\!]\,(\overrightarrow{d}\,) = [\![\Gamma \vdash M\colon \sigma \to \tau]\!]\,(\overrightarrow{d}\,)\Big([\![\Gamma \vdash N\colon \sigma]\!]\,(\overrightarrow{d}\,)\Big)$$

$$[\![\Gamma \vdash \mathtt{true}\colon \mathtt{bool}]\!]\,(\overrightarrow{d}\,) = \text{true}$$

$$[\![\Gamma \vdash \mathtt{false}\colon \mathtt{bool}]\!]\,(\overrightarrow{d}\,) = \text{false}$$

$$[\![\Gamma \vdash (0 =)(M)\colon \mathtt{bool}]\!]\,(\overrightarrow{d}\,) = (0 =)\Big([\![\Gamma \vdash M\colon \mathtt{nat}]\!]\,(\overrightarrow{d}\,)\Big)$$

$$[\![\Gamma \vdash \mathtt{if}_\gamma(B, M, N)\colon \gamma]\!]\,(\overrightarrow{d}\,) = \begin{cases} [\![\Gamma \vdash M\colon \gamma]\!]\,(\overrightarrow{d}\,) & \text{if } [\![\Gamma \vdash B\colon \mathtt{bool}]\!]\,(\overrightarrow{d}\,) = \text{true} \\ [\![\Gamma \vdash N\colon \gamma]\!]\,(\overrightarrow{d}\,) & \text{if } [\![\Gamma \vdash B\colon \mathtt{bool}]\!]\,(\overrightarrow{d}\,) = \text{false} \\ \bot_{D_\gamma} & \text{otherwise} \end{cases}$$

$$[\![\Gamma \vdash \underline{n}\colon \mathtt{nat}]\!]\,(\overrightarrow{d}\,) = n$$

$$[\![\Gamma \vdash \mathtt{succ}(M)\colon \mathtt{nat}]\!]\,(\overrightarrow{d}\,) = \text{succ}\Big([\![\Gamma \vdash M\colon \mathtt{nat}]\!]\,(\overrightarrow{d}\,)\Big)$$

$$[\![\Gamma \vdash \mathtt{pred}(M)\colon \mathtt{nat}]\!]\,(\overrightarrow{d}\,) = \text{pred}\Big([\![\Gamma \vdash M\colon \mathtt{nat}]\!]\,(\overrightarrow{d}\,)\Big)$$

are easily shown to be Scott continuous.

$$\begin{aligned}
&\mathrm{succ}\colon \mathbb{N}_\perp \to \mathbb{N}_\perp \\
&\mathrm{succ}(n) = \begin{cases} n+1 & \text{if } n \in \mathbb{N} \\ \perp_{\mathbb{N}_\perp} & \text{otherwise} \end{cases} \\
&\mathrm{pred}\colon \mathbb{N}_\perp \to \mathbb{N}_\perp \\
&\mathrm{pred}(n) = \begin{cases} 0 & \text{if } n = 0 \\ n-1 & \text{if } n \in \mathbb{N} \setminus \{0\} \\ \perp_{\mathbb{N}_\perp} & \text{otherwise} \end{cases} \\
&0 =\colon \mathbb{N}_\perp \to \mathbb{B}_\perp \\
&0 =(n) = \begin{cases} \mathrm{true} & \text{if } n = 0 \\ \mathrm{false} & \text{if } n \in \mathbb{N} \setminus \{0\} \\ \perp_{\mathbb{N}_\perp} & \text{otherwise} \end{cases}
\end{aligned} \tag{2.2.6}$$

We more often speak of the denotation of a term rather than that of a typing judgement. This is justified by the fact that, for any term, there exists a unique context of minimal length in which this term can be typed. For closed terms, the minimal context is the empty context (a closed term being by definition one typed in the empty context).

Total and partial elements

For a ground type γ, an element in D_γ is said to be *total* if it is a maximal element for the information order. Hence, all natural numbers are total in $\mathbb{N}_\perp$ but $\perp$ is not. A continuous function $f\colon D_\sigma \to D_\tau$ is total if $f(x)$ is total in D_τ whenever x is total in D_σ. Similarly, a closed term is total if its denotation is total. Notice that in $\mathbb{N}_\perp$ and $\mathbb{B}_\perp$, total elements are all denotations of values.

Operational meaning of programs and adequacy

The operational semantics of PCF has been given via a reduction relation $\to$. A *computation* from a term M_0 is a maximal sequence of terms (M_n) whose first term is M_0 and such that each term reduces to the next one:

$$M_0 \to M_1 \to \ldots \to M_i \to M_{i+1} \to \ldots \tag{2.2.7}$$

The reduction relation is deterministic, in the sense that a term may only reduce to at most one other term in one step. So there is a unique computation from any given closed term. Furthermore, a computation from a closed term M of ground type is either infinite,

in which case we say that the computation *diverges*, or ends with a *value*, that is program `true`, `false` or $\underline{n}$, for some $n \in \mathbb{N}$, in which case we say that the computation *terminates* and that M *evaluates* to `true`, `false` or $\underline{n}$, respectively. The operational meaning $[M]$ of a program M is defined in the same dcpo as $[\![M]\!]$ and describes the outcome of the computation from M.

Definition 2.2.1. If M is a program of type `nat`, the *operational meaning* $[M]$ of M is the element of $\mathbb{N}_\perp$ defined by:

$$[M] = \begin{cases} n & \text{if the computation from } M \text{ terminates with value } \underline{n}, \text{ for some } n \in \mathbb{N} \\ \perp_{\mathbb{N}_\perp} & \text{otherwise.} \end{cases} \tag{2.2.8}$$

Similarly, if M is a program of type `bool`, the *operational meaning* $[M]$ of M is the element of $\mathbb{B}_\perp$ defined by:

$$[M] = \begin{cases} \text{true} & \text{if the computation from } M \text{ terminates with value } \texttt{true} \\ \text{false} & \text{if the computation from } M \text{ terminates with value } \texttt{false} \\ \perp_{\mathbb{B}_\perp} & \text{otherwise.} \end{cases} \tag{2.2.9}$$

Adequacy

An important feature of PCF is the match between the operational meaning and denotational meaning of programs of ground type.

Theorem 2.2.5. *For any PCF program,* $[\![M]\!] = [M]$.

This was proven by Plotkin [19]. We can split the adequacy property in two inequalities:

1. completeness: $[\![M]\!] \sqsubseteq [M]$,
2. correctness: $[\![M]\!] \sqsupseteq [M]$.

We refer to these properties as completeness and correctness because they are equivalent to the traditional definitions of completeness and correctness with regard to the language PCF and its Scott model (*cf.* [26]). Here are some details showing that inequality 1 entails completeness and inequality 2 entails correctness.

Completeness

Suppose a program M of type `nat` denotes a natural number m, that is $[\![M]\!] = m$. The first inequality above, namely $[\![M]\!] \sqsubseteq [M]$, entails that $m \sqsubseteq [M]$, and hence that $[M] = m$, because m is a maximal element in $\mathbb{N}_\perp$. Therefore, by definition of $[M]$, the program M evaluates to $\underline{m}$. In short:

$$[\![M]\!] = m \implies M \to^* \underline{m}. \tag{2.2.10}$$

This property is called *completeness* of the operational semantics with respect to the denotational semantics. It says that there are enough reduction rules to ensure that a program of ground type outputs what it is supposed to output.

Correctness

Suppose that M is a program of type `nat` which evaluates to $\underline{m}$, for some $m \in \mathbb{N}$. This means that $[M] = m$ and from the second inequality, namely $[\![M]\!] \sqsupseteq [M]$, it follows that the denotation of M is above m and hence $[\![M]\!] = m$, as m is maximal in $\mathbb{N}_\perp$. In short, we have that

$$M \to^* \underline{m} \implies [\![M]\!] = m. \tag{2.2.11}$$

This property is called *correctness* of the operational semantics with respect to the denotational semantics. It says that the reduction relation "preserves semantical equality" (see [26, p. 5]). Similar remarks apply to programs of type `bool`.

Chapter 3

PCF extended with real numbers

We extend the language PCF with a type for real numbers and related basic constructors, thus completing the description of the programming language considered in this thesis. Types are given by

$$\sigma = \texttt{real} \mid \texttt{nat} \mid \texttt{bool} \mid \sigma \to \sigma. \tag{3.0.1}$$

The typing judgements and reduction rules for the entire language are recapitulated in Table 3.1 and Table 3.2 respectively, on the next two pages. The expressions $|M| = \bot$, $|L| < 1$ and $0 < |L|$, to the right of certain new reduction rules, are side conditions indicating when these rules are applicable (*cf.* Remark 3.1.3, Section 3.1.3 on page 29).

In Section 3.1, we present basic constructors $\texttt{bound}_a$, $p+$ and $p\times$, which are used to approximate real numbers with intervals. The fragment of the language consisting of PCF augmented with these constructors is called the *weak fragment*, because its expressivity is very limited [10]. The operational semantics is defined via a reduction relation between closed terms and—as far as the weak fragment is concerned—is deterministic: any term in the weak fragment that reduces at all, reduces to a unique term (also in the weak fragment). The *output* of each *reduction path* is defined as an element of the domain $\mathbb{N}_\bot$, $\mathbb{B}_\bot$ or the *interval domain* $\mathcal{R}$ (Definitions 3.1.5, 3.1.4 and 3.1.1).

The full language is obtained by adding the `rtest` constructor to perform tests on real numbers (Section 3.1.3). Although it renders the language expressive, as we shall see in the next chapters, this is at the expense of introducing non-determinism.

In Section 3.2, we define the operational meaning of programs of ground type as elements of Smyth power domains. This section is original as our presentation is different from the one found in [17], which uses strategies whereas we use fair paths.

Table 3.1: Typing judgements.

$\dfrac{}{\Gamma, x\colon \sigma \vdash x\colon \sigma}$	$\dfrac{\Gamma \vdash M\colon \sigma \to \tau \qquad \Gamma \vdash N\colon \sigma}{\Gamma \vdash M(N)\colon \tau}$
$\dfrac{\Gamma, x\colon \sigma \vdash M\colon \tau}{\Gamma \vdash (\lambda x\colon \sigma.M)\colon \sigma \to \tau}$	$\dfrac{\Gamma \vdash M\colon \sigma \to \sigma}{\Gamma \vdash \mathtt{Y}_\sigma(M)\colon \sigma}$
$\dfrac{}{\Gamma \vdash \mathtt{true}\colon \mathtt{bool}}$	$\dfrac{}{\Gamma \vdash \mathtt{false}\colon \mathtt{bool}}$
$\dfrac{\Gamma \vdash L\colon \mathtt{nat}}{\Gamma \vdash (0 =)\,(L)\,:\mathtt{bool}}$	
$\dfrac{\Gamma \vdash L\colon \mathtt{bool}, \quad \Gamma \vdash M\colon \gamma, \quad \Gamma \vdash N\colon \gamma}{\Gamma \vdash \mathtt{if}_\gamma(L, M, N)\colon \gamma}$	$\gamma \in \{\mathtt{bool}, \mathtt{nat}, \mathtt{real}\}$
$\dfrac{}{\Gamma \vdash \underline{n}\colon \mathtt{nat}}$	for all $n \in \mathbb{N}$
$\dfrac{\Gamma \vdash M\colon \mathtt{nat}}{\Gamma \vdash \mathtt{succ}(M)\colon \mathtt{nat}}$	$\dfrac{\Gamma \vdash M\colon \mathtt{nat}}{\Gamma \vdash \mathtt{pred}(M)\colon \mathtt{nat}}$
$\dfrac{\Gamma \vdash M\colon \mathtt{real}}{\Gamma \vdash \mathtt{bound}_a(M)\colon \mathtt{real}}$	$a \in \mathcal{R}_\mathbb{Q}$, where $\mathcal{R}_\mathbb{Q} = \{[p,q] \mid p \le q, \quad p, q \in \mathbb{Q}\}$ (*cf.* remarks 3.1.1 and 3.1.2 on page 24)
$\dfrac{\Gamma \vdash M\colon \mathtt{real}}{\Gamma \vdash p + (M)\colon \mathtt{real}}$	$p \in \mathbb{Q}$
$\dfrac{\Gamma \vdash M\colon \mathtt{real}}{\Gamma \vdash p \times (M)\colon \mathtt{real}}$	$p \in \mathbb{Q}$
$\dfrac{\Gamma \vdash L\colon \mathtt{real}}{\Gamma \vdash \mathtt{rtest}(L)\colon \mathtt{bool}}$	

Table 3.2: Reduction relation.

(1) $\dfrac{}{\mathtt{Y}_\sigma(M) \to M(\mathtt{Y}_\sigma(M))}$

(2) $\dfrac{}{(\lambda x\colon \sigma.M)(N) \to M[N/x]}$

(3) $\dfrac{M \to M'}{M(N) \to M'(N)}$

(4) $\dfrac{}{\mathtt{if}(\mathtt{true}, M, N) \to M}$

(5) $\dfrac{}{\mathtt{if}(\mathtt{false}, M, N) \to N}$

(6) $\dfrac{B \to B'}{\mathtt{if}(B, M, N) \to \mathtt{if}(B', M, N)}$

(7) $\dfrac{}{(0 =)(\underline{0}) \to \mathtt{true}}$ (8) $\dfrac{}{(0 =)(\underline{n+1}) \to \mathtt{false}}$ (9) $\dfrac{L \to L'}{(0 =)(L) \to (0 =)(L')}$

(10) $\dfrac{}{\mathtt{succ}(\underline{n}) \to \underline{n+1}}$

(11) $\dfrac{M \to M'}{\mathtt{succ}(M) \to \mathtt{succ}(M')}$

(12) $\dfrac{}{\mathtt{pred}(\underline{0}) \to \underline{0}}$ (13) $\dfrac{}{\mathtt{pred}(\underline{n+1}) \to \underline{n}}$ (14) $\dfrac{M \to M'}{\mathtt{pred}(M) \to \mathtt{pred}(M')}$

(15) $\dfrac{}{\mathtt{bound}_a\left(\mathtt{bound}_b(M)\right) \to \mathtt{bound}_{\mathrm{bound}_a(b)}(M)}$

(16) $\dfrac{M \to M'}{\mathtt{bound}_a(M) \to \mathtt{bound}_a(M')} |M| = \bot$

(17) $\dfrac{}{p + \left(\mathtt{bound}_a(M)\right) \to \mathtt{bound}_{p+a}\left(p + (M)\right)}$ (18) $\dfrac{M \to M'}{p + (M) \to p + (M')} |M| = \bot$

(19) $\dfrac{}{p \times \left(\mathtt{bound}_a(M)\right) \to \mathtt{bound}_{p\times a}\left(p \times (M)\right)}$ (20) $\dfrac{M \to M'}{p \times (M) \to p \times (M')} |M| = \bot$

(21) $\dfrac{}{\mathtt{rtest}(L) \to \mathtt{true}} |L| < 1$

(22) $\dfrac{}{\mathtt{rtest}(L) \to \mathtt{false}} 0 < |L|$

(23) $\dfrac{L \to L'}{\mathtt{rtest}(L) \to \mathtt{rtest}(L')}$

3.1 Approximating real numbers with intervals

We add new constructors $\texttt{bound}_a$, $p+$ and $p\times$ to approximate real numbers with intervals and to operate on these approximations. The weak fragment of the language thus obtained is essentially the corresponding weak fragment of Real PCF, introduced by Escardó [7]. However the same author later proposed a more convenient set of basic operators in unpublished notes [8], which we use here.

Functions $\mathrm{bound}_{[r,s]} : \mathbb{R} \to \mathbb{R}$

Example

The equation $t = \frac{1}{2}t$ has a unique solution, the real number 0. However this equation doesn't bear any computational content in the sense that, when seen as a recursive definition of t, it unfolds as $t = \frac{1}{2}\left(\frac{1}{2}t\right) = \ldots = \frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\left(\ldots\right)\right)\right)\right)$, which doesn't display any information about the solution. In that sense, the equation $t = \frac{1}{2}t$ can be seen as a divergent program. We are going to modify equation $t = \frac{1}{2}t$ so as to give it more computational content.

For each interval of the form $[r, s]$, we define the following continuous function:

$$\begin{array}{l} \mathrm{bound}_{[r,s]} : \mathbb{R} \to \mathbb{R} \\ \mathrm{bound}_{[r,s]}(t) = \max\left(r, \min\left(s, t\right)\right) \end{array} \tag{3.1.1}$$

The number $\mathrm{bound}_{[r,s]}\left(t\right)$ is the one in the interval $[r, s]$ that is the closest to t. It is t itself if t is already in $[r, s]$. The graph of $\mathrm{bound}_{[r,s]}$ is represented in figure 3.1. Its range is the interval $[r, s]$ itself. If a number t is expressed as $t = \mathrm{bound}_{[r,s]}\left(\ldots\right)$, one knows at once that this number must be between r and s and the interval $[r, s]$ can be regarded as an approximation of t immediately readable from the expression $\mathrm{bound}_{[r,s]}\left(\ldots\right)$.

Let's now consider the equation $t = \mathrm{bound}_{[-1,1]}\left(\frac{1}{2}t\right)$ instead of the previous $t = \frac{1}{2}t$. One immediately reads off the new equation that a solution must be in the interval $[-1, 1]$, the range of the function $\mathrm{bound}_{[-1,1]}$. After one unfolding, one gets

$$t = \mathrm{bound}_{[-1,1]}\left(\frac{1}{2}\,\mathrm{bound}_{[-1,1]}(t)\right) \tag{3.1.2}$$

We can reduce the right-hand side by using the straightforward equations

$$p \times \mathrm{bound}_{[r,s]}\left(t\right) = \mathrm{bound}_{p\times[r,s]}\left(p \times t\right) \text{ and} \tag{3.1.3}$$

$$\mathrm{bound}_{[r',s']}\left(\mathrm{bound}_{[r,s]}\left(t\right)\right) = \mathrm{bound}_{\mathrm{bound}_{[r',s']}([r,s])}\left(t\right) \tag{3.1.4}$$

where $p \times [r, s]$ and $\mathrm{bound}_{[r',s']}\left([r, s]\right)$ are the direct images of the interval $[r, s]$ through the functions $p \times -$ and $\mathrm{bound}_{[r',s']}$ respectively. They are themselves non-empty, closed

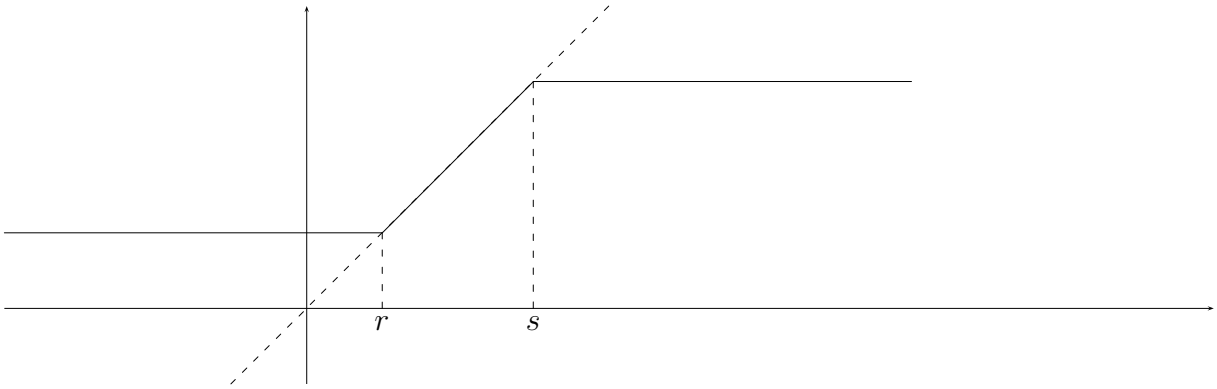


Figure 3.1: Graph of $\mathrm{bound}_{[r,s]} : \mathbb{R} \to \mathbb{R}$.

bounded intervals and can also be expressed as $[\min(p \times r, p \times s), \max(p \times r, p \times s)]$ and $[\mathrm{bound}_{[r',s']}(r), \mathrm{bound}_{[r',s']}(s)]$, respectively. After reducing equation 3.1.2, we obtain

$$t = \mathrm{bound}_{[-\frac{1}{2},\frac{1}{2}]}\left(\frac{1}{2}t\right) \tag{3.1.5}$$

and immediately read that the solution to this equation is in the interval $[\frac{1}{2}, \frac{1}{2}]$ which constitutes a better approximation than the preceding one, which was the interval $[-1, 1]$. Continuing unfolding and reducing in a similar manner, we get

$$t = \mathrm{bound}_{[-\frac{1}{2},\frac{1}{2}]}\left(\frac{1}{2}t\right) = \mathrm{bound}_{[-\frac{1}{4},\frac{1}{4}]}\left(\frac{1}{4}t\right) = \ldots = \mathrm{bound}_{[-\frac{1}{2^n},\frac{1}{2^n}]}\left(\frac{1}{2^n}t\right) = \ldots \tag{3.1.6}$$

reading better and better approximations (narrower and narrower nested intervals) at the head of each expression. The intersection of those intervals, though never reached in finitely many unfolding, is the interval $[0, 0]$, that is the singleton $\{0\}$.

Constructors $\mathtt{bound}_a$

Following those ideas, we add constructors $\mathtt{bound}_a$ to the language, one for each non-empty, closed and bounded interval a with rational bounds. Their syntax is governed by the typing rule

$$\frac{\Gamma \vdash M : \mathtt{real}}{\Gamma \vdash \mathtt{bound}_a(M) : \mathtt{real}} \tag{3.1.7}$$

given in Table 3.1 on page 20. We only consider intervals with rational end points because we need a recursive set of constructors. Thus, the set of non-empty, closed and bounded

intervals with rational end-points,

$$\mathcal{R}_{\mathbb{Q}} = \{[p,q] \mid p \leq q \text{ and } p,q \in \mathbb{Q}\} \tag{3.1.8}$$

is *the set of approximations of real numbers* that the language manipulates.

Remark 3.1.1. Notice that, because a programming language only manipulates formal symbols, we ought to consider formal pairs (p,q) of rational numbers rather than intervals $[p,q]$. We here choose to emphasize the intention rather than the formalism.

The following reduction rule concerning the `bound` constructors (*cf.* Table 3.2 on page 21)

$$\overline{\texttt{bound}_a\left(\texttt{bound}_b(M)\right) \to \texttt{bound}_{\mathrm{bound}_a(b)}(M)}\ {}^{(15)} \tag{3.1.9}$$

is directly inspired by Equality 3.1.4.

Writing $a = [\underline{a}, \overline{a}]$ and $b = [\underline{b}, \overline{b}]$, the interval $\mathrm{bound}_a(b)$ is explicitly given by:

$$\mathrm{bound}_a(b) = \begin{cases} [\underline{a}, \underline{a}] & \text{if } \overline{b} < \underline{a} \\ [\max(\underline{a}, \underline{b}), \min(\overline{b}, \overline{a})] & \text{if } a \cap b \neq \emptyset \\ [\overline{a}, \overline{a}] & \text{if } \overline{a} < \underline{b}. \end{cases} \tag{3.1.10}$$

(See also Section 4.1.1 on page 43, Equation 4.1.6.)

We will sometimes use the notation $a \overrightarrow{\sqcup} b$ for $\mathrm{bound}_a(b)$, especially in indices. For instance, it is more convenient to write

$$\texttt{bound}_a\left(\texttt{bound}_b(M)\right) \to \texttt{bound}_{a \overrightarrow{\sqcup} b}(M)\,.$$

Remark 3.1.2. For any rational number q, rule (15) reduces term $\texttt{bound}_{[q-1,q]}\left(\texttt{bound}_{[q,q+1]}(M)\right)$ to term $\texttt{bound}_{[q,q]}(M)$ where the index of the `bound` constructor is a singleton. Accordingly, the set $\mathcal{R}_{\mathbb{Q}}$ contains all singletons $\{q\}$ where $q \in \mathbb{Q}$. This is why we allow p to be equal to q in the definition of $\mathcal{R}_{\mathbb{Q}}$, in Equation 3.1.8.

Hence, any rational number q can be defined directly as $\texttt{bound}_{[q,q]}(M)$, where M can be any term of type `real`.

Addition and multiplication by rational numbers

To operate on real numbers, addition and multiplication by rational numbers are put in as basic constructors, according to the typing judgement rules

$$\frac{\Gamma \vdash M : \texttt{real}}{\Gamma \vdash p + (M) : \texttt{real}} \text{ and } \frac{\Gamma \vdash M : \texttt{real}}{\Gamma \vdash p \times (M) : \texttt{real}}. \tag{3.1.11}$$

Constructors $p+$ and $p\times$ operate on approximations of real numbers via the `bound` constructors according to the reduction rules

$$\begin{array}{rl} & \overset{(17)}{\dfrac{}{p + \left(\mathtt{bound}_a(M)\right) \to \mathtt{bound}_{p+a}\left(p + (M)\right)}} \\ \text{and} & \overset{(19)}{\dfrac{}{p \times \left(\mathtt{bound}_a(M)\right) \to \mathtt{bound}_{p\times a}\left(p \times (M)\right)}}, \end{array} \tag{3.1.12}$$

which correspond to the equalities

$$p + \mathrm{bound}_a\,(x) = \mathrm{bound}_{p+a}\,(p + x) \text{ and} \tag{3.1.13}$$

$$p \times \mathrm{bound}_a\,(x) = \mathrm{bound}_{p\times a}\,(p \times x)\,, \tag{3.1.14}$$

satisfied by any real numbers p, x and any non-empty, closed bounded interval a.

Example: Reduction of a program for 0.

Here is the program for 0 that corresponds to Equation 3.1.2.

$$\mathrm{zero} = \mathtt{Y}_{\mathtt{real}}\left(\left(\lambda z\colon \mathtt{real}.\frac{1}{2} \times \left(\mathtt{bound}_{[-1,1]}(z)\right)\right)\right). \tag{3.1.15}$$

There is only one, infinite reduction path from this program. Figure 3.3 on the following page details the beginning of this path. Its uniqueness follows from the side condition $|M| = \bot$ appearing to the left of certain reduction rules in Table 3.2. The purpose of this condition is to restrict the applicability of rules (16), (18) and (20), so as to give priority to rules (15), (17) and (19). Condition $|M| = \bot$ is equivalent to M not being of the form $\mathtt{bound}_a(N)$, as will be explained in Section 3.1.3.

One could show that the unique path from program zero is of the form

$$\mathrm{zero} \to \ldots \to \mathtt{bound}_{[-\frac{1}{2},\frac{1}{2}]}\,\frac{1}{2}\,\mathrm{zero} \to \ldots \to \mathtt{bound}_{[-\frac{1}{2^n},\frac{1}{2^n}]}\,\frac{1}{2} \times \frac{1}{2} \times \ldots \times \frac{1}{2}\,\mathrm{zero} \to \ldots \tag{3.1.16}$$

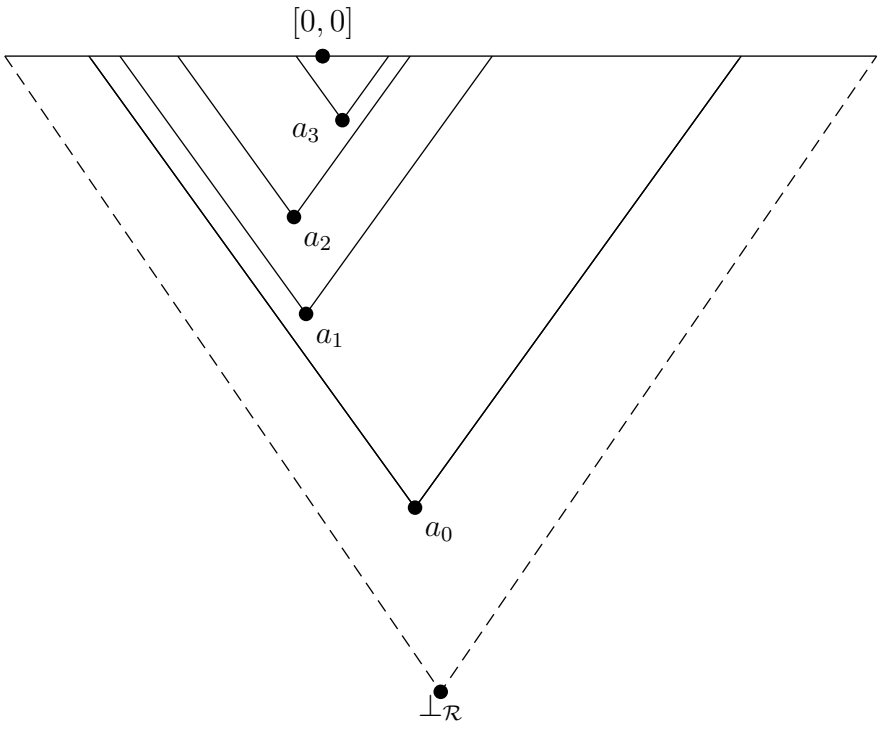
3.1.1 The interval domain

The reduction relation reduces a program to another program that possibly displays a better, finite approximation of its result. If the program to be reduced is of type `nat` or `bool` and total, it may eventually be reduced to a *value*, a program that cannot be reduced further and has a total meaning, such has $\underline{16}$ or `true`. The situation is different with programs for real numbers: the reduction relation brings forward possibly more precise estimates in the form of intervals a, at the head of programs such as $\mathtt{bound}_a(M)$, but the evaluation process never terminates. We say that a is the *immediate output* of

Table 3.3: Formal reduction of a program for 0.

$$
\begin{aligned}
&\text{zero}\\
&= \mathtt{Y}_{\mathtt{real}}\left(\left(\lambda z\colon \mathtt{real}.\frac{1}{2}\times\left(\mathtt{bound}_{[-1,1]}(z)\right)\right)\right)\\
&\to \left(\lambda z\colon \mathtt{real}.\frac{1}{2}\times\left(\mathtt{bound}_{[-1,1]}(z)\right)\right)(\text{zero})\\
&\to \frac{1}{2}\times\left(\mathtt{bound}_{[-1,1]}(\text{zero})\right)\\
&\to \mathtt{bound}_{\left[-\frac{1}{2},\frac{1}{2}\right]}\left(\frac{1}{2}\times(\text{zero})\right)\\
&\to^3 \mathtt{bound}_{\left[-\frac{1}{2},\frac{1}{2}\right]}\left(\frac{1}{2}\times\left(\mathtt{bound}_{\left[-\frac{1}{2},\frac{1}{2}\right]}\left(\frac{1}{2}\times(\text{zero})\right)\right)\right)\\
&\to \mathtt{bound}_{\left[-\frac{1}{2},\frac{1}{2}\right]}\left(\mathtt{bound}_{\left[-\frac{1}{4},\frac{1}{4}\right]}\left(\frac{1}{2}\times\left(\frac{1}{2}\times(\text{zero})\right)\right)\right)\\
&\to \mathtt{bound}_{\left[-\frac{1}{4},\frac{1}{4}\right]}\left(\frac{1}{2}\times\left(\frac{1}{2}\times(\text{zero})\right)\right)\\
&\to^3 \mathtt{bound}_{\left[-\frac{1}{4},\frac{1}{4}\right]}\left(\frac{1}{2}\times\left(\frac{1}{2}\times\left(\mathtt{bound}_{\left[-\frac{1}{2},\frac{1}{2}\right]}\left(\frac{1}{2}\times(\text{zero})\right)\right)\right)\right)\\
&\to^2 \mathtt{bound}_{\left[-\frac{1}{4},\frac{1}{4}\right]}\left(\mathtt{bound}_{\left[-\frac{1}{8},\frac{1}{8}\right]}\left(\frac{1}{2}\times\left(\frac{1}{2}\times\left(\frac{1}{2}\times(\text{zero})\right)\right)\right)\right)\\
&\to \mathtt{bound}_{\left[-\frac{1}{8},\frac{1}{8}\right]}\left(\frac{1}{2}\times\left(\frac{1}{2}\times\left(\frac{1}{2}\times(\text{zero})\right)\right)\right)\\
&\textit{etc.}
\end{aligned}
$$

Figure 3.2: A few approximations of 0 in the interval domain.



program $\mathtt{bound}_a(M)$. It is the information that can be read at once from the head of a program without further evaluation. The supremum of those approximates in the *interval domain* along a reduction path constitutes the *output* of the programs being reduced.

Definition 3.1.1. The *interval domain* $\mathcal{R}$ is the set of all non-empty, closed and bounded real intervals ordered by reverse inclusion, along with a least element $\bot_{\mathcal{R}}$:

$$\mathcal{R} = \{\bot_{\mathcal{R}}\} \cup \{[x, y] \mid x, y \in \mathbb{R} \text{ and } x \leq y\}. \tag{3.1.17}$$

The least element is defined by

$$\bot_{\mathcal{R}} = \mathbb{R} = (-\infty, +\infty).$$

The interval domain is a continuous Scott domain in which the supremum of a directed set is its intersection. The singletons $\{r\}$, where r ranges over $\mathbb{R}$, constitute its set of maximal elements, called *total elements*. As a Scott subspace of $\mathcal{R}$, the set of total elements is isomorphic to the real line, that is $\mathbb{R}$ with the standard Euclidian topology. The set $\mathcal{R}_{\mathbb{Q}} \cup \{\bot_{\mathcal{R}}\}$ is a countable base of the interval domain: any non-bottom element x of $\mathcal{R}$ is the supremum of those elements in $\mathcal{R}_{\mathbb{Q}}$ that are way below x, which are the intervals $[p, q]$ in $\mathcal{R}_{\mathbb{Q}}$ whose interior (p, q) contains the interval x. Figure 3.2 represents the interval domain with a few elements way-below $\{0\}$, which we view as approximates of 0.

3.1.2 Continuous domains

For the reader who isn't acquainted with continuous domains, we briefly recall some definitions and properties. See *e.g.* [1, 7] for further details.

The way-below relation

Let $(D, \sqsubseteq)$ be a dcpo. The *way below* relation $\ll$ on D is defined as follows. For all $d, e \in D$, $d \ll e$ if and only if, for all directed subsets $A \subseteq D$,

$$e \sqsubseteq \bigsqcup A \implies \exists a \in A, d \sqsubseteq a. \tag{3.1.18}$$

The way below relation is a transitive relation (not reflexive in general) and is also called the *order of approximation* on D. If $d \ll e$, we say that d *approximates* e.

The set of elements that are way-below (resp. way-above) an element d is written $\mathop{\downarrow\!\!\downarrow} d$ (resp. $\mathop{\uparrow\!\!\uparrow} d$).

Definition 3.1.2. A dcpo D is a *continuous domain* if, for all $d \in D$,

1. the subset $\mathop{\downarrow\!\!\downarrow} d$ of elements way below d is directed and

2. it holds that $d = \bigsqcup \mathop{\downarrow} d$.

The dcpo's $\mathbb{N}_\perp$ and $\mathbb{B}_\perp$ are trivial examples of continuous domains because all elements in those dcpo's are *finite*, that is way-below themselves. A more interesting example is the interval domain, whose only finite element is the least element. We recall the main results that are used in our proofs in the following theorem.

Theorem 3.1.1. *Let D be a continuous domain and $x, y \in D$.*

1. *$x \ll y$ if and only if for any directed subset A of D such that $y = \bigsqcup A$, there exists $a \in A$ such that $x \sqsubseteq a$.*
2. *The subsets $\Uparrow z$ form a basis for the Scott topology.*
3. *If $x \ll y$ then there exists $z \in D$ such that $x \ll z \ll y$ (Interpolation property).*

All dcpo's considered in this work are in fact *bounded-complete domains*, that is pointed, continuous dcpo's in which each bounded subset has an supremum.

3.1.3 Reduction paths and their output

Immediate outputs of programs

Definition 3.1.3. The *immediate output* of a program of type `real` is an element of the interval domain $\mathcal{R}$ and is defined by:

$$\begin{array}{ll} |\,\mathtt{bound}_a(N)\,| = a & \text{for any term } N\colon \mathtt{real} \text{ and any } a \in \mathcal{R}_{\mathbb{Q}}, \\ |M| = \perp_{\mathcal{R}} & \text{for any program not of the form } \mathtt{bound}_a(N). \end{array} \tag{3.1.19}$$

The immediate output of a program of type `bool` is an element of the lifted booleans $\mathbb{B}_\perp$, defined by:

$$\begin{array}{ll} |\,\mathtt{true}\,| = true, & \\ |\,\mathtt{false}\,| = false & \text{and} \\ |M| = \perp & \text{for all other programs of type } \mathtt{bool}. \end{array} \tag{3.1.20}$$

Similarly, the immediate output of a program of type `nat` is an element of the lifted integers $\mathbb{N}_\perp$, defined by:

$$\begin{array}{ll} |\underline{n}| = n & \text{for all natural numbers } n \text{ and} \\ |M| = \perp & \text{for all other programs of type } \mathtt{nat}. \end{array} \tag{3.1.21}$$

Remark 3.1.3. Notice that the relations $|M| = \perp$ and $|M| < q$ for $q \in \mathbb{Q}$ are decidable. For $|M| = \perp$, this amounts to checking whether the head of term M is `bound`, `true` or `false`. For a program M of type real, the condition $|M| < q$ is equivalent to M being of the form $\mathtt{bound}_{[p',q']}(N)$ with $q' < q$. Such tests can be performed during evaluation and are used to decide which reduction rules are applicable (*cf.* the side conditions for some rules in Table 3.2).

A program reduces to a program with equal or greater immediate output.

When reducing a program of the form $\texttt{bound}_a(N)$, only three exclusive cases arise:

1. The program N is itself of the form $\texttt{bound}_b(L)$. Then $\texttt{bound}_a(N)$ reduces to exactly one program, namely $\texttt{bound}_{a\overrightarrow{\sqcup}b}(L)$ and there is only one derivation of this fact:

$$\frac{}{\texttt{bound}_a\left(\texttt{bound}_b(L)\right) \to \texttt{bound}_{a\overrightarrow{\sqcup}b}(L)}.$$

2. N is not of the form $\texttt{bound}_b(L)$, *i.e.* $|N| = \bot$, and N reduces to some program. Then $\texttt{bound}_a(N)$ reduces to at least one program and any corresponding derivation is of the form

$$\frac{\dfrac{\ldots}{N \to N'}}{\texttt{bound}_a(N) \to \texttt{bound}_a(N')}.$$

3. N is not of the form $\texttt{bound}_b\,(L)$ and N cannot be reduced. Then $\texttt{bound}_a(N)$ cannot be reduced. However, it is easy to see that a *closed* term of type `real` can always be reduced.

In particular, noticing that $a \sqsubseteq a\overrightarrow{\sqcup}b$ for all $a, b \in \mathcal{R}_{\mathbb{Q}}$, we obtain that, for any $a \in \mathcal{R}_{\mathbb{Q}}$ and any terms $N, M'\colon \texttt{real}$, if $\texttt{bound}_a(N) \to M'$ then M' is of the form $\texttt{bound}_b(N')$ for some $b \in \mathcal{R}_{\mathbb{Q}}$ such that $a \sqsubseteq b$ and some term $N'\colon \texttt{real}$. Hence:

Lemma 3.1.2. *A term of ground type can only reduce to terms of equal or greater immediate outputs.*

$$M \to M' \implies |M| \sqsubseteq |M'| \tag{3.1.22}$$

This is of course true for terms of type `nat` and `bool`, since such terms can't be reduced if their output is not $\bot$.

Definition 3.1.4 (Reduction paths)**.** A *reduction path* is a sequence of terms, each reducing to the next one.

Output of a path

Lemma 3.1.2 states that if $M \to M'$ then $|M| \sqsubseteq |M'|$: A term of type real can only reduce to terms with equal or greater immediate outputs. Hence, given a reduction path (M_n), the corresponding sequence of immediate outputs (M_n) is a chain in the interval domain $\mathcal{R}$ and so possesses a supremum.

Definition 3.1.5. The supremum $\bigsqcup |M_n|$ of immediate outputs of programs along a reduction path (M_n) is called the *output of the path.*

For terms of type `bool` or `nat`, it is an element of the dcpo $\mathbb{B}_\perp$ or $\mathbb{N}_\perp$, respectively. At type `real`, the output of a path is an element of the interval domain $\mathcal{R}$. For example the output of the path

$$\text{zero} \to \ldots \to \texttt{bound}_{[-\frac{1}{2},\frac{1}{2}]}\left(\frac{1}{2}\,\text{zero}\right) \to \ldots \to \texttt{bound}_{[-\frac{1}{2^n},\frac{1}{2^n}]}\left(\frac{1}{2}\times\frac{1}{2}\times\ldots\times\frac{1}{2}\,\text{zero}\right) \to \ldots,$$

from program zero seen earlier, is $\{0\}$.

It is easy to see that there exists only one reduction path from any given term in the weak fragment. In this deterministic setting, the *operational meaning* $[M]$ of a program M can simply be defined as the output of the unique reduction path from it, that is as an element of $\mathcal{R}$, $\mathbb{N}_\perp$ or $\mathbb{B}_\perp$. A term $F\colon \texttt{real} \to \texttt{real}$ is then said to *compute* a given numerical function $f\colon \mathbb{R} \to \mathbb{R}$ if for all $M\colon \texttt{real}$ and all $r \in \mathbb{R}$,

$$[M] = \{r\} \implies [F(M)] = \{f(r)\}\,. \tag{3.1.23}$$

However, Farjudian showed in his thesis [10] that only a very restricted set of unary numerical functions is definable in the weak fragment, namely those functions whose graph has the same shape as the one of the function $bound_a$, represented in figure 3.1 on page 23: constant-increasing-constant or constant-decreasing-constant. In particular, the absolute value function is not definable in the weak fragment.

Redundant test: the `rtest` constructor.

We have already motivated the introduction of the `rtest` operator in the main introduction (page 1). Recall that we require that, for each real x,

1. rtest(x) must always evaluate to true or to false,
2. rtest(x) may evaluate to true iff $x < 1$, and
3. rtest(x) may evaluate to false iff $0 < x$.

We thus complete the description of typing judgements with the rule

$$\frac{\Gamma \vdash L\colon \texttt{real}}{\Gamma \vdash \texttt{rtest}(L)\colon \texttt{bool}}. \tag{3.1.24}$$

Reduction rules for rtest constructors are the only ones with overlapping premises, which renders the operational semantics non-deterministic, in the sense that one term may reduces to several ones:

$$(21)\frac{}{\texttt{rtest}(L) \to \texttt{true}}|L| < 1 \qquad (22)\frac{}{\texttt{rtest}(L) \to \texttt{false}}0 < |L| \tag{3.1.25}$$

$$(23)\frac{L \to L'}{\texttt{rtest}(L) \to \texttt{rtest}(L')}. \tag{3.1.26}$$

We only introduce one constructor $\texttt{rtest}_{0,1}$, simply written `rtest`, because the other ones are definable in the language, using the fact that $\mathrm{rtest}_{p,q}(x) = \mathrm{rtest}_{0,1}\left(\frac{x-p}{q-p}\right)$.

We have now fully described the language. Notice that all reduction rules have at most one premise, so any derivation that a term reduces to another is a finite sequence. Furthermore, we will prove that, for any two closed terms M and N such that $M \to N$, the derivation of $M \to N$ is unique (*cf.* Lemma 3.2.1 on the next page).

3.2 Operational meaning of programs

Due to the reduction rules for `rtest`, the reduction relation $\to$ is non-deterministic: a term may reduce to several terms. Recall that the rules for `rtest` are

$$(21)\frac{}{\texttt{rtest}(L) \to \texttt{true}}|L| < 1, \qquad (22)\frac{}{\texttt{rtest}(L) \to \texttt{false}}0 < |L| \text{ and} \tag{3.2.1}$$

$$(23)\frac{L \to L'}{\texttt{rtest}(L) \to \texttt{rtest}(L')}. \tag{3.2.2}$$

The purpose of rule (23) is to eventually enable the applicability of one or both rules (21) and (22). Only when none of (21) and (22) become applicable, should rule (23) be allowed to be applied infinitely often. This is made precise by the introduction of fair reduction paths. Even if one considers only fair paths, a program may still output inconsistent results. For example, the program rtest (half), where half is a program that computes 1/2, may reduce to both true and false. This behaviour leads us to define the *set of outputs of a program* in order to capture the necessary information that will allow us to define the operational meaning of a program (section 3.2.4).

3.2.1 Fair reduction paths

In order to avoid infinite computations arising from repeated applications of rule (23), one could give priority to the application of rules (21) and (22) over the application of rule (23), by stipulating that rule (23) should not be applied whenever one of the two other rules can be applied. But this would make `rtest` able to distinguish between programs with the same total denotation. For example, the term

$$\texttt{rtest}\left(\texttt{bound}_{\left[0,\frac{1}{2}\right]}(\text{half})\right)$$

would only reduce to `true` whereas the term

$$\texttt{rtest}\left(\texttt{bound}_{\left[\frac{1}{2},1\right]}(\text{half})\right)$$

would only reduce to `false`, although both terms

$$\mathtt{bound}_{\left[0,\frac{1}{2}\right]}\,(\mathrm{half})$$

and

$$\mathtt{bound}_{\left[\frac{1}{2},1\right]}\,(\mathrm{half})$$

compute the number $\frac{1}{2}$. We wish to remain as general as possible and not put artificial constraints on the operational behaviour of the language. It should be up to a particular implementation to make specific decisions to avoid undesirable paths. In order to rule out undesirable paths without imposing too much constraint, we stipulate that only *fair paths* should be allowed. Roughly, a fair path is one along which rules (21) and (22) are eventually applied when possible. To give a more precise definition, we will use the following lemma.

Lemma 3.2.1. *If a term M reduces to a term M' in one step then the derivation of $M \to M'$ is unique.*

Proof. There is no pair of distinct reduction rules in Table 3.2, or more precisely no pair of distinct *instances* of reduction rules, that share the same conclusion. So given M and M' such that $M \to M'$, there is exactly one rule whose conclusion is $M \to M'$. This entails that two derivations of $M \to M'$ possess, as finite sequences, the same set of finite suffixes, hence are equal. □

So a term may reduce to several other terms, but for each of these, the corresponding derivation is unique. For example, let consider the term

$$\mathtt{rtest}\left(\mathtt{bound}_{\left[\frac{1}{4},\frac{3}{4}\right]}(\Omega_{\mathtt{real}})\right)$$

where $\Omega_{\mathtt{real}} = \mathtt{Y}\,(\lambda x\colon \mathtt{real}.x)$. It reduces to the three terms

$$\mathtt{true},\ \mathtt{false} \text{ and } \mathtt{rtest}\left(\mathtt{bound}_{\left[\frac{1}{4},\frac{3}{4}\right]}\left((\lambda x\colon \mathtt{real}.x)(\Omega_{\mathtt{real}})\right)\right)$$

in one step, and the unique corresponding derivations are:

$$(21)\frac{}{\mathtt{rtest}\left(\mathtt{bound}_{\left[\frac{1}{4},\frac{3}{4}\right]}(\Omega_{\mathtt{real}})\right) \to \mathtt{true}}, \quad (22)\frac{}{\mathtt{rtest}\left(\mathtt{bound}_{\left[\frac{1}{4},\frac{3}{4}\right]}(\Omega_{\mathtt{real}})\right) \to \mathtt{false}}$$

and

$$(23)\frac{(1)\ \dfrac{}{\Omega_{\mathtt{real}} \to (\lambda x\colon \mathtt{real}.x)(\Omega_{\mathtt{real}})}}{\mathtt{rtest}\left(\mathtt{bound}_{\left[\frac{1}{4},\frac{3}{4}\right]}(\Omega_{\mathtt{real}})\right) \to \mathtt{rtest}\left(\mathtt{bound}_{\left[\frac{1}{4},\frac{3}{4}\right]}\left((\lambda x\colon \mathtt{real}.x)(\Omega_{\mathtt{real}})\right)\right)}.$$

We distinguish two kinds of derivations. A *weak derivation* is one in which rule (23) is applied where either rules (21) or (22) could be applied instead. Thus, a weak derivation is of the form

$$\dfrac{\dfrac{\dfrac{\vdots}{L \to L'}}{\texttt{rtest}(L) \to \texttt{rtest}(L')}\,(23)}{\dfrac{\vdots}{M \to M'}} \tag{3.2.3}$$

where the term L satisfies $|L| < 1$ or $0 < |L|$. Other derivations are called *strong derivations*.

Definition 3.2.1 (Fair reduction paths, computations)**.** A *fair path* is either a finite path or a path with infinitely many strong reduction steps. A *computation* is a maximal fair path. More precisely, a computation is either an infinite, fair reduction path or a finite, non-empty sequence $(M_n)_{0 \leq n \leq N}$ such that $M_n \to M_{n+1}$ for all integers n smaller than N and such that the last term M_N doesn't reduce to another term. To avoid cumbersome bookkeeping, we will speak of a computation (M_n) without mentioning the range of the index n, unless it becomes important to mention it.

Computations are the only paths that any implementation of the language should consider. What is important, is that there are strategies that allow to produce maximal fair paths—for example, any strategy that only performs strong reduction steps. The two following propositions are easy to prove.

Proposition 3.2.2. *Any finite path extends to a computation.*

Proposition 3.2.3. *Any computation from a program of type* `real` *is infinite.*

3.2.2 The set of outputs of a program

Even if we only consider computations from a fixed program, these may have different outputs. We have already mentioned the program `rtest`(half), which reduces to both `true` and `false`.

Definition 3.2.2. We define the *outputs of a program* as the set of outputs of all *computations*, *i.e.* maximal fair paths, from this program:

$$\text{Outputs}\,(M) = \left\{ \bigsqcup_n |M_n| \;\middle|\; (M_n) \text{ is a computation from } M \right\}. \tag{3.2.4}$$

Since there is at least one fair path from a given term M, the set Outputs(M) is not empty. Here is an example of a program with an infinite set of outputs in $\mathbb{N}_\perp$.

$$\begin{array}{l}\phi(0)\\ \text{where}\\ \phi(n) = \text{if } \big(\,\text{rtest}\,(\text{half})\,\big) \text{ then } n \text{ else } \phi\,(1+n)\,.\end{array} \tag{3.2.5}$$

Its set of outputs is

$$\Big\{\perp, 0, 1, 2, 3, \ldots, 0, n+1, \ldots\Big\}. \tag{3.2.6}$$

In particular, the least element $\perp$ belongs to this set because there is a fair path from $\phi(0)$ along which rule

$$\overset{(21)}{\dfrac{}{\text{rtest}\,\big(\,(\text{half})\,\big) \to \texttt{true}}} \tag{3.2.7}$$

is never applied.

The set of outputs of a program is thus a subset of $\mathcal{R}$, $\mathbb{N}_\perp$ or $\mathbb{B}_\perp$, depending on the the type of this program. Let us emphasize this:

1. $\forall M : \texttt{real}$ Outputs(M) $\in \mathcal{P}(\mathcal{R})$, *i.e.* is a set of intervals, with possibly $\perp_{\mathcal{R}}$ in it.
2. $\forall M : \texttt{nat}$ Outputs(M) $\in \mathcal{P}(\mathbb{N}_\perp)$, *i.e.* is a set of natural numbers with possibly the least element $\perp$ in it.
3. $\forall M : \texttt{bool}$ Outputs(M) $\in \mathcal{P}(\mathbb{B}_\perp)$, *i.e.* is a set of boolean values with possibly the least element $\perp$ in it.

Here are some examples of programs of type `bool` with different sets of outputs:

1. Outputs($\perp_{\mathbb{B}}$) = $\{\perp_{\mathbb{B}}\}$,
2. Outputs(`true`) = $\{\text{true}\}$,
3. Outputs $\big(\,\texttt{rtest}(\text{half})\big) = \{\text{true}, \text{false}\}$,
4. Outputs $(\texttt{if rtest}\,(\text{half})\,\texttt{then}\,\perp_{\mathbb{B}}\,\texttt{else false}) = \{\perp_{\mathbb{B}}, \text{false}\}$.

In fact, for any non-empty subset of $\mathbb{B}_\perp$, there is a program whose outputs is exactly this set. This cannot be the case with $\mathcal{R}$, since there are uncountably many subsets of $\mathcal{R}$. Nevertheless, to any non-empty finite subset of $\mathcal{R}$ corresponds a program whose set of outputs is this subset and some programs have an infinite, countable set of outputs.

As a more interesting example, here is a program for the absolute value, inspired by the definition using *rtest* that we gave in the introduction (Equation 1.0.1 on page 3).

$$\begin{array}{l} \mathrm{abs}\colon \mathtt{real} \to \mathtt{real} \\ \mathrm{abs}\,(x) = \mathtt{if}\,\mathtt{rtest}_{-1,0}(x)\ \mathtt{then}\ \ -x \\ \qquad\qquad\qquad\qquad\ \ \mathtt{else}\ \ \mathtt{if}\,\mathtt{rtest}_{0,1}(x)\ \mathtt{then}\ \ \tfrac{1}{2}\,\mathtt{bound}_{[-1,1]}(\mathrm{abs}\,(2x)) \\ \qquad\qquad\qquad\qquad\qquad\qquad\ \ \mathtt{else}\ \ x, \end{array} \tag{3.2.8}$$

where

$\mathtt{rtest}_{0,1}(x) = \mathtt{rtest}(x)$ and
$\mathtt{rtest}_{-1,0} = \mathtt{rtest}(1 + x)$

We will use denotational methods in chapter 5 to show that the program $\mathrm{abs}\colon \mathtt{real} \to \mathtt{real}$ computes the numerical function $|\cdot|\colon \mathbb{R} \to \mathbb{R}$, in the sense that, for any program $M\colon \mathtt{real}$ that computes some real number r, the program $\mathrm{abs}\,(M)$ computes the absolute value of r (see Chapter 5). This can also be easily verified directly, from the definition of the reduction relation.

3.2.3 Power domains

To account for the multi-valuedness of programs, both the operational meaning $[M]$ and the denotation $[\![M]\!]$ of a program M of ground type will be defined in a power domain $\mathcal{P}D$ over the dcpo D in which the outputs of computations are defined ($\mathcal{R}$, $\mathbb{N}_\perp$ or $\mathbb{B}_\perp$). A power domain PD over a dcpo D is a set of subsets of D with a suitable order that makes PD itself a dcpo. Concretely, $[M]$ and $[\![M]\!]$ will be defined as subsets of D that contain $\mathrm{Outputs}\,(M)$.

There is a detailed account of power domains in the literature. They can be constructed as ideal completions of finite subsets of basis elements [14], as formal dcpo-algebras [1] or as topological representations [20, 1, 15]. We provide basic definitions and properties, without proofs, based on the latter.

Smyth power domains

Definition 3.2.3 (Smyth power domains)**.** Given a dcpo D, its *Smyth power domain* $\mathcal{P}^{\mathcal{S}}\,D$ is the set of non empty, Scott compact and upper subsets of D, ordered by reverse inclusion.

The Hasse diagram of the Smyth power domain over $\mathbb{B}_\perp$ is represented in Figure 3.3. It plays a prominent role in this thesis, as it will be used to provide a denotation for the constructor `rtest`. Since $\mathbb{B}_\perp$ is a finite dcpo, the elements of $\mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$ are the non-empty upper subsets of $\mathbb{B}_\perp$, ordered by reverse inclusion.

Here are the main properties of Smyth power domains that we will use.

1. If D is a continuous domain, so is $\mathcal{P}^{\mathcal{S}}\, D$. The supremum of a directed family in $\mathcal{P}^{\mathcal{S}}\, D$ is its intersection:

$$\bigsqcup A = \bigcap A. \tag{3.2.9}$$

2. For any two elements in $\mathcal{P}^{\mathcal{S}}\, D$, their union as subsets of D is their meet in $\mathcal{P}^{\mathcal{S}}\, D$:

$$\forall x, y \in \mathcal{P}^{\mathcal{S}}\, D \qquad x \sqcap y = x \cup y. \tag{3.2.10}$$

3. There is a natural topological embedding, for the Scott topologies,

$$\begin{array}{rcl} \eta_D : D & \longrightarrow & \mathcal{P}^{\mathcal{S}}\, D \\ d & \longmapsto & \uparrow d. \end{array} \tag{3.2.11}$$

   We will often write $\eta(d)$ rather than $\eta_D(d)$. Notice that a maximal element d in D is sent to the maximal element $\{d\}$ in $\mathcal{P}^{\mathcal{S}}\, D$. In particular, for any real number r, the singleton $\{r\}$, or $[r, r]$, which is a maximal element of the interval domain $\mathcal{R}$, is sent to $\eta_{\mathcal{R}}\big(\{r\}\big) = \{\{r\}\} = \{[r, r]\}$ in $\mathcal{P}^{\mathcal{S}}\, \mathcal{R}$.

4. Given a continuous function $f\colon D \to \mathcal{P}^{\mathcal{S}}\, E$ from a dcpo to a Smyth power domain, the function $f^*\colon\ \mathcal{P}^{\mathcal{S}}\, D \to \mathcal{P}^{\mathcal{S}}\, E$ defined by

$$f^*(X) = \bigcup_{x \in X} f(x) \tag{3.2.12}$$

   is the unique extension of f to $\mathcal{P}^{\mathcal{S}}\, D$ that preserves binary meets such that

$$\begin{array}{ccc} D & & \\ \downarrow{\scriptstyle \eta} & \searrow^{f} & \\ \mathcal{P}^{\mathcal{S}}\, D & \xrightarrow{f^*} & \mathcal{P}^{\mathcal{S}}\, E. \end{array} \tag{3.2.13}$$

5. In particular, given a continuous function $f\colon D \to E$ between dcpo's, we write $f^{\mathcal{S}}$ for $(f \circ \eta_D)^*$. It is the unique continuous function from $\mathcal{P}^{\mathcal{S}}\, D$ to $\mathcal{P}^{\mathcal{S}}\, E$ preserving binary meets such that

$$\begin{array}{ccc} D & \xrightarrow{f} & E \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ \mathcal{P}^{\mathcal{S}}\, D & \xrightarrow{f^{\mathcal{S}}} & \mathcal{P}^{\mathcal{S}}\, E, \end{array} \tag{3.2.14}$$

   and is defined by:

$$f^{\mathcal{S}}(X) = \bigcup_{x \in X} \eta\big(f(x)\big). \tag{3.2.15}$$

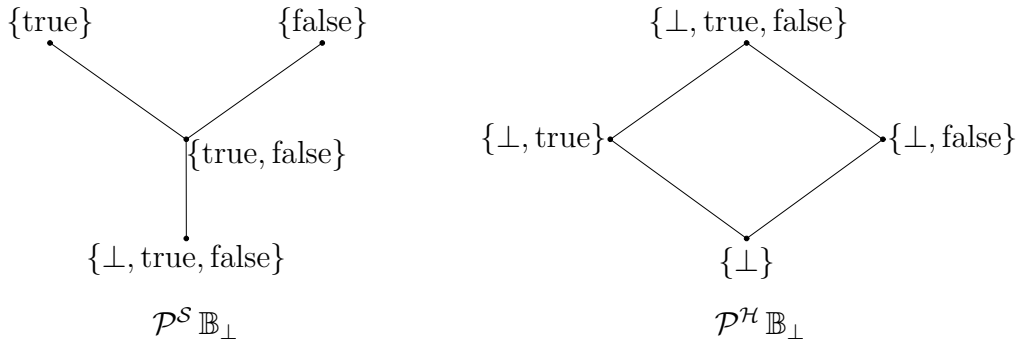


Figure 3.3: Smyth and Hoare power domains over the lifted booleans $\mathbb{B}_\perp$.

Hoare power domains

Definition 3.2.4 (Hoare power domains)**.** Given a dcpo D, its *Hoare power domain* $\mathcal{P}^{\mathcal{H}}\,D$ is defined as the set of non empty, Scott closed subsets of D, ordered by inclusion.

The Hasse diagram of the Hoare power domain over $\mathbb{B}_\perp$ is represented in Figure 3.3. As $\mathbb{B}_\perp$ is a finite dcpo, the elements of $\mathcal{P}^{\mathcal{H}}\,\mathbb{B}_\perp$ are the non-empty down subsets of $\mathbb{B}_\perp$, ordered by inclusion.

Here are some of the properties of Hoare power domains, corresponding to the one mentioned above for Smyth power domain:

1. If D is a continuous domain, so is $\mathcal{P}^{\mathcal{H}}\,D$. The supremum of a directed family in $\mathcal{P}^{\mathcal{H}}\,D$ is the Scott closure of its union:

$$\bigsqcup\nolimits^{\uparrow} A = \mathrm{cl}\left(\bigcup A\right). \tag{3.2.16}$$

2. For any two elements in $\mathcal{P}^{\mathcal{H}}\,D$, their intersection as subsets of D is their meet in $\mathcal{P}^{\mathcal{H}}\,D$:

$$\forall x, y \in \mathcal{P}^{\mathcal{S}}\,D \qquad x \sqcap y = x \cap y. \tag{3.2.17}$$

3. There is a natural topological embedding, for the Scott topologies,

$$\begin{array}{rcl} \eta : D & \longrightarrow & \mathcal{P}^{\mathcal{H}}\,D \\ d & \longmapsto & \downarrow d. \end{array} \tag{3.2.18}$$

   Any element in $\mathcal{P}^{\mathcal{H}}\,D$ contains the least element of D. This is why a denotational semantics based on Hoare power domains does not account for total correctness.

3.2.4 Operational meaning of a program

The operational meaning of a program of ground type γ is defined in the same dcpo $[\![\gamma]\!]$ as the one in which the denotation of the program will be defined (in Chapter 4). The interpretation $[\![\gamma]\!]$ of types γ is given by:

$$\begin{aligned} [\![\texttt{nat}]\!] &= \mathcal{P}^{\mathcal{S}}\,\mathbb{N}_\perp, \\ [\![\texttt{bool}]\!] &= \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp, \\ [\![\texttt{real}]\!] &= \mathcal{P}^{\mathcal{S}}\,\mathcal{R}. \end{aligned} \tag{3.2.19}$$

Definition 3.2.5. The *operational meaning* $[M]$ of a closed term M of ground type γ is the greatest element in the domain $[\![\gamma]\!]$ that contains the set of outputs of M:

$$[M] = \bigsqcup \{K \in [\![\gamma]\!] \mid K \supseteq \mathrm{Outputs}(M)\}\,. \tag{3.2.20}$$

One should bear in mind that the dcpo $[\![\gamma]\!]$ is a Smyth power domain, hence ordered by reverse inclusion, so the greatest element in the domain is the smallest for subset inclusion. Furthermore, the supremum of a directed family is its intersection and we could write: $[M] = \bigcap \{K \in [\![\gamma]\!] \mid K \supseteq \mathrm{Outputs}(M)\}$. Notice that Definition 3.2.5 relies on the fact that, for any closed term M of ground type γ, the set $\{K \in [\![\gamma]\!] \mid K \supseteq \mathrm{Outputs}\,(M)\}$ is directed. This follows from the fact that, in continuous Scott domains (bounded complete continuous dcpos), the intersection of finitely many Scott compact upper sets is Scott compact [1].

The operational meaning of a program of type `bool` is in $\mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$ and coincides with its set of outputs when its set of outputs is already compact and upper, *i.e.* is either $\{\mathrm{true}\}$, $\{\mathrm{false}\}$, $\{\mathrm{true}, \mathrm{false}\}$ or $\mathbb{B}_\perp$. In the other cases, its set of outputs is $\perp_{\mathbb{B}_\perp}$, $\{\perp_{\mathbb{B}_\perp}, \mathrm{true}\}$ or $\{\perp_{\mathbb{B}_\perp}, \mathrm{false}\}$ and is not an element of $\mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$ (and is therefore a strict subset of the operational meaning of the program). Similar remarks could be made for programs of type `nat`.

Hence it is not always possible to recover the set of outputs of a program from its operational meaning. However, for any program M of ground type, if the operational meaning of M is a finite set of total elements of $\mathbb{N}_\perp$, $\mathbb{B}_\perp$ or $\mathcal{R}$, then we know that $\mathrm{Outputs}\,(M) = [M]$. In particular, if a program M computes a total element d of $\mathbb{N}_\perp$, $\mathbb{B}_\perp$ or $\mathcal{R}$, in the sense that all computations from M output d, then $[M] = \eta\,(d)$. The main motivation of our work is to define a denotational semantics such that $[\![M]\!]$ coincides with $[M]$ and hence with $\mathrm{Outputs}\,(M)$, at least for enough programs M for the denotational semantics to be useful. This will be the subject of Chapters 4, 5 and 6.

Chapter 4

An approximate semantics for total correctness

Marcial-Romero showed in his thesis [16] that it is possible to have an expressive programming language for exact real-number computation with a sequential operational semantics. However, he left as an open question whether it is possible to find a denotational semantics which allows to prove total correctness without the need to resort to operational methods. The main result of the present work is that this is possible. We describe here a semantics for the language and prove the following result (Theorem 4.3.1, Section 4.3 on page 51).

For any closed term M of ground type, $[\![M]\!] \sqsubseteq [M]$.

This result will be used in chapter 5 to prove that a few common functions (absolute value, addition, division, limits, power series, ...) are definable in the language. In chapter 6, we will even establish that our semantics, although below the operational semantics, is "close enough" to account for the definability of all computable, total first-order numerical functions. Hence the expression "approximate semantics".

In the first section, we present the semantics of the fragment without rtest in Smyth power domains. However, the key point in defining the semantics is to provide a denotation for the non-deterministic `rtest` construct (Section 4.2).

The remainder of the chapter is then dedicated to the proof of Theorem 4.3.1.

4.1 Approximate semantics in Smyth power domains

The interpretation of types as Smyth power domains is given in Table 4.1. Smyth power domains were introduced in Chapter 3, Section 3.2.3 on page 36 because they were needed to define the operational meaning of programs. The interpretations of ground types were already given in Chapter 3, Section 3.2.4 on page 39, as the domains in which the operational meanings of programs are defined.

Table 4.1: Interpretation of types.

$[\![\texttt{nat}]\!] = D_{\texttt{nat}} = \mathcal{P}^{\mathcal{S}}\,\mathbb{N}_{\perp}$,
$[\![\texttt{bool}]\!] = D_{\texttt{bool}} = \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_{\perp}$,
$[\![\texttt{real}]\!] = D_{\texttt{real}} = \mathcal{P}^{\mathcal{S}}\,\mathcal{R}$,
$[\![\sigma \to \tau]\!] = D_{\sigma\to\tau} = \big[\,[\![\sigma]\!] \to [\![\tau]\!]\,\big]$, the dcpo of Scott continuous functions from $[\![\sigma]\!]$ to $[\![\tau]\!]$.

The denotation of a typing judgement $x_1\colon \sigma_1, \ldots, x_k\colon \sigma_k \vdash M\colon \sigma$ is a continuous function from $[\![\sigma_1]\!] \times \ldots \times [\![\sigma_k]\!]$ to $[\![\sigma]\!]$ in the category of bounded-complete continuous domains. Table 4.2 on the following page gives the recursive definition of the denotations of typing judgements. The fact that the denotations of terms are continuous comes from classical results of domain theory (see *e.g.* [26]) and the fact that the denotations of basic constructors other than `rtest` are given as functions known to be continuous. The denotation of `rtest` is also continuous, as will be shown in Section 4.2.

The denotation of a basic constructor F where F is `succ`, `pred`, $(0 =)$, `true`, `false`, $\texttt{bound}_a$, $p+$, or $p\times$ is of the form

$$[\![\Gamma \vdash F\,(M) : \sigma]\!]\,(\,\overrightarrow{d}\,) = \left(\mathcal{P}^{\mathcal{S}}\,f\right)\left([\![M]\!]\,(\,\overrightarrow{d}\,)\right). \tag{4.1.1}$$

We follow the convention of [26] by using the notation $(\,\overrightarrow{d}\,)$ for $(d_0, \ldots, d_k)$ where $d_i \in [\![\sigma_i]\!]$ and $\Gamma = x_0\colon \sigma, \ldots, x_k\colon \sigma_k$.

The element $\mathcal{P}^{\mathcal{S}}(f)$ is the image of f through the endofunctor $\mathcal{P}^{\mathcal{S}}$ on the category of continuous domains. The function f belongs the hierarchy of domains based on $\mathbb{N}_{\perp}$, $\mathbb{B}_{\perp}$ and the interval domain $\mathcal{R}$ and is the greatest extension of the corresponding function between sets $\mathbb{N}$, $\mathbb{B}$ or $\mathbb{R}$. Before defining the denotation of rtest, we give a few more details on the denotation of basic operators.

Table 4.2: Denotation of typing judgements.

$$[\![x_1 \colon \sigma_1, \ldots, x_k \colon \sigma_k \vdash x_i \colon \sigma_i]\!]\,(\overrightarrow{d}) = d_i$$

$$[\![\Gamma \vdash (\lambda x \colon \sigma . M) \colon \sigma \to \tau]\!]\,(\overrightarrow{d})(e) = [\![\Gamma, x \colon \sigma \vdash M \colon \tau]\!]\,(\overrightarrow{d}, e)$$

$$[\![\Gamma \vdash \mathtt{Y}_\sigma(M) \colon \sigma]\!]\,(\overrightarrow{d}) = \mu_{D_\sigma}\left([\![\Gamma \vdash M \colon \sigma \to \sigma]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash M(N) \colon \tau]\!]\,(\overrightarrow{d}) = [\![\Gamma \vdash M \colon \sigma \to \tau]\!]\,(\overrightarrow{d})\Big([\![\Gamma \vdash N \colon \sigma]\!]\,(\overrightarrow{d})\Big)$$

$$[\![\Gamma \vdash (0 =)(M) \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \big(\mathcal{P}^{\mathcal{S}}(0 =)\big)\left([\![\Gamma \vdash M \colon \mathtt{nat}]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash \mathtt{succ}(M) \colon \mathtt{nat}]\!]\,(\overrightarrow{d}) = \big(\mathcal{P}^{\mathcal{S}}(\mathrm{succ})\big)\left([\![\Gamma \vdash M \colon \mathtt{nat}]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash \mathtt{pred}(M) \colon \mathtt{nat}]\!]\,(\overrightarrow{d}) = \big(\mathcal{P}^{\mathcal{S}}(\mathrm{pred})\big)\left([\![\Gamma \vdash M \colon \mathtt{nat}]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash \mathtt{true} \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\mathrm{true}\}$$

$$[\![\Gamma \vdash \mathtt{false} \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\mathrm{false}\}$$

$$[\![\Gamma \vdash \underline{n} \colon \mathtt{nat}]\!]\,(\overrightarrow{d}) = \{n\}$$

$$[\![\Gamma \vdash \mathtt{rtest}(L) \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \mathrm{ttest}^*\left([\![\Gamma \vdash L \colon \mathtt{real}]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash \mathtt{if}_\gamma(B, M, N) \colon \gamma]\!]\,(\overrightarrow{d})$$
$$= \begin{cases} [\![\Gamma \vdash M \colon \gamma]\!]\,(\overrightarrow{d}) & \text{if } [\![\Gamma \vdash B \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\mathrm{true}\} \\ [\![\Gamma \vdash N \colon \gamma]\!]\,(\overrightarrow{d}) & \text{if } [\![\Gamma \vdash B \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\mathrm{false}\} \\ \left([\![\Gamma \vdash M \colon \gamma]\!]\,(\overrightarrow{d})\right) \cup \left([\![\Gamma \vdash N \colon \gamma]\!]\,(\overrightarrow{d})\right) & \text{if } [\![\Gamma \vdash B \colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\mathrm{true}, \mathrm{false}\} \\ \bot_{D_\gamma} & \text{otherwise} \end{cases}$$

$$[\![\Gamma \vdash \mathtt{bound}_a(M) \colon \mathtt{real}]\!]\,(\overrightarrow{d}) = \big(\mathcal{P}^{\mathcal{S}}(\mathrm{bound}_a)\big)\left([\![\Gamma \vdash M \colon \mathtt{real}]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash p + (M) \colon \mathtt{real}]\!]\,(\overrightarrow{d}) = \big(\mathcal{P}^{\mathcal{S}}(p+)\big)\left([\![\Gamma \vdash M \colon \mathtt{real}]\!]\,(\overrightarrow{d})\right)$$

$$[\![\Gamma \vdash p \times (M) \colon \mathtt{real}]\!]\,(\overrightarrow{d}) = \big(\mathcal{P}^{\mathcal{S}}(p\times)\big)\left([\![\Gamma \vdash M \colon \mathtt{real}]\!]\,(\overrightarrow{d})\right)$$

4.1.1 Denotation of $\mathtt{bound}_a$, $p+$ and $p\times$.

It is easy to see that any continuous function $f\colon \mathbb{R} \to \mathbb{R}$ has a greatest continuous extension $\hat{f}\colon \mathcal{R} \to \mathcal{R}$, that is the greatest element in $[\mathcal{R} \to \mathcal{R}]$ such that the following diagram commutes:

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{\eta_{\mathbb{R}}} & \mathcal{R} \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \hat{f}} \\ \mathbb{R} & \xrightarrow{\eta_{\mathbb{R}}} & \mathcal{R} \end{array} \tag{4.1.2}$$

where

$$\begin{aligned} &\eta_{\mathbb{R}}\colon \mathbb{R} \to \mathcal{R} \\ &\eta_{\mathbb{R}}\,(r) = [r, r]. \end{aligned}$$

The function $\hat{f}$ can be explicitly expressed by

$$\hat{f}\,(x) = \{f\,(r) \mid r \in x\} = f[x]$$

for $x \neq \bot$, and $\hat{f}(\bot)$ is the smallest interval in $\mathcal{R}$ containing $f[\bot] = f[\mathbb{R}]$. If f is monotone, then, for $x \neq \bot$, we have:

- $\hat{f}([\underline{x}, \overline{x}]) = [f(\underline{x}), f(\overline{x})]$ if f is increasing,
- $\hat{f}([\underline{x}, \overline{x}]) = [f(\overline{x}), f(\underline{x})]$ if f is decreasing.

Because $\hat{f}\colon \mathcal{R} \to \mathcal{R}$ is Scott continuous, it sends compact subsets to compact subsets. It is not hard to see that it also sends non-empty upper sets to non-empty upper sets.

Lemma 4.1.1. *Let $f\colon \mathbb{R} \to \mathbb{R}$ be a continuous function and $\hat{f}\colon \mathcal{R} \to \mathcal{R}$ be its greatest continuous extension. Let X be a non-empty upper subset of $\mathcal{R}$. Then the direct image $\hat{f}[X]$ of X via $\hat{f}$ is a non-empty upper subset of $\mathcal{R}$.*

In particular, $\mathcal{P}^{\mathcal{S}}(\hat{f})(X) = \hat{f}[X]$.

Proof. Let f, $\hat{f}$ and X be as in the lemma. Let $y \in \hat{f}[X]$ and $z \in \mathcal{R}$ such that $y \sqsubseteq z$. We have to show that $z \in \hat{f}[X]$. Let us write $y = [\underline{y}, \overline{y}]$ and $z = [\underline{z}, \overline{z}]$. Since $y \sqsubseteq z$, we have that

$$\underline{y} \leq \underline{z} \leq \overline{z} \leq \overline{y}.$$

Because $y \in \hat{f}[X]$, there exists $x \in \mathcal{R}$ such that $\hat{f}(x) = y$. For simplicity, we only consider the case $x \neq \bot$. By definition of $\hat{f}$ and the intermediate value theorem, there are real

numbers $r_0, r_1 \in x$ such that $\hat{f}(x) = [f(r_0), f(r_1)] = [\underline{y}, \overline{y}]$. Let $r_0' = \min(r_0, r_1)$ and $r_1' = \max(r_0, r_1)$. Again by the intermediate value theorem, there exist $s_0, s_1 \in [r_0', r_1']$ such that $f(s_0) = \underline{z}$ and $f(s_1) = \overline{z}$. Hence, defining $u = [s_0, s_1]$, we have that $\hat{f}(u) = z$ and $x \sqsubseteq u$. Since X is upper, we have $u \in X$ and, therefore, $z \in \hat{f}[X]$. □

We find it useful to have the following diagram in mind:

$$\begin{array}{ccccc} \mathbb{R} & \xrightarrow{\eta_{\mathbb{R}}} & \mathcal{R} & \xrightarrow{\eta_{\mathcal{R}}} & \mathcal{P}^{\mathcal{S}}\,\mathcal{R} \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle \hat{f},\ x \mapsto f[x]} & & \downarrow{\scriptstyle \mathcal{P}^{\mathcal{S}}\,\hat{f},\ X \mapsto \hat{f}[X]} \\ \mathbb{R} & \xrightarrow{\eta_{\mathbb{R}}} & \mathcal{R} & \xrightarrow{\eta_{\mathcal{R}}} & \mathcal{P}^{\mathcal{S}}\,\mathcal{R} \end{array} \tag{4.1.3}$$

where

$$\begin{aligned} &\eta_{\mathcal{R}} \colon \mathcal{R} \to \mathcal{P}^{\mathcal{S}}\,\mathcal{R} \\ &\eta_{\mathcal{R}}\,(x) = \uparrow \{x\}\,. \end{aligned} \tag{4.1.4}$$

We will often just write η for $\eta_{\mathbb{R}}$, $\eta_{\mathcal{R}}$ and $\eta_{\mathcal{R}} \circ \eta_{\mathbb{R}}$ as which embedding is meant should always be clear from the context.

Because the functions $\mathrm{bound}_a \colon \mathbb{R} \to \mathbb{R}$ is continuous, for all $a \in \mathcal{R}_{\mathbb{Q}}$, Lemma 4.1.1 applies to this function. In particular, for each $a \in \mathcal{R}_{\mathbb{Q}}$, the function $\mathrm{bound}_a \colon \mathbb{R} \to \mathbb{R}$ defined in Section 3.1 has a greatest continuous extension, which is simply given by

$$\begin{aligned} &\widehat{\mathrm{bound}_a} \colon \mathcal{R} \to \mathcal{R} \\ &\widehat{\mathrm{bound}_a}\,(x) = \mathrm{bound}_a[x]. \end{aligned} \tag{4.1.5}$$

The function $\widehat{\mathrm{bound}_a} : \mathcal{R} \to \mathcal{R}$ can be seen as a total, continuous extension of the partial function $\lambda x.a \sqcup x$, which is only defined when $a \uparrow x$ (*i.e.* when $a \cap x \neq \emptyset$). More precisely, writing $a = [\underline{a}, \overline{a}]$ and $x = [\underline{x}, \overline{x}]$, we have the following:

$$\widehat{\mathrm{bound}_a}(x) = \begin{cases} \eta_{\mathbb{R}}(\underline{a}) & \text{if } \overline{x} < \underline{a} \\ a \sqcup x & \text{if } a \uparrow x \\ \eta_{\mathbb{R}}(\overline{a}) & \text{if } \overline{a} < \underline{x}. \end{cases} \tag{4.1.6}$$

Notice that

$$\widehat{\mathrm{bound}_a}\,(\bot) = a \tag{4.1.7}$$

and hence that

$$\mathcal{P}^{\mathcal{S}}\,\widehat{\mathrm{bound}}_a\,(\bot) = \uparrow a = \eta_{\mathcal{R}}(a). \tag{4.1.8}$$

Similarly, for each $p \in \mathbb{Q}$, the functions $(p+)\colon \mathbb{R} \to \mathbb{R}, r \mapsto p + r$ and $(p\times)\colon \mathbb{R} \to \mathbb{R}, r \mapsto p \times r$ have the following greatest continuous extensions:

$$\widehat{p+}\colon \mathcal{R} \to \mathcal{R} \tag{4.1.9}$$
$$\widehat{p+}\,(x) = \begin{cases} [p + \underline{x}, p + \overline{x}] & \text{if } x \neq \bot_{\mathcal{R}} \\ \bot_{\mathcal{R}} & \text{otherwise} \end{cases}$$

and

$$\widehat{p\times}\colon \mathcal{R} \to \mathcal{R} \tag{4.1.10}$$
$$\widehat{p\times}\,(x) = \begin{cases} \{p \times r \mid r \in x\} & \text{if } x \neq \bot_{\mathcal{R}} \\ \bot_{\mathcal{R}} & \text{otherwise.} \end{cases}$$

The functions $\mathcal{P}^{\mathcal{S}}\,\widehat{\mathrm{bound}}_a$, $\mathcal{P}^{\mathcal{S}}\,\widehat{p+}$ and $\mathcal{P}^{\mathcal{S}}\,\widehat{p\times}$ are used to to define the denotations of terms of the form $\mathtt{bound}_a(M)$, $p + (M)$ and $p \times (M)$ (see Table 4.2).

Notation

We overload our notation by writing bound_a, $p+$ and $p\times$ instead of $\widehat{\mathrm{bound}}_a$, $\widehat{p+}$ and $\widehat{p\times}$, as in Table 4.2.

4.1.2 Denotation of `succ`, `pred` and $(0 =)$

The situation is simpler for constructors `succ`, `pred` and $(0 =)$. For example we have the following:

$$\begin{array}{ccccc} \mathbb{N} & \xrightarrow{\eta_{\mathbb{N}}} & \mathbb{N}_\bot & \xrightarrow{\eta_{\mathbb{N}_\bot}} & \mathcal{P}^{\mathcal{S}}\,\mathbb{N}_\bot \\ \downarrow{\scriptstyle \mathrm{succ}} & & \downarrow{\scriptstyle \mathrm{succ}'} & & \downarrow{\scriptstyle \mathcal{P}^{\mathcal{S}}\,\mathrm{succ}'} \\ \mathbb{N} & \xrightarrow{\eta_{\mathbb{N}}} & \mathbb{N}_\bot & \xrightarrow{\eta_{\mathbb{N}_\bot}} & \mathcal{P}^{\mathcal{S}}\,\mathbb{N}_\bot, \end{array} \tag{4.1.11}$$

where

$$\eta_{\mathbb{N}} \colon \mathbb{N} \to \mathbb{N}_\perp \qquad \eta_{\mathbb{N}_\perp} \colon \mathbb{N}_\perp \to \mathcal{P}^{\mathcal{S}}\, \mathbb{N}_\perp$$
$$\eta_{\mathbb{N}}\,(n) = n, \qquad \eta_{\mathbb{N}_\perp}\,(n) = \uparrow \{n\} = \begin{cases} \{n\} & \text{if } n \neq \perp \\ \mathbb{N}_\perp & \text{otherwise,} \end{cases}$$

$$\mathrm{succ} \colon \mathbb{N} \to \mathbb{N} \qquad \mathrm{succ}' \colon \mathbb{N}_\perp \to \mathbb{N}_\perp$$
$$\mathrm{succ}(n) = n + 1, \qquad \mathrm{succ}'(n) = \begin{cases} n+1 & \text{if } n \neq \perp \\ \perp & \text{otherwise,} \end{cases}$$

and again, we will write succ rather than succ′. The denotations of the constructors `pred` and $(0 =)$ are based on the following functions:

$$\mathrm{pred} \colon \mathbb{N} \to \mathbb{N}$$
$$\mathrm{pred}(n) = \max\,(0, n-1) \text{ and}$$

$$0 = \colon \mathbb{N} \to \mathbb{B}$$
$$0 =(n) = \begin{cases} \mathrm{true} & \text{if } n = 0 \\ \mathrm{false} & \text{otherwise.} \end{cases}$$

4.1.3 Denotation of $\mathtt{if}_\gamma$

For each ground type γ, the function

$$\mathrm{if}_\gamma \colon \mathbb{B}_\perp \times D_\gamma \times D_\gamma \to D_\gamma \tag{4.1.12}$$
$$\mathrm{if}_\gamma(b, x, y) = \begin{cases} x & \text{if } b = \mathrm{true} \\ y & \text{if } b = \mathrm{false} \\ \perp_{D_\gamma} & \text{otherwise} \end{cases}$$

is continuous and so is

$$\mathrm{if}^*_\gamma \colon D_{\mathtt{bool}} \times D_\gamma \times D_\gamma \to D_\gamma \tag{4.1.13}$$
$$\mathrm{if}^*_\gamma(B, x, y) = \bigcup_{b \in B} \mathrm{if}_\gamma\,(b, x, y)\,,$$

which can be expressed as

$$\mathrm{if}^*_\gamma(B, x, y) = \begin{cases} x & \text{if } B = \{\mathrm{true}\} \\ y & \text{if } B = \{\mathrm{false}\} \\ x \cup y & \text{if } B = \{\mathrm{true}, \mathrm{false}\} \\ \perp_{D_\gamma} & \text{otherwise.} \end{cases}$$

This is the expression we use in Table 4.2 to give the denotation of judgements of the form $\Gamma \vdash \mathtt{if}_\gamma(B, M, N)\colon \gamma$. Notice that the asterisk in if^*_γ is an abuse of notation and doesn't have quite the same meaning as in ttest*, defined later in Section 4.2, because $\mathrm{if}^*_\gamma\colon \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp \times D_\gamma \times D_\gamma \to D_\gamma$ is not the extension of $\mathrm{if}_\gamma\colon \mathbb{B}_\perp \times D_\gamma \times D_\gamma \to D_\gamma$. However, if one fixes (x, y) in $D_\gamma \times D_\gamma$, then $\mathrm{if}^*_\gamma(-, x, y)\colon \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp \to D_\gamma$ is the extension of $\mathrm{if}_\gamma(-, x, y)\colon \mathbb{B}_\perp \to D_\gamma$. Also notice that if^*_γ could have been equivalently defined as arising from the Smyth power domain monad, which is strong and commutative (See [18]).

4.2 Denotation of `rtest`.

Theorem 4.2.1. *The function* $\mathrm{ttest}\colon \mathcal{R} \to \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$ *defined as follows is continuous.*

$$\mathrm{ttest}(l) = \begin{cases} \{\mathrm{true}\} & \text{if } \overline{l} < 0 \\ \{\mathrm{false}\} & \text{if } 1 < \underline{l} \\ \{\mathrm{true}, \mathrm{false}\} & \text{if } \overline{l} \not< 0, 1 \not< \underline{l} \text{ and } l \not\sqsubseteq [0,1] \\ \{\perp_{\mathbb{B}_\perp}\} & \text{if } l \sqsubseteq [0,1] \end{cases} \tag{4.2.1}$$

Proof. Notice that the sets of elements $l \in \mathcal{R}$ satisfying to a condition of definition 4.2.1, namely the four sets

$$\begin{aligned} &\left\{l \in \mathcal{R} \mid \overline{l} < 0\right\}, \\ &\left\{l \in \mathcal{R} \mid 1 < \underline{l}\right\}, \\ &\left\{l \in \mathcal{R} \mid l \not< 0, 1 \not< \underline{l} \text{ and } l \not\sqsubseteq [0,1]\right\} \\ \text{and } &\left\{l \in \mathcal{R} \mid l \sqsubseteq [0,1]\right\}, \end{aligned}$$

form a partition of $\mathcal{R}$ and so equality 4.2.1 does define a function. To see that this function, ttest, is continuous, we show that the inverse image of any open set via ttest is an open set. There are only three non trivial open sets in $\mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$: the upper sets $\uparrow\{\mathrm{true}\}$, $\uparrow\{\mathrm{false}\}$ and $\uparrow\{\mathrm{true}, \mathrm{false}\}$. Writing ttest^{-1} for the function that maps a subset of $\mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$ to its inverse image through ttest, *i.e.* to a subset of $\mathcal{R}$, one has:

$$\begin{aligned} \mathrm{ttest}^{-1}(\uparrow\{\mathrm{true}\}) &= \mathrm{ttest}^{-1}\big(\{\{\mathrm{true}\}\}\big) && (4.2.2)\\ &= \{l \in \mathcal{R} \mid \mathrm{ttest}(l) = \{\mathrm{true}\}\} && (4.2.3)\\ &= \left\{l \in \mathcal{R} \mid \overline{l} < 0\right\} && (4.2.4)\\ &= \{l \in \mathcal{R} \mid \exists r \in \mathbb{R} \quad r < 0 \text{ and } l \subset (r, 0)\} && (4.2.5)\\ &= \{l \in \mathcal{R} \mid \exists r \in \mathbb{R} \quad r < 0 \text{ and } l \gg [r, 0]\} && (4.2.6)\\ &= \bigcup_{r \in \mathbb{R}, r < 0} \Uparrow [r, 0]. && (4.2.7) \end{aligned}$$

The set $\Uparrow x$ of elements way above an element x in any continuous domain is open. Hence, being a union of open sets as shown by 4.2.7, the set $\mathrm{ttest}^{-1}(\uparrow\{\mathrm{true}\})$ is open. Similarly, the set $\mathrm{ttest}^{-1}(\uparrow\{\mathrm{false}\})$ is open because:

$$\begin{aligned}
\mathrm{ttest}^{-1}(\uparrow\{\mathrm{false}\}) &= \mathrm{ttest}^{-1}\big(\{\{\mathrm{false}\}\}\big) && (4.2.8)\\
&= \{l \in \mathcal{R} \mid \mathrm{ttest}(l) = \{\mathrm{false}\}\} && (4.2.9)\\
&= \{l \in \mathcal{R} \mid 1 < \underline{l}\} && (4.2.10)\\
&= \bigcup_{r\in\mathbb{R},1<r} \Uparrow[1,r]. && (4.2.11)
\end{aligned}$$

Finally, the set $\mathrm{ttest}^{-1}(\uparrow\{\mathrm{true},\mathrm{false}\})$ is open as well:

$$\begin{aligned}
\mathrm{ttest}^{-1}(\uparrow\{\mathrm{true},\mathrm{false}\}) &= \mathrm{ttest}^{-1}(\{\{\mathrm{true},\mathrm{false}\},\{\mathrm{true}\},\{\mathrm{false}\}\}) && (4.2.12)\\
&= \{l \in \mathcal{R} \mid \mathrm{ttest}(l) \neq \{\bot_{\mathbb{B}_\bot}\}\} && (4.2.13)\\
&= \{l \in \mathcal{R} \mid l \not\sqsubseteq [0,1]\} && (4.2.14)\\
&= \mathcal{R}\backslash \downarrow [0,1]. && (4.2.15)
\end{aligned}$$

The set $\downarrow x$ of elements below or equal to x is Scott closed in any dcpo, so the subset $\mathrm{ttest}^{-1}(\uparrow\{\mathrm{true},\mathrm{false}\})$, being the complement of a closed subset, is Scott open in $\mathcal{R}$. □

The denotation of `rtest` is the continuous function $\mathrm{ttest}^*\colon \mathcal{P}^{\mathcal{S}}\,\mathcal{R} \to \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\bot$ that maps K to $\bigcup\{\uparrow \mathrm{ttest}(k) \mid k \in K\}$.

4.2.1 Comparison of ttest and rtest

Figure 4.1 represents the values taken by the function ttest on the interval domain. The black wedges indicate to which area the boundary lines belong. In particular

$$\mathrm{ttest}\,([0,0]) = \mathrm{ttest}\,([1,1]) = \{\mathrm{true},\mathrm{false}\}\,. \tag{4.2.16}$$

Similarly, Figure 4.2 represents the values taken by `rtest` when seen as a function rtest from $\mathcal{R}$ to $\mathcal{P}(\mathbb{B}_\bot)$, the power set of $\mathbb{B}_\bot$, in the following way: given a program $M\colon$ `real` such that $[\![M]\!] = \eta\,(x) = \uparrow\{x\}$ for some interval $x \in \mathcal{R}$, Figure 4.2 indicates the set of outputs of the program `rtest` (M). On the areas that are of different colours from one figure to the other, we have that $\mathrm{ttest}(x)$ strictly contains $\mathrm{rtest}(x)$. In particular $\mathrm{rtest}\,([0,0]) = \{\mathrm{true}\}$ and $\mathrm{rtest}([1,1]) = \{\mathrm{false}\}$. It follows that for some programs $M\colon$ `real`, even some total ones, one has

$$[\![\texttt{rtest}\,(M)]\!] \sqsubseteq [\texttt{rtest}\,(M)] \tag{4.2.17}$$

and

$$[\![\texttt{rtest}\,(M)]\!] \neq [\texttt{rtest}\,(M)]\,. \tag{4.2.18}$$

Proving inequality 4.2.17 holds in general is the subject of the next section.

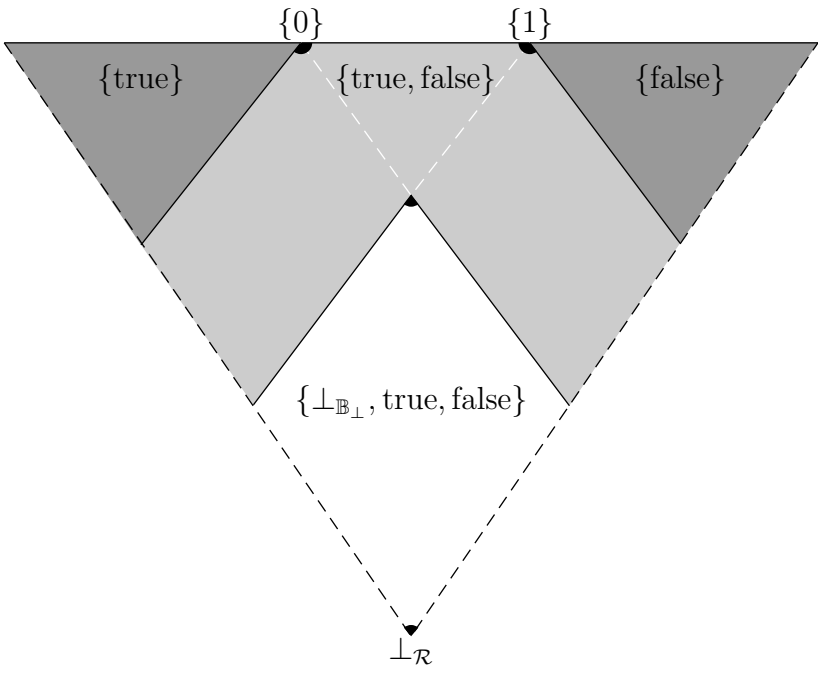


Figure 4.1: Values of ttest(l) for $l \in \mathcal{R}$.

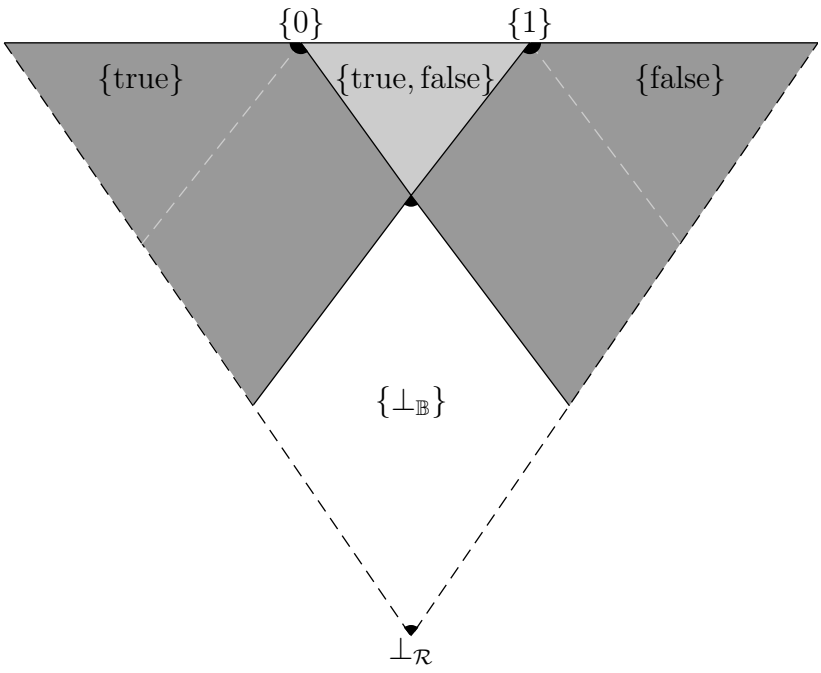


Figure 4.2: Values of $[\texttt{rtest}\, L]$ for $[L] \in \{\{x\} \mid x \in \mathcal{R}\}$.

4.3 Adequacy for total programs

This section is dedicated to proving the following theorem.

Theorem 4.3.1. *For any program M of ground type, $[\![M]\!] \sqsubseteq [M]$.*

In particular, if M denotes a total element, then equality holds. In itself, Theorem 4.3.1 does not say much about our semantics, because the semantics that assigns bottom to every term also satisfies the conclusion of the theorem. However, as we shall see in Chapters 5 (applications of the semantics) and 6 (universality), our model is "close enough" to the operational semantics.

Remark 4.3.1. It is not hard to adapt the proof of Theorem 4.3.1 to show that our semantics is fully adequate for the weak fragment of the language, in the sense that for any program M of ground type in which `rtest` does not occur, it holds that $[\![M]\!] = [M]$.

4.3.1 The Logical Relation $\trianglelefteq$ and the Main Lemma

To prove Theorem 4.3.1, we employ a standard technique, the use of a *logical relation* $\trianglelefteq$ that compares elements of the model with closed terms of corresponding types. Our presentation is influenced by the one found in textbook [26] to show the adequacy of the Scott model. Most of the work done in this section is to check that standard proofs carry through in our context and we only claim originality in the details that are specific to the particular language considered here.

Notation. From now on, we shorten our notation by writing $\mathsf{B} = \mathcal{P}^{\mathcal{S}}\,\mathbb{B}_{\perp}$, $\mathsf{N} = \mathcal{P}^{\mathcal{S}}\,\mathbb{N}_{\perp}$ and $\mathsf{R} = \mathcal{P}^{\mathcal{S}}\,\mathcal{R}$. The set of closed terms of type σ is written $\mathtt{Prg}_{\sigma}$.

Definition 4.3.1 (The logical relation $\trianglelefteq$)**.** The logical relation $\trianglelefteq$ is the family of the relations $\trianglelefteq_{\sigma}$, indexed by types and defined by the following clauses.
For all $x \in \mathsf{R}$ and all *closed terms* $M\colon \mathtt{real}$,

$$x \trianglelefteq_{\mathtt{real}} M \qquad \text{iff} \qquad x \sqsubseteq [M]\,. \tag{4.3.1}$$

For all $\forall b \in \mathsf{B}$, and all *closed terms* $M\colon \mathtt{bool}$,

$$b \trianglelefteq_{\mathtt{bool}} M \qquad \text{iff} \qquad b \sqsubseteq [M]\,. \tag{4.3.2}$$

For all $\forall n \in \mathsf{N}$, and all *closed terms* $M\colon \mathtt{nat}$,

$$n \trianglelefteq_{\mathtt{nat}} M \qquad \text{iff} \qquad n \sqsubseteq [M]\,. \tag{4.3.3}$$

For all higher types $\sigma \to \tau$, all $f \in D_{\sigma\to\tau}$ and all *closed terms* $M\colon \sigma \to \tau$,

$$f \trianglelefteq_{\sigma\to\tau} M \qquad \text{iff} \qquad \forall d \in D_{\sigma}.\ \forall N \in \mathtt{Prg}_{\sigma}.\ d \trianglelefteq_{\sigma} N \implies f(d) \trianglelefteq_{\tau} M(N). \tag{4.3.4}$$

Any type is of the form $\sigma_1, \ldots, \sigma_k \to \gamma$, where $\gamma \in \{\mathtt{bool}, \mathtt{nat}, \mathtt{real}\}$ and $k \in \mathbb{N}$, so the four clauses defining $\trianglelefteq$ can be replaced by the unique clause

$$f \trianglelefteq_{\sigma_1 \to \ldots \to \sigma_k \to \gamma} M \text{ iff}$$
$$\forall d_1 \trianglelefteq_{\sigma_1} N_1 \ldots \forall d_k \trianglelefteq_{\sigma_k} N_k \quad f(d_1)\ldots(d_k) \sqsubseteq [M(N_1)\ldots(N_k)]\,. \quad (4.3.5)$$

We say that $N_1, \ldots, N_k$ are *grounding* arguments for term M.

Theorem 4.3.1 is a direct consequence of the definition of $\trianglelefteq$ and the following lemma.

Lemma 4.3.2 (Main Lemma for $\trianglelefteq$)**.** *For all judgements* $x_1\colon \sigma_1, \ldots, x_k\colon \sigma_k \vdash M\colon \tau$,

$$[\![x_1\colon \sigma_1, \ldots, x_k\colon \sigma_k \vdash M\colon \tau]\!]\,(d_1, \ldots, d_k) \trianglelefteq_\tau M[N_1/x_1, \ldots, N_k/x_k]$$

for all elements $d_1 \in D_{\sigma_1}, \ldots, d_k \in D_{\sigma_k}$ *and all closed terms* $N_1\colon \sigma_1, \ldots, N_k\colon \sigma_k$ *such that* $d_1 \trianglelefteq_{\sigma_1} N_1, \ldots, d_k \trianglelefteq_{\sigma_k} N_k$.

If M is a program, that is a closed term and τ is either `real`, `nat` or `bool`, then Lemma 4.3.2 simply reads $[\![M]\!] \trianglelefteq_\tau M$, that is $[\![M]\!] \sqsubseteq [M]$, by definition of the logical relation $\trianglelefteq$ at ground types. We thus retrieve Theorem 4.3.1 as a particular case of Lemma 4.3.2.

The proof of Lemma 4.3.2 is in three parts. The first one shows that the operational behaviour of the constructors $\mathtt{bound}_a(-)$, $p + (-)$, $p \times (-)$, $\mathtt{rtest}(-)$, `succ` and `pred` is correct with respect to their denotation in the sense that $[\![C]\!]\,(M) \sqsubseteq [C(M)]$ for each constructor (Lemma 4.3.3). The second part is a verification that standard preliminary results concerning the recursion operator Y (see [26]) carry through. The third part is the proof of the Main Lemma itself, by induction on the derivation of typing judgements.

Preliminary results on basic constructors

Lemma 4.3.3. *Let* M *be a program of type* `real` *and* N *one of type* `nat`. *The following properties are satisfied.*

1. $\mathcal{P}^{\mathcal{S}}\,\mathrm{bound}_a\big([M]\big) \sqsubseteq [\mathtt{bound}_a(M)]$,
2. $\mathcal{P}^{\mathcal{S}}\,(p+)\,\big([M]\big) \sqsubseteq [p + (M)]$,
3. $\mathcal{P}^{\mathcal{S}}\,(p\times)\,\big([M]\big) \sqsubseteq [p \times (M)]$,
4. $\mathrm{ttest}^*\,\big([M]\big) \sqsubseteq [\mathtt{rtest}(M)]$,
5. $\mathcal{P}^{\mathcal{S}}\,\mathrm{succ}\big([N]\big) \sqsubseteq [\mathtt{succ}(N)]$ *and*
6. $\mathcal{P}^{\mathcal{S}}\,\mathrm{pred}\big([N]\big) \sqsubseteq [\mathtt{pred}(N)]$.

for all $a \in \mathcal{R}_{\mathbb{Q}}$ *and* $p \in \mathbb{Q}$.

We could show stronger results by replacing inequalities by equalities, except for the one concerning ttest, but this is not needed for Theorem 4.3.1. Moreover, one cannot in general obtain $\mathrm{ttest}^*\big([M]\big) = [\mathtt{rtest}(M)]$. For example, one has that

$$\Big[\mathtt{rtest}\left(\mathtt{bound}_{[0,\frac{1}{2}]}(\mathtt{Y}\lambda x.x)\right)\Big] = \{\mathrm{true}\}$$

whereas

$$\mathrm{ttest}^*\left(\left[\mathtt{bound}_{[0,\frac{1}{2}]}(\mathtt{Y}\lambda x.x)\right]\right) = \{\mathrm{true}, \mathrm{false}\}\,.$$

This is why we can only show the inequality $[\![M]\!] \sqsubseteq [M]$ and not the reverse one, $[\![M]\!] \sqsupseteq [M]$.

Proof. Notice that the three first inequalities take place in the power domain R, the fourth one in B and the last two ones in N. The operational meaning $[M]$ of a program M being by definition the greatest element containing the set of outputs of M, it suffices to show that $\mathrm{Outputs}\,(M) \subseteq d$ to show that $d \sqsubseteq [M]$ for a given element d in R, B or N. Hence, to prove the inequalities of Lemma 4.3.3, it suffices to show respectively the following properties about elements in the interval domain $\mathcal{R}$, in $\mathbb{N}_\perp$ or in $\mathbb{B}_\perp$. Recall from Section 3.1.3 that $|M|$ is the immediate output of a program M and that $\bigsqcup |M_n|$ is by definition the output of a path (M_n).

1. Let (M_n) be a computation from program $\mathtt{bound}_a(N_0)$. There exists a computation (N_n) from N_0 such that $\bigsqcup_n |M_n| = \mathrm{bound}_a\left(\bigsqcup_n |N_n|\right)$.

2. Let (M_n) be a computation from program $p + (N_0)$. There exists a computation (N_n) from N_0 such that $\bigsqcup_n |M_n| = p + \left(\bigsqcup_n |N_n|\right)$.

3. Let (M_n) be a computation from program $p \times (N_0)$. There exists a computation (N_n) from N_0 such that $\bigsqcup_n |M_n| = p \times \left(\bigsqcup_n |N_n|\right)$.

4. Let (M_n) be a computation from program $\mathtt{rtest}(N_0)$. There exists a computation (N_n) from N_0 such that $\bigsqcup_n |M_n| \in \mathrm{ttest}\left(\bigsqcup_n |N_n|\right)$.

5. Let (M_n) be a computation from program $\mathtt{succ}(N_0)$. There exists a computation (N_n) from N_0 such that $\bigsqcup_n |M_n| = \mathrm{succ}\left(\bigsqcup_n |N_n|\right)$.

6. Let (M_n) be a computation from program $\mathtt{pred}(N_0)$. There exists a computation (N_n) from N_0 such that $\bigsqcup_n |M_n| = \mathrm{pred}\left(\bigsqcup_n |N_n|\right)$.

Recall that ttest is defined as a function from $\mathcal{R}$ to B and notice that the relation $\bigsqcup_n |M_n| \in \text{ttest}\left(\bigsqcup_n |N_n|\right)$ is equivalent to the inequality $\text{ttest}^*\left(\uparrow\{\bigsqcup_n |N_n|\}\right) \sqsubseteq \uparrow\{\bigsqcup_n |M_n|\}$, which holds in B.

The proofs of these properties follow the same pattern. We only provide proofs of properties 1 and 4, as they are the most difficult. Properties 2 and 3 would be proved in a similar way as property 1. Property 5 and 6 would be proved in a similar way as property 4.

1. We prove the property concerning bound_a. Given a computation (M_n) from program $\texttt{bound}_a(N_0)$, we construct a computation from N_0 such that $\bigsqcup_n |M_n| = \text{bound}_a\left(\bigsqcup_n |N_n|\right)$.

Since (M_n) is an infinite computation (Proposition 3.2.3 on page 34), there exists a sequence of derivations (π_n) such that

i. for all $n \in \mathbb{N}$, the derivation π_n is a derivation of $M_n \to M_{n+1}$ and

ii. for infinitely many $n \in \mathbb{N}$, the derivation π_n is strong (*cf.* Chapter 3, Section 3.2).

Figure 4.3.1 page 55 provides an overview of how to define the path (N_n) and a corresponding sequence of derivations (π'_n) such that

i. for all $n \in \mathbb{N}$, the derivation π'_n is a derivation of $N_n \to N_{n+1}$ and

ii. for infinitely many $n \in \mathbb{N}$, the derivation π'_n is strong.

The idea is to define each N_n and $\pi'(n)$ according to the last steps in the proof π_n, or sometimes π_{n-1}. An important fact, which is a straightforward consequence of the definition of the reduction relation, is that each derivation π_n is either of the form

$$(16)\frac{\dfrac{\cdots}{N \to N'}}{\texttt{bound}_c(N) \to \texttt{bound}_{c'}(N')}|N| = \bot \tag{4.3.6}$$

or of the form

$$(15)\frac{}{\texttt{bound}_c\left(\texttt{bound}_b(N)\right) \to \texttt{bound}_{c\vec{\square}b}(N')} \tag{4.3.7}$$

and that there is a unique possibility among the two.

To focus on one difficulty at a time, we split the proof in three cases, as follows:

Case A: For all $n \in \mathbb{N}$, the last step in π_n is not an application of rule (15).

Case B: The last step in the *first* derivation π_0 is an application of rule (15).

Case C: Other cases. To construct path (N_n), we use a mixture of the techniques used in the two previous cases.

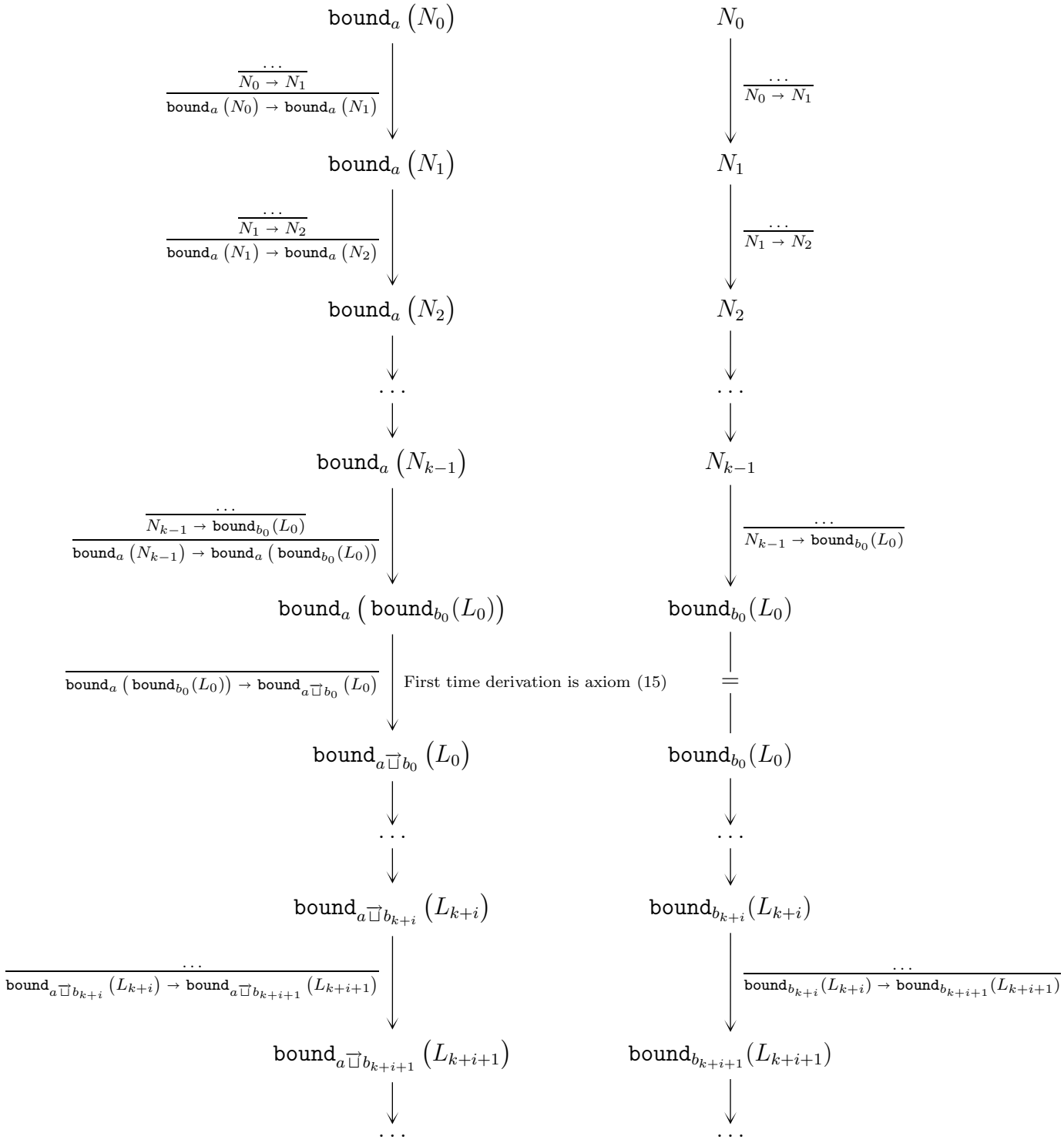


Figure 4.3: A computation from N_0, given a computation from $\mathtt{bound}_a(N_0)$.

Case A. No program M_n is of the form $\mathtt{bound}_{-}\left(\mathtt{bound}_{-}(-)\right)$. This implies that, for all $n \in \mathbb{N}$, program M_n is of the form $\mathtt{bound}_a(N_n)$ for some program N_n that is not of the form $\mathtt{bound}_{-}(-)$, hence $|N_n| = \bot_{\mathcal{R}}$, and the derivation π_n of $M_n \to M_{n+1}$ ends as follows:

$$(16)\frac{\dfrac{\cdots}{N_n \to N_{n+1}}}{\mathtt{bound}_a(N_n) \to \mathtt{bound}_a(N_{n+1})}.$$

By removing the last step from the derivation π_n, one obtains a derivation π'_n where

$$\pi'_n = \frac{\cdots}{N_n \to N_{n+1}}$$

of $N_n \to N_{n+1}$, for each $n \in \mathbb{N}$. If the derivation π_n is strong, then so is the derivation π'_n. Since the path M_n is fair, infinitely many derivations π_n are strong, by definition of a computation, and so infinitely many derivations π'_n are strong as well. So the sequence (N_n) is a path from N_0, and is fair. Furthermore, one has that $\mathrm{bound}_a\left(\bigsqcup |N_n|\right) = \mathrm{bound}_a(\bot_{\mathcal{R}}) = a = \bigsqcup |M_n|$.

Case B. $M_0 = \mathtt{bound}_a\left(\mathtt{bound}_{b_0}(L_0)\right)$ for some $b_0 \in \mathcal{R}_{\mathbb{Q}}$ and program L_0. Then M_0 can only be reduced to $\mathtt{bound}_{a \overrightarrow{\sqcup} b_0}(L_0)$ in one step, so $M_1 = \mathtt{bound}_{a \overrightarrow{\sqcup} b_0}(L_0)$.

We show by induction on n that, for each $n \geq 0$, there exist a program L_n and an element $b_n \in \mathcal{R}_{\mathbb{Q}}$ such that $M_{n+1} = \mathtt{bound}_{a \overrightarrow{\sqcup} b_n}(L_n)$ and $\mathtt{bound}_{b_{n-1}}(L_{n-1}) \to \mathtt{bound}_{a \overrightarrow{\sqcup} b_n}(L_n)$ if $n \geq 1$.

We have just seen that it is true for the base case $n = 0$.

Suppose as induction hypothesis that we have already defined $L_0, \ldots, L_n$ and $b_0, \ldots, b_n$ with the required conditions, in particular $M_{n+1} = \mathtt{bound}_{a \overrightarrow{\sqcup} b_n}(L_n)$. We proceed according to whether L_n is of the form $\mathtt{bound}_b(L')$.

If L_n is of the form $\mathtt{bound}_b(L')$ for some $b \in \mathcal{R}_{\mathbb{Q}}$ and some program L', then $M_{n+1} = \mathtt{bound}_{a \overrightarrow{\sqcup} b_n}\left(\mathtt{bound}_b(L')\right)$ and, since $M_{n+1} \to M_{n+2}$, one has that $M_{n+2} = \mathtt{bound}_{a \overrightarrow{\sqcup} b_n \overrightarrow{\sqcup} b}(L')$ because the program $\mathtt{bound}_{a \overrightarrow{\sqcup} b_n}\left(\mathtt{bound}_b(L')\right)$ can only be reduced to $\mathtt{bound}_{a \overrightarrow{\sqcup} b_n \overrightarrow{\sqcup} b}(L')$. Define $b_{n+1} = b_n \overrightarrow{\sqcup} b$ and $L_{n+1} = L'$. Program $\mathtt{bound}_{b_n}\left(\mathtt{bound}_b(L')\right)$ only reduces to $\mathtt{bound}_{b_n \overrightarrow{\sqcup} b}(L')$, *i.e.* $\mathtt{bound}_{b_n}(L_n) \to \mathtt{bound}_{b_{n+1}}(L_{n+1})$, hence $M_{n+2} = \mathtt{bound}_{a \overrightarrow{\sqcup} b_{n+1}}(L_{n+1})$.

If program L_n is not of the form $\mathtt{bound}_{-}(-)$ then program M_{n+1} is not of the form $\mathtt{bound}_{-}\left(\mathtt{bound}_{-}(-)\right)$. Hence the proof of $M_{n+1} \to M_{n+2}$ necessarily ends by

$$\frac{\dfrac{\cdots}{L_n \to L}}{\mathtt{bound}_{a \overrightarrow{\sqcup} b_n}(L_n) \to \mathtt{bound}_{a \overrightarrow{\sqcup} b_n}(L)}$$

for some term L. Define $b_{n+1} = b_n$ and $L_{n+1} = L$. Then $M_{n+2} = \mathtt{bound}_{a \overrightarrow{\sqcup} b_{n+1}}(L_{n+1})$, and, because L_n is not of the form $\mathtt{bound}_{-}(-)$ and $L_n \to L_{n+1}$ and $b_{n+1} = b_n$, we have that $\mathtt{bound}_{b_n}(L_n) \to \mathtt{bound}_{b_{n+1}}(L_{n+1})$, according to the derivation

$$\frac{\dfrac{\cdots}{L_n \to L_{n+1}}}{\mathtt{bound}_{b_n}(L_n) \to \mathtt{bound}_{b_n}(L_{n+1})}.$$

To conclude this second case, we define $N_n = \mathtt{bound}_{b_n}(L_n)$. Then

$$\begin{aligned}\bigsqcup_{n\geq 0} |M_n| &= \bigsqcup_{n\geq 1} |M_n| \\ &= \bigsqcup_{n\geq 0} |\,\mathtt{bound}_{a \overrightarrow{\sqcup} b_n}(L_n)| = \bigsqcup_{n\geq 0} (a \overrightarrow{\sqcup} b_n) = \mathtt{bound}_a\Big(\bigsqcup_{n\geq 0} b_n\Big) \\ &= \mathtt{bound}_a\Big(\bigsqcup_{n\geq 0} |\,\mathtt{bound}_{b_n}(L_n)|\Big) \\ &= \mathtt{bound}_a\Big(\bigsqcup_{n\geq 0} |N_n|\Big).\end{aligned}$$

Case C. There exists a least $k \in \mathbb{N}$, which is greater than 0, such that M_n is of the form $\mathtt{bound}_a\big(\mathtt{bound}_{b_0}(L_0)\big)$ for some program L_0 and some $b_0 \in \mathcal{R}_{\mathbb{Q}}$. For $n = 0, \ldots, k$, define N_n as in case 1, *i.e.* as the term matching $M_n = \mathtt{bound}_a(N_n)$. To define the remainder of the path (N_n), *i.e.* for $n > k$, first define the path $(N'_i)_{i\in\mathbb{N}}$ as in case 2, using the path $(M_{k+i})_{i\in\mathbb{N}}$ instead of $(M_n)_{n\in\mathbb{N}}$ and then define $N_{k+i} = N'_i$ for all $i \geq 1$. Since $N_k = \mathtt{bound}_{b_0}(L_0) = N'_0 \to N'_1 = N_{k+1}$, the sequence (N_n) is a path. It is easy to check that this path satisfies the required properties.

4. We show that the fourth property holds: for any computation (M_n) from a program of the form $\mathtt{rtest}(N_0)$, there exists a computation (N_n) from program N_0 such that $\bigsqcup_n |M_n| \in \mathrm{ttest}\big(\bigsqcup_n |N_n|\big)$. Let $N_0 \colon \mathtt{real}$ be a program and (M_n) be a computation from $\mathtt{rtest}\,(N_0)$.

Let us first suppose that the path (M_n) is infinite. This means that none of its terms is `true` or `false`, because neither `true` nor `false` can be reduced. In this case, the output of the path (M_n) is $\bot_{\mathbb{B}_\bot}$. Since (M_n) is a computation, there exists a sequence of derivations (π_n) such that

i. for all $n \in \mathbb{N}$, the derivation π_n is a derivation of $M_n \to M_{n+1}$ and

ii. for infinitely many $n \in \mathbb{N}$, the derivation π_n is strong.

Furthermore, all derivations π_n are of the form

$$(23)\frac{\dfrac{\cdots}{N_n \to N_{n+1}}}{\mathtt{rtest}(N_n) \to \mathtt{rtest}(N_{n+1})}.$$

By removing the last step in each π_n, we obtain derivations π'_n such that

i. for all $n \in \mathbb{N}$, the derivation π'_n is a derivation of $N_n \to N_{n+1}$ and

ii. for infinitely many $n \in \mathbb{N}$, the derivation π'_n is strong.

The immediate output of each N_n is such that $\texttt{rtest}\,(N_n)$ cannot be reduced to $\texttt{true}$ nor to $\texttt{false}$. Indeed, if it existed some N_k such that $\texttt{rtest}\,(N_k) \to \texttt{true}$ or $\texttt{rtest}\,(N_k) \to \texttt{false}$, this would mean that $N_k = \texttt{bound}_{b_0}(L)$ for some $b_0 \in \mathcal{R}_{\mathbb{Q}}$ such that $b_0 < 1$ or $b_0 > 0$. Then, because a term only reduces to terms with greater or equal outputs (Lemma 3.1.2), all subsequent terms N_{k+i}, for $i \in \mathbb{N}$, would be of the form $\texttt{bound}_{b_i}\,(L_i)$ with $b_i < 1$ or $b_i > 0$, and so all derivations π_{k+i} would be weak, contradicting the fact that (M_n) is a computation.

Therefore, the immediate outputs of all programs N_n are such that neither $|N_n| < 1$ nor $|N_n| > 0$, hence such that $|N_n| \sqsubseteq [0,1]$. So the output of the path (N_n), that is $\bigsqcup_n |N_n|$, is below $[0,1]$ and $\text{ttest}\,(\bigsqcup_n |N_n|) = \bot = \{\text{true}, \text{false}, \bot_{\mathbb{B}_\bot}\}$. We thus have that $\bigsqcup_n |M_n| \in \text{ttest}\left(\bigsqcup_n |N_n|\right)$.

Let us now suppose that the computation (M_n) is finite, of length $k+1$, that is term M_k is the last term of the path (M_n). Let $(\pi_n)_{0 \le n \le k}$ be a finite sequence of derivations such that the derivation π_n is a derivation of $M_n \to M_{n+1}$ for each $n < k$.

The case $|M_k| = \bot_{\mathbb{B}_\bot}$ is impossible as it would mean that $M_k = \texttt{rtest}(N_k)$ for some term N_k, which can always be reduced (*cf.* Prop. 3.2.3).

If now $|M_k| = \text{true}$ then $M_k = \texttt{true}$ and, for $n < k$, the program M_n is of the form $\texttt{rtest}(N_n)$ for some program $N_n \colon \texttt{real}$. Furthermore, the last derivation π_k must be of the form

$$(21)\frac{}{\texttt{rtest}(N_{k-1}) \to \texttt{true}}\,|N_{k-1}| < 1$$

and the other derivations are of the form

$$(23)\frac{\dfrac{\cdots}{N_n \to N_{n+1}}}{\texttt{rtest}(N_n) \to \texttt{rtest}(N_{n+1})}.$$

By removing the last step in the derivation π_n, we obtain a derivation of $N_n \to N_{n+1}$, for each $n \le k-2$. The finite path $(N_n)_{0 \le n \le k-1}$ can be extended to an infinite computation $(N_n)_{n \in \mathbb{N}}$ (Prop. 3.2.2). Since $\bigsqcup_{n \le k-1} |N_n| = |N_{k-1}| < 1$, we have that $\text{ttest}\left(\bigsqcup_{n \in \mathbb{N}} |N_n|\right) \supseteq \{\text{true}\}$ and hence that $\bigsqcup_{n \le k} |M_n| \in \text{ttest}\left(\bigsqcup_{n \in \mathbb{N}} |N_n|\right)$.

□

Preliminary results on recursion operators

The difficulty in proving adequacy is mainly due to the recursion operators $\texttt{Y}_\sigma$, which prevent simpler proofs by induction on the structure of terms. The three following lemmata and their proofs are standard ones adapted to our context. We recall the proofs as those show the core arguments that justify the use of a logical relation.

Lemma 4.3.4. *For any type $\sigma \equiv \sigma_1 \to \dots \sigma_k \to \gamma$, where γ is a ground type, for any term M of type $\sigma \to \sigma$ and any sequence of arguments $N_1 \colon \sigma_1, \dots, N \colon \sigma_k$,*

$$\left[M\big(\mathtt{Y}_\sigma(M)\big)(N_1)\dots(N_k)\right] \sqsubseteq \left[\mathtt{Y}_\sigma(M)(N_1)\dots(N_k)\right].$$

Proof. Let $y \in \text{Outputs}\big(\mathtt{Y}_\sigma(M)(N_0)\dots(N_k)\big)$. By definition of the set of outputs of a program, there exists a computation $(M_n)_{n\in\mathbb{N}}$ from $M_0 = \mathtt{Y}_\sigma(M)(N_0)\dots(N_k)$ such that $y = \bigsqcup_n |M_n|$. Since $\mathtt{Y}_\sigma(M)(N_0)\dots(N_k)$ can only reduce to $M\big(\mathtt{Y}_\sigma(M)\big)(N_1)\dots(N_k)$ in one step, the sequence $(M_{n+1})_{n\in\mathbb{N}}$ is a computation from term $M_1 = M\big(\mathtt{Y}_\sigma(M)\big)(N_0)\dots(N_k)$. So $y \in \text{Outputs}\big(M\big(\mathtt{Y}_\sigma(M)\big)(N_0)\dots(N_k)\big)$. The set of outputs of $\mathtt{Y}_\sigma(M)(N_0)\dots(N_k)$ is thus included in the set $\text{Outputs}\big(M\big(\mathtt{Y}_\sigma(M)\big)(N_0)\dots(N_k)\big)$. Therefore, the desired inequality $\left[M\big(\mathtt{Y}_\sigma(M)\big)(N_1)\dots(N_k)\right] \sqsubseteq \left[\mathtt{Y}_\sigma(M)(N_1)\dots(N_k)\right]$ holds, by definition of the operational meaning of programs. □

Lemma 4.3.5. *For all types σ and closed terms $M \colon \sigma$, the set $\{d \mid d \trianglelefteq_\sigma M\}$ is closed under directed suprema and contains $\bot_\sigma$.*

Proof. We prove Lemma 4.3.5 by induction on type σ. If sigma is a ground type `real`, `nat` or `bool`, the lemma holds trivially, by definition of the logical relation. Suppose as induction hypothesis that the lemma is satisfied for type τ, namely that for all closed terms L of type τ, the set $\{e \in D_\tau \mid e \trianglelefteq_\tau L\}$ is closed under directed suprema and contains $\bot_\tau$. We show that the lemma holds for types σ of the form $\kappa \to \tau$, where κ is any type.

For this purpose, let G be a directed subset of $\{f \in D_{\kappa\to\tau} \mid f \trianglelefteq_{\kappa\to\tau} M\}$. For any $d \in D_\kappa$ and $N \colon \kappa$ such that $d \trianglelefteq_\kappa N$, the set $\{g(d)\}_{g\in G}$ is a subset of $\{e \in D_\tau \mid e \trianglelefteq_\tau M(N)\}$, by definition of $\trianglelefteq_{\kappa\to\tau}$, for each $g \in G$. Because G is directed and by definition of the order $\sqsubseteq_{\kappa\to\tau}$ on $D_{\kappa\to\tau}$, the subset $\{g(d)\}_{g\in G}$ is also directed in D_τ. By induction hypothesis, its supremum $\bigsqcup_{g\in G}\big(g(d)\big)$ belongs to the set $\{e \in D_\tau \mid e \trianglelefteq_\tau M(N)\}$, *i.e.* $\big(\bigsqcup_{g\in G}\big(g(d)\big)\big) \trianglelefteq_\tau M(N)$. It follows that $\big(\bigsqcup G\big)(d) \trianglelefteq_\tau M(N)$, because the supremum $\bigsqcup G$ of G in $D_{\kappa\to\tau}$ satisfies the following equalities in D_τ: $\big(\bigsqcup G\big)(d) = \big(\bigsqcup_{g\in G} g\big)(d) = \bigsqcup_{g\in G}\big(g(d)\big)$. By definition of $\trianglelefteq_{\kappa\to\tau}$, this means that $\big(\bigsqcup G\big) \trianglelefteq_{\kappa\to\tau} M$. Hence the set $\{f \in D_{\kappa\to\tau} \mid f \trianglelefteq_{\kappa\to\tau} M\}$ is closed under directed suprema.

It remains to see that $\bot_{\kappa\to\tau} \trianglelefteq_{\kappa\to\tau} M$ for all closed terms $M \colon \kappa \to \tau$. Let M be a closed term of type $\kappa \to \tau$ and let $d \in D_\kappa$ and $N \colon \kappa$ such that $d \trianglelefteq_\kappa N$. Since $\bot_{\kappa\to\tau}(d) = \bot_\tau$ and $\bot_\tau \trianglelefteq_\tau M(N)$ by induction hypothesis, we have that $\bot_{\kappa\to\tau} \trianglelefteq_{\kappa\to\tau} M$, by definition of $\trianglelefteq_{\kappa\to\tau}$. So the set $\{f \in D_{\kappa\to\tau} \mid f \trianglelefteq_{\kappa\to\tau} M\}$ contains $\bot_{\kappa\to\tau}$. □

Lemma 4.3.6. *For all types σ, for all functions $f \in D_{\sigma\to\sigma}$ and closed terms $M \colon \sigma \to \sigma$, if $f \trianglelefteq_{\sigma\to\sigma} M$ then $\mu_\sigma(f) \trianglelefteq_\sigma \mathtt{Y}_\sigma(M)$.*

Proof. Suppose $f \trianglelefteq_{\sigma\to\sigma} M$, for some closed term $M \colon \sigma \to \sigma$ and some function $f \in D_{\sigma\to\sigma}$. By definition, we have that $\mu_\sigma(f) = \bigsqcup_{n\in\mathbb{N}}\big(f^n(\bot_\sigma)\big)$. So, by Lemma 4.3.4, to show that $\mu_\sigma(f) \trianglelefteq_\sigma \mathtt{Y}_\sigma(M)$, it suffices to show that $f^n(\bot_\sigma) \trianglelefteq \mathtt{Y}_\sigma(M)$ for all $n \in \mathbb{N}$. This is done by

induction on n. For $n = 0$, it is immediate as $f^0(\bot_\sigma) = \bot_\sigma$ and $\bot_\sigma \trianglelefteq_\sigma \mathtt{Y}_\sigma\,(M)$, again by Lemma 4.3.4. Now, suppose $f^n(\bot_\sigma) \trianglelefteq \mathtt{Y}_\sigma(M)$ for $n \geq 0$ and let us show that it also holds for $n+1$. Since we have both $f \trianglelefteq_{\sigma\to\sigma} M$ and $f^n(\bot_\sigma) \trianglelefteq_\sigma \mathtt{Y}_\sigma(M)$, it follows that $f(f^n(\bot_\sigma)) \trianglelefteq_{\sigma\to\sigma} M(\mathtt{Y}_\sigma\,(M))$, by definition of the logical relation. Hence, for any terms $N_0 \colon \sigma_0, \ldots, N_k \colon \sigma_k$ such that $MN_0 \ldots N_k$ is of ground type and elements $d_0 \trianglelefteq_{\sigma_0} N_0, \ldots, d_k \trianglelefteq_{\sigma_k} N_k$, it holds that $f^{n+1}\,(\bot_\sigma)\,(d_0)\ldots(d_k) \sqsubseteq [M\,(\mathtt{Y}_\sigma\,(M))\,(N_0)\ldots(N_k)]$, using characterization (4.3.5) of the logical relation. Since $[M(\mathtt{Y}\,(M))(N_0)\ldots(N_k)] \sqsubseteq [\mathtt{Y}_\sigma\,(M)(N_0)\ldots(N_k)]$ by Lemma 4.3.4, it follows that $f^{n+1}\,(\bot_\sigma)\,(d_0)\ldots(d_k) \sqsubseteq [\mathtt{Y}_\sigma\,(M)(N_0)\ldots(N_k)]$, that is $f^{n+1}(\bot_\sigma) \trianglelefteq_\sigma \mathtt{Y}_\sigma(M)$, which concludes the inductive argument. We have shown that $f^n(\bot_\sigma) \trianglelefteq_\sigma \mathtt{Y}_\sigma(M)$ for all $n \in \mathbb{N}$, which entails, as already argued, that $\mu_\sigma(f) \trianglelefteq_\sigma \mathtt{Y}_\sigma(M)$. □

Proof of Main Lemma

We have to show that, for any judgement $\overrightarrow{x\colon\sigma} \vdash M \colon \tau$,

$$\overrightarrow{d} \trianglelefteq_{\overrightarrow{\sigma}} \overrightarrow{N} \implies [\![\Gamma \vdash M \colon \tau]\!]\,(\overrightarrow{d}\,) \trianglelefteq_\tau M[\overrightarrow{N}/\overrightarrow{x}]$$

where $\overrightarrow{x\colon\sigma}$, $\overrightarrow{d}$ and $\overrightarrow{N}$ respectively stand for $x_0 \colon \sigma_0, \ldots, x_k \colon \sigma_k$, for $d_0, \ldots, d_k$ in $D_{\sigma_0}, \ldots, D_{\sigma_k}$ and for closed terms $N_0, \ldots, N_k$ of types $\sigma_0, \ldots, \sigma_k$. The implication should be understood as being quantified over all $\overrightarrow{d}$ and $\overrightarrow{N}$ of types given by $\overrightarrow{\sigma}$ but we will omit such quantifications.

Proof of Lemma 4.3.2. By induction on the derivation of typing judgements: for each inductive rule in Table 3.1 on page 20, we assume that

$$\overrightarrow{d} \trianglelefteq_{\overrightarrow{\sigma}} \overrightarrow{N} \implies [\![\overrightarrow{x\colon\sigma} \vdash M \colon \tau]\!]\,(\overrightarrow{d}\,) \trianglelefteq_\tau M[\overrightarrow{N}/\overrightarrow{x}] \tag{IH}$$

for each judgement $\overrightarrow{x\colon\sigma} \vdash M \colon \tau$ mentioned in the premise of the rule and we show that the same implication holds for the judgement in the conclusion of the rule.

Constants $\mathtt{bound}_a(-)$, $p + (-)$ and $p \times (-)$. Let $\overrightarrow{x\colon\sigma} \vdash M \colon \mathtt{real}$ be a judgement satisfying (IH) and let $\overrightarrow{d}$ and $\overrightarrow{N}$ be such that $\overrightarrow{d} \trianglelefteq_{\overrightarrow{\sigma}} \overrightarrow{N}$. By induction hypothesis, we have that

$$[\![\overrightarrow{x\colon\sigma} \vdash M \colon \mathtt{real}]\!]\,(\overrightarrow{d}\,) \trianglelefteq_{\mathtt{real}} M[\overrightarrow{N}/\overrightarrow{x}], \tag{4.3.8}$$

that is, by definition of $\trianglelefteq_{\mathtt{real}}$,

$$[\![\overrightarrow{x\colon\sigma} \vdash M \colon \mathtt{real}]\!]\,(\overrightarrow{d}\,) \sqsubseteq \left[M[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.9}$$

Then, by continuity of the function $\mathcal{P}^{\mathcal{S}}\,\mathrm{bound}_a \colon \mathsf{R} \to \mathsf{R}$,

$$\mathcal{P}^{\mathcal{S}}\,\mathrm{bound}_a\left(\left[\!\left[\overrightarrow{x\colon\sigma} \vdash M\colon \mathtt{real}\right]\!\right](\overrightarrow{d})\right) \sqsubseteq \mathcal{P}^{\mathcal{S}}\,\mathrm{bound}_a\left(\left[M[\overrightarrow{N}/\overrightarrow{x}]\right]\right). \tag{4.3.10}$$

By definition of the denotation of judgements (Table 4.2 on page 42), the left hand side is equal to $\left[\!\left[\overrightarrow{x\colon\sigma} \vdash \mathtt{bound}_a\,(M)\right]\!\right](\overrightarrow{d})$ and, by Lemma 4.3.3 on page 52, the right hand side is below $\left[\mathtt{bound}_a\left(M[\overrightarrow{N}/\overrightarrow{x}]\right)\right]$. Therefore

$$\left[\!\left[\overrightarrow{x\colon\sigma} \vdash \mathtt{bound}_a(M)\colon \mathtt{real}\right]\!\right](\overrightarrow{d}) \sqsubseteq \left[\mathtt{bound}_a\,(M)\,[\overrightarrow{N}/\overrightarrow{x}]\right], \tag{4.3.11}$$

that is

$$\left[\!\left[\overrightarrow{x\colon\sigma} \vdash \mathtt{bound}_a(M)\colon \mathtt{real}\right]\!\right](\overrightarrow{d}) \trianglelefteq_{\mathtt{real}} \mathtt{bound}_a\,(M)\,[\overrightarrow{N}/\overrightarrow{x}]. \tag{4.3.12}$$

We omit the proofs for the other constructors $p+(-)$ and $p\times(-)$, as these would be very similar to this one. (Just replace $\mathcal{P}^{\mathcal{S}}\,\mathrm{bound}_a$ by $\mathcal{P}^{\mathcal{S}}\,(p+)$ and $\mathtt{bound}_a$ by $p+$, etc.).

Constant $\mathtt{rtest}(-)$**.** Let $\overrightarrow{x\colon\sigma} \vdash M\colon \mathtt{real}$ be a judgement satisfying (IH) and let $\overrightarrow{d}$ and $\overrightarrow{N}$ be such that $\overrightarrow{d} \trianglelefteq_{\overrightarrow{\sigma}} \overrightarrow{N}$. By induction hypothesis, we have that

$$\left[\!\left[\overrightarrow{x\colon\sigma} \vdash M\colon \mathtt{real}\right]\!\right](\overrightarrow{d}) \trianglelefteq_{\mathtt{real}} M[\overrightarrow{N}/\overrightarrow{x}], \tag{4.3.13}$$

that is, by definition of $\trianglelefteq_{\mathtt{real}}$,

$$\left[\!\left[\overrightarrow{x\colon\sigma} \vdash M\colon \mathtt{real}\right]\!\right](\overrightarrow{d}) \sqsubseteq \left[M[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.14}$$

Then, by continuity of the function $\mathrm{ttest}^*\colon \mathsf{R} \to \mathsf{R}$,

$$\mathrm{ttest}^*\left(\left[\!\left[\overrightarrow{x\colon\sigma} \vdash M\colon \mathtt{real}\right]\!\right](\overrightarrow{d})\right) \sqsubseteq \mathrm{ttest}^*\left(\left[M[\overrightarrow{N}/\overrightarrow{x}]\right]\right). \tag{4.3.15}$$

By definition of the denotation of judgements (Table 4.2 on page 42), the left hand side is equal to $\left[\!\left[\overrightarrow{x\colon\sigma} \vdash \mathtt{rtest}\,(M)\right]\!\right](\overrightarrow{d})$ and, by Lemma 4.3.3 on page 52, the right hand side is below $\left[\mathtt{rtest}\left(M[\overrightarrow{N}/\overrightarrow{x}]\right)\right]$. Therefore

$$\left[\!\left[\overrightarrow{x\colon\sigma} \vdash \mathtt{rtest}(M)\colon \mathtt{real}\right]\!\right](\overrightarrow{d}) \sqsubseteq \left[\mathtt{rtest}\,(M)\,[\overrightarrow{N}/\overrightarrow{x}]\right], \tag{4.3.16}$$

that is

$$\left[\!\left[\overrightarrow{x\colon\sigma} \vdash \mathtt{rtest}(M)\colon \mathtt{real}\right]\!\right](\overrightarrow{d}) \trianglelefteq_{\mathtt{real}} \mathtt{rtest}\,(M)\,[\overrightarrow{N}/\overrightarrow{x}]. \tag{4.3.17}$$

Constants `true`, `false` and $\underline{0}$. Programs `true`, `false` and $\underline{0}$ cannot be reduced so their operational meanings are respectively {true}, {false} and {0}. So it holds trivially that true $\trianglelefteq_{\mathtt{bool}}$ `true`, false $\trianglelefteq_{\mathtt{bool}}$ `false` and that $0 \trianglelefteq_{\mathtt{nat}} \underline{0}$. Hence we have

$$\begin{aligned}
&[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{true}]\!]\,(\overrightarrow{d}) = \text{true} \trianglelefteq_{\mathtt{bool}} \mathtt{true} = \mathtt{true}[\overrightarrow{N}/\overrightarrow{x}],\\
&[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{false}]\!]\,(\overrightarrow{d}) = \text{true} \trianglelefteq_{\mathtt{bool}} \mathtt{false} = \mathtt{false}[\overrightarrow{N}/\overrightarrow{x}] \text{ and}\\
&[\![\overrightarrow{x\colon\sigma} \vdash \underline{0}]\!]\,(\overrightarrow{d}) = \{0\} \trianglelefteq_{\mathtt{nat}} \underline{0} = \underline{0}[\overrightarrow{N}/\overrightarrow{x}],
\end{aligned} \tag{4.3.18}$$

independently of $\overrightarrow{d}$ and $\overrightarrow{N}$.

The $\mathtt{if}_\gamma$ constructors. Assume (IH) holds for the three typing judgements $\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}$ and $\overrightarrow{x\colon\sigma} \vdash M_0\colon \gamma$ and $\overrightarrow{x\colon\sigma} \vdash M_1\colon \gamma$, where γ is a ground type. Let $\overrightarrow{d}$ and $\overrightarrow{N}$ be such that $\overrightarrow{d} \trianglelefteq_{\overrightarrow{\sigma}} \overrightarrow{N}$.

To prove that

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1) : \gamma]\!]\,(\overrightarrow{d}) \trianglelefteq_\gamma \mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}], \tag{4.3.19}$$

it suffices to show that

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) \trianglelefteq_\gamma \mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}] \tag{4.3.20}$$

or equivalently that

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) \sqsubseteq \left[\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]\right] \tag{4.3.21}$$

We proceed by cases on $[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!]$ to show that inequality 4.3.21 holds.

Case $[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\text{true}\}$. Then

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) = \left[\!\left[\overrightarrow{x\colon\sigma} \vdash M_0(\overrightarrow{d})\right]\!\right].$$

Our induction hypothesis implies that $[\![\overrightarrow{x\colon\sigma} \vdash M_0\colon \gamma]\!]\,(\overrightarrow{d}) \trianglelefteq_\gamma M_0[\overrightarrow{N}/\overrightarrow{x}]$. So

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) \sqsubseteq \left[M_0[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.22}$$

We also have by the induction hypothesis that

$$[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!]\,(\overrightarrow{d}) \sqsubseteq \left[L[\overrightarrow{N}/\overrightarrow{x}]\right], \tag{4.3.23}$$

so, since $[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\text{true}\}$, we also have $\left[L[\overrightarrow{N}/\overrightarrow{x}]\right] = \{\text{true}\}$, which means that all fair reduction paths from $L[\overrightarrow{N}/\overrightarrow{x}]$ output true, hence that all fair paths

from $\left[L[\overrightarrow{N}/\overrightarrow{x}]\right]$ are finite with the term true as their last term. It can then easily be shown from the definition of the reduction relation that any computation from the term $\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]$ has a suffix which is a computation from $M_0[\overrightarrow{N}/\overrightarrow{x}]$. The outputs of $\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]$ are thus outputs of $M_0[\overrightarrow{N}, \overrightarrow{x}]$ and one has

$$\left[M_0[\overrightarrow{N}/\overrightarrow{x}]\right] \sqsubseteq \left[\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.24}$$

Combining this last inequality with inequality 4.3.22, we obtain

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) \sqsubseteq \left[\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]\right], \tag{4.3.25}$$

as desired.

Case $[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\text{false}\}$ is similar. One has to replace true by false and M_0 by N_0 where relevant.

Case $[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!]\,(\overrightarrow{d}) = \{\text{true}, \text{false}\}$ is also similar, with the following differences. On the one hand we obtain that

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) \sqsubseteq \left[M_0[\overrightarrow{N}/\overrightarrow{x}]\right] \cup \left[N_0[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.26}$$

On the other hand we obtain that $\left[L[\overrightarrow{N}/\overrightarrow{x}]\right] \subseteq \{\text{true}, \text{false}\}$, which entails that the outputs of the term $\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]$ are either outputs of the term $M_0[\overrightarrow{N}/\overrightarrow{x}]$ or outputs of the term $N_0[\overrightarrow{N}/\overrightarrow{x}]$, from which it follows that

$$\left[M_0[\overrightarrow{N}/\overrightarrow{x}]\right] \cup \left[M_0[\overrightarrow{N}/\overrightarrow{x}]\right] \sqsubseteq \left[\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.27}$$

For the remaining case where $[\![\overrightarrow{x\colon\sigma} \vdash L\colon \mathtt{bool}]\!] = \{\bot_{\mathbb{B}_\bot}\}$ we have that

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) = \bot, \tag{4.3.28}$$

so it holds trivially that

$$[\![\overrightarrow{x\colon\sigma} \vdash \mathtt{if}_\gamma\,(L, M_0, M_1)]\!]\,(\overrightarrow{d}) \sqsubseteq \left[\mathtt{if}_\gamma\,(L, M_0, M_1)\,[\overrightarrow{N}/\overrightarrow{x}]\right]. \tag{4.3.29}$$

Remaining cases are standard but included for completeness.

Variables. Suppose the typing judgement $\overrightarrow{x}\colon\overrightarrow{\sigma} \vdash x_i\colon\sigma_i$ satisfies (IH)—with $x_i\colon\sigma_i$ appearing in context $\overrightarrow{x}\colon\overrightarrow{\sigma}$. If $\overrightarrow{d} \trianglelefteq_\sigma \overrightarrow{N}$, then, since $[\![\overrightarrow{x}\colon\overrightarrow{\sigma} \vdash x_i\colon\sigma_i]\!]\,(\overrightarrow{d}) = d_i$ and $x_i[\overrightarrow{N}/\overrightarrow{x}] = N_i$, we immediately have that $[\![\overrightarrow{x}\colon\overrightarrow{\sigma} \vdash x_i\colon\sigma_i]\!]\,(\overrightarrow{d}) \trianglelefteq_{\sigma_i} x_i[\overrightarrow{N}/\overrightarrow{x}]$.

Lambda abstraction. As induction hypothesis, we suppose that the typing judgement $\overrightarrow{x}\colon \overrightarrow{\sigma}, y\colon \rho \vdash M\colon \tau$ satisfies (IH) . We want to prove that judgement $\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash (\lambda y\colon \rho.M) : \rho \to \tau$ also satisfies (IH). Let $\overrightarrow{d}$ and $\overrightarrow{N}$ be such that $\overrightarrow{d} \trianglelefteq \overrightarrow{N}$. From our induction hypothesis, it follows that, for all $e \in D_\rho$ and all closed terms $E\colon \rho$,

$$e \trianglelefteq_\rho E \implies [\![\overrightarrow{x\colon \sigma}, y\colon \rho \vdash M\colon \tau]\!]\,(\overrightarrow{d}, e) \trianglelefteq_\tau M[\overrightarrow{N}/\overrightarrow{x}, E/e]. \tag{4.3.30}$$

By definition of $[\![\overrightarrow{x\colon \sigma} \vdash (\lambda y\colon \rho.M) : \rho \to \tau]\!]$, the above implication is rewritten as

$$e \trianglelefteq_\rho E \implies [\![\overrightarrow{x\colon \sigma} \vdash (\lambda y\colon \rho.M) : \rho \to \tau]\!]\,(\overrightarrow{d})(e) \trianglelefteq_\tau (M[\overrightarrow{N}/\overrightarrow{x}])[E/e]. \tag{4.3.31}$$

Therefore, by definition of $\trianglelefteq_{\rho\to\tau}$,

$$[\![\overrightarrow{x\colon \sigma} \vdash (\lambda y\colon \rho.M) : \rho \to \tau]\!]\,(\overrightarrow{d}) \trianglelefteq_{\rho\to\tau} (\lambda y\colon \rho.M)[\overrightarrow{N}/\overrightarrow{x}]. \tag{4.3.32}$$

Application. As induction hypothesis, we assume that (IH) holds for both judgements $\overrightarrow{x\colon \sigma} \vdash M\colon \rho \to \tau$ and $\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash L\colon \rho$.

Let $\overrightarrow{d}$ and $\overrightarrow{N}$ be such that $\overrightarrow{d} \trianglelefteq \overrightarrow{N}$. By induction hypothesis, we have

$$[\![\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash M\colon \rho \to \tau]\!]\,(\overrightarrow{d}) \trianglelefteq_{\rho\to\tau} M[\overrightarrow{N}/\overrightarrow{x}], \tag{4.3.33}$$

which, by definition of $\trianglelefteq_{\rho\to\tau}$ and because $[\![\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash L\colon \rho]\!]\,(\overrightarrow{d}) \trianglelefteq_\rho L[\overrightarrow{N}/\overrightarrow{x}]$, entails

$$[\![\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash M\colon \rho \to \tau]\!]\,(\overrightarrow{d})\big([\![\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash L\colon \rho]\!]\,(\overrightarrow{d})\big) \trianglelefteq_\tau (M[N/\overrightarrow{x}])\big(L[\overrightarrow{N}/\overrightarrow{x}]\big), \tag{4.3.34}$$

that is

$$[\![\overrightarrow{x}\colon \overrightarrow{\sigma} \vdash M(L)\colon \tau]\!]\,(\overrightarrow{d}) \trianglelefteq_\tau (M(L))[\overrightarrow{N}/\overrightarrow{x}]. \tag{4.3.35}$$

□

Chapter 5

Applications of the semantics

Marcial-Romero showed in his thesis [16] that the programming language described in Chapter 3 is expressive enough to compute a few common functions. However, his programs were proved correct using both denotational methods (to establish partial correctness, using a Hoare power domain semantics) and operational methods (to establish convergence and hence total correctness).

In this chapter, we justify our approach by showing that, with our approximate Smyth semantics, it becomes possible to define the same functions, and others, without invoking operational techniques. The fact that it is possible at all is not immediate, because there are programs whose denotational interpretations are strictly below their operational meanings. To give a simple example, consider a program M that computes some function; then the program $M' = \texttt{if}\,\texttt{rtest}(0)\ \texttt{then}\ M\ \texttt{else}\ \bot$ computes the same functions as M but the denotation of M' is the bottom element of the corresponding Smyth power domain. However, such an example is quite artificial and, as we shall see in this chapter, it is often possible to use `rtest` in such a way that the operational and denotational meanings of programs coincide, at least as far as total behaviour is concerned. Moreover, in the next chapter, we show that, using our approximate semantics of `rtest`, it is possible to define all first-order computable total functions on the reals.

We provide definitions and proofs of correctness for the following operators:

- the absolute value: $\mathbb{R} \to \mathbb{R}$,
- addition: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$,
- division by rational numbers : $\mathbb{R} \times \mathbb{Q} \setminus \{0\} \to \mathbb{R}$,
- the square root function $\sqrt{-}$: $[0, +\infty) \to \mathbb{R}$,
- an operator root: $(\mathbb{R} \to \mathbb{R}) \to \mathbb{R}$ that gives the zero of certain functions $\mathbb{R} \to \mathbb{R}$ with a unique zero,

- multiplication: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and the inverse function: $\mathbb{R} \setminus \{0\} \to \mathbb{R}$, and
- a limit operator lim: $(\mathbb{N} \to \mathbb{R}) \to \mathbb{R}$ that takes limits of certain Cauchy sequences of real numbers,

among a few others.

Organization

In Section 5.1, we introduce some convenient notation. In Section 5.2, we explain what it means for a program to *define* a partial function and relate this denotational property to the operational property of a program *computing* a partial function. In the remaining sections, we consider each function from the list above, as well as a few others. For each f among these functions, we exhibit a continuous function ϕ in the model that *represents* the function f (Definition 5.2.1) and is the denotation of a program defining f.

5.1 Working with the approximate model

Rather than writing programs, in this chapter we construct elements and functions of our denotational model in such a way that it is clear that they are denotable by terms of our language. In practice, most of these denotational constructions are similar to the corresponding, implicit programs.

We continue to use the notation for Smyth power domains introduced at the beginning of Section 4.3.1 on page 51 and write B, N and R for $\mathcal{P}^{\mathcal{S}}\,\mathbb{B}_\perp$, $\mathcal{P}^{\mathcal{S}}\,\mathbb{N}_\perp$ and $\mathcal{P}^{\mathcal{S}}\,\mathcal{R}$ respectively. From now on, we also use a simpler notation for continuous functions arising from the Smyth power construction by writing f rather than $\mathcal{P}^{\mathcal{S}}\,f$ or f^*. Thus:

$$\begin{aligned}
&\mathrm{bound}_a = \mathcal{P}^{\mathcal{S}}\,\mathrm{bound}_a \colon \mathsf{R} \to \mathsf{R},\\
&(p+) = \mathcal{P}^{\mathcal{S}}(p+) \colon \mathsf{R} \to \mathsf{R},\\
&(p\times) = \mathcal{P}^{\mathcal{S}}(p\times) \colon \mathsf{R} \to \mathsf{R},\\
&\mathrm{succ} = \mathcal{P}^{\mathcal{S}}\,\mathrm{succ} \colon \mathsf{N} \to \mathsf{N},\\
&\mathrm{pred} = \mathcal{P}^{\mathcal{S}}\,\mathrm{pred} \colon \mathsf{N} \to \mathsf{N},\\
&(0=) = \mathcal{P}^{\mathcal{S}}(0=) \colon \mathsf{N} \to \mathsf{B},\\
&\mathrm{if}_\gamma = \mathrm{if}^*_\gamma \colon \mathsf{B} \times D_\gamma \times D_\gamma \to D_\gamma \text{ and } \mathrm{if}\,x\,\mathrm{then}\,y\,\mathrm{else}\,z = \mathrm{if}_\gamma(x,y,z), \text{ and}\\
&\mathrm{ttest}_{0,1} = \mathrm{ttest} = \mathrm{ttest}^* \colon \mathsf{R} \to \mathsf{B}.
\end{aligned}$$

We will make use of the following notation in many of our definitions. For all $p, q \in \mathbb{Q}$ with $p < q$, all $x \in \mathsf{R}$ and elements y, z of the same domain,

$$\mathrm{ttest}_{p,q}\,(x) = \mathrm{ttest}\left(\frac{x-p}{q-p}\right)$$

and

$$\begin{array}{lrcl} \text{cases} & x \leq q & \to & y \\ & p \leq x & \to & z \end{array} = \text{if } \mathrm{ttest}_{p,q}(x) \text{ then } y \text{ else } z.$$

We write $x \leq q$ and $p \leq x$ rather that the strict inequalities $x < q$ and $p < x$ because we are talking about the model rather than the language. Notice that, for any $r \in \mathbb{R}$,

$$\begin{array}{lrcl} \text{cases} & \eta(r) \leq q & \to & y \\ & p \leq \eta(r) & \to & z \end{array} = \begin{cases} y & \text{if } r < p \\ y \cup z & \text{if } p \leq r \leq q \\ z & \text{if } q < r. \end{cases} \tag{5.1.1}$$

5.2 Domain representation of functions on the reals

We now make precise the concepts of representation and definability for first-order partial functions on the reals.

Definition 5.2.1. Define domains $\mathsf{R}^{(n)} \to \mathsf{R}$ by induction on n by letting $\mathsf{R}^{(0)} \to \mathsf{R}$ be R and $\mathsf{R}^{(n+1)} \to \mathsf{R}$ be $(\mathsf{R} \to (\mathsf{R}^{(n)} \to \mathsf{R}))$.

1. An element $x \in \mathsf{R}$ *represents* $r \in \mathbb{R}$ if $x = \eta_{\mathbb{R}}(r)$.
2. A function $\phi \in \left(\mathsf{R}^{(n)} \to \mathsf{R}\right)$ *represents* a function $f\colon A \to \mathbb{R}$, where $A \subseteq \mathbb{R}^n$, if whenever $(x_1, \ldots, x_n)$ represents $(r_1, \ldots, r_n) \in A$ (coordinatewise), we have that $\phi(x_1) \ldots (x_n)$ represents $f(r_1, \ldots, r_n)$.

Define types $\mathtt{real}^{(n)} \to \mathtt{real}$ by induction on n by letting $\mathtt{real}^{(0)} \to \mathtt{real}$ be $\mathtt{real}$ and $\mathtt{real}^{(n+1)} \to \mathtt{real}$ be $(\mathtt{real} \to (\mathtt{real}^{(n)} \to \mathtt{real}))$.

1. A program $M\colon \mathtt{real}$ *computes* $r \in \mathbb{R}$ if all computations of M output r (cf. Definition 3.1.5).
2. A program $F\colon \mathtt{real}^{(n)} \to \mathtt{real}$ *computes* a function $f\colon A \to \mathbb{R}$, where $A \subseteq \mathbb{R}^n$, if for all $M_1, \ldots, M_n\colon \mathtt{real}$ that compute $(r_1, \ldots, r_n) \in A$, we have that $FM_1 \ldots M_n$ computes $f(r_1, \ldots, r_n)$.

Finally, we say that

1. A program $M\colon \mathtt{real}$ *defines* a real number r if the denotation of M represents r.
2. A program $F\colon \mathtt{real}^{(n)} \to \mathtt{real}$ *defines* a function $f\colon A \to \mathbb{R}$, where $A \subseteq \mathbb{R}^n$, if the denotation of F represents f.

We also work with the obvious generalization of the above notions to first-order functions $A \to Y$ where $A \subseteq \prod_{i=0}^{n} X_i$ and where X_i and Y are among the sets $\mathbb{R}, \mathbb{N}, \mathbb{B}$.

Theorem 5.2.1.

1. *If a program* $M\colon \texttt{real}$ *defines a real number* r*, then* M *computes* r*.*
2. *If program* $F\colon \texttt{real}^{(n)} \to \texttt{real}$ *defines a function* $f\colon A \to \mathbb{R}$*, where* $A \subseteq \mathbb{R}^n$*, and if* $M_1, \ldots, M_n\colon \texttt{real}$ *define* $(r_1, \ldots, r_n) \in A$ *respectively, then* $FM_1, \ldots M_n$ *computes* $f(r_1, \ldots, r_n)$*.*

Proof. The first item follows directly from Theorem 4.3.1 (Adequacy) because the representation of any real number is a maximal element, and the second follows from the first and by definition of function definability. □

Notice, however, that one cannot conclude that F computes f. But, because every computable real number is definable in the language, the conclusion formulated in the above theorem is useful for computing values of f from a program F that defines f. As discussed above, in this chapter we explicitly provide specific examples of domain representations of functions on the reals, and implicitly provide programs that define such functions.

5.3 Absolute value

The denotation of the closed term we gave earlier (Section 3.2, page 36) for the absolute value function is the map $\mathrm{abs}\colon \mathsf{R} \to \mathsf{R}$ considered in the next theorem.

Theorem 5.3.1. *The following recursively defined function represents the absolute value map on real numbers:*

$$\begin{array}{l} \mathrm{abs}\colon \mathsf{R} \to \mathsf{R} \\ \mathrm{abs}\,(x) = \text{cases}\ \ x \leq 0 \ \ \to\ \ -x \\ \qquad\qquad\quad -1 \leq x \ \ \to\ \ \text{cases}\ \ x \leq 1 \ \ \to\ \ \mathrm{bound}_{[-1,1]}\left(\tfrac{1}{2}\,\mathrm{abs}(2x)\right) \\ \qquad\qquad\qquad\qquad\quad\ \ 0 \leq x \ \ \to\ \ x. \end{array} \tag{5.3.1}$$

Proof. We first show by induction on n that whenever $|r| > \frac{1}{2^n}$, it holds that $\mathrm{abs}(\eta\,(r)) = \eta\,(|r|)$. It is immediate that it holds for $r < -1$ or $r > 1$, *i.e.* when $|r| > \frac{1}{2^0}$. Suppose as induction hypothesis that it holds for $n \in \mathbb{N}$ and consider $r \in \mathbb{R}$ such that $\frac{1}{2^{n+1}} < r \leq 1$. We assume r positive; the negative case would be treated similarly. By definition of the function abs, because $0 < r \leq 1$, we have that

$$\mathrm{abs}\,(\eta\,(r)) = \eta\,(|r|) \cup \left\{ \mathrm{bound}_{[-1,1]}\left(\frac{1}{2}\mathrm{abs}\left(2\eta\,(r)\,\right)\right)\right\}. \tag{5.3.2}$$

Since $2\eta(r) = \eta(2r)$ and $2r > \frac{1}{2^n}$, it holds by induction hypothesis that $\mathrm{abs}\big(2\eta(r)\big) = \eta(|2r|)$ and hence $\frac{1}{2}\mathrm{abs}\big(2\eta(r)\big) = \eta(|r|)$. Since furthermore we have $|r| \leq 1$, it follows that $\mathrm{bound}_{[-1,1]}\big(\frac{1}{2}\mathrm{abs}\big(2\eta(r)\big)\big) = \eta(|r|)$. So equality 5.3.2 can be simplified to $\mathrm{abs}(\eta(r)) = \eta(|r|)$.

It remains to see that $\mathrm{abs}(\eta(0)) = \eta(|0|)$. By definition of the function abs,

$$\mathrm{abs}\big(\eta(0)\big) = \mathrm{bound}_{[-1,1]}\left(\frac{1}{2}\,\mathrm{abs}\big(\eta(0)\big)\right) \cup \eta(0) \tag{5.3.3}$$

$$= \frac{1}{2}\,\mathrm{bound}_{[-2,2]}\big(\,\mathrm{abs}(\eta(0)) \cup \eta(0)\,\big). \tag{5.3.4}$$

Thus

$$\mathrm{abs}\big(\eta(0)\big) \cup \eta(0) = \frac{1}{2}\,\mathrm{bound}_{[-2,2]}\big(\,\mathrm{abs}(\eta(0)) \cup \eta(0)\,\big), \tag{5.3.5}$$

so $\mathrm{abs}(\eta(0)) \cup \eta(0)$ is the unique solution of the equation $x = \frac{1}{2}\,\mathrm{bound}_{[-2,2]}(x)$, that is $\eta(0)$. Therefore $\mathrm{abs}(\eta(0)) = \eta(0)$. □

It follows from Theorem 4.3.1 (Adequacy) that the program discussed in Section 3.2 computes absolute values, as explained in Section 5.2.

5.4 Identity on real numbers

Before showing that the addition operation on real numbers is definable, in the next section, we consider the case of the identity function because it is simpler and addition will be treated in a similar way. The identity on $\mathbb{R}$ can of course be defined by $\lambda x\colon \mathtt{real}.x$, whose denotation is the identity on $\mathcal{P}^{\mathcal{S}}\mathcal{R}$. Nevertheless, we give another representation of $\mathrm{id}_{\mathbb{R}}$ that is not id_{R}, which works by de-constructing its argument (with ttest) and then reconstructing it (with bound).

Theorem 5.4.1. *The following recursively defined function represents the identity on the reals:*

$$\begin{array}{l}
\mathrm{id}'\colon \mathsf{R} \to \mathsf{R} \\
\mathrm{id}'(x) = \text{cases}\;\; x \leq -2 \;\;\to\;\; -2 + \mathrm{id}'(x+2) \\
\qquad\qquad\quad\;\; x \geq -3 \;\;\to\;\; \text{cases}\;\; x \geq 2 \;\;\to\;\; 2 + \mathrm{id}'(x-2) \\
\qquad\qquad\qquad\qquad\qquad\;\; x \leq 3 \;\;\to\;\; \mathrm{bound}_{[-3,3]}\big(\tfrac{1}{4}\,(\mathrm{id}'(4x))\big)\,.
\end{array}$$

The proof of this theorem will be in two steps, each using a technical lemma. The first lemma will be used to show that $\mathrm{id}'(\eta(r)) \sqsubseteq \eta(r)$ for all reals r, and the second to show that id' is total.

Lemma 5.4.2. *Let D, E be dcpo's, $f_0 \colon D \to E$ a function that sends maximal elements to maximal elements and $F \colon (D \to E) \to (D \to E)$ a continuous operator such that, whenever a function $f \colon D \to E$ agrees with f_0 at maximal elements, then so does $F(f)$. Then $\mu(F)(d) \sqsubseteq f_0(d)$ for all total elements d in D.*

Proof. By induction on $n \in \mathbb{N}$, the hypothesis entails that each iterate $F^n(f)$ also agrees with f_0 at total elements, provided that f does. In particular, if f is f_0, we have by monotonicity of F^n that $F^n(\bot)(x) \sqsubseteq F^n(f_0)(x) = f_0(x)$, for all maximal x in D and all $n \in \mathbb{N}$. Therefore $\bigsqcup_n F^n(\bot)(x) \sqsubseteq f_0(x)$, *i.e.* $\mu F(x) \sqsubseteq f_0(x)$, for all maximal x in D. □

The second lemma makes use of two related concepts to give a sufficient condition for a function to be total: the *diameter* of elements in R, useful to characterize maximal elements as those with a null diameter, and *non-expansive* functions, which are those functions that don't increase the diameter.

Definition 5.4.1. For any $a \in \mathsf{R}$, the diameter $\delta(a)$ of a is the supremum of distances between any two real numbers in $\cup a$, the union of the real intervals in a:

$$\delta(a) = \sup \left\{ |r - r'| \;\middle|\; r, r' \in \bigcup_{x \in a} x \right\}. \tag{5.4.1}$$

A function $\psi \colon \mathsf{R} \to \mathsf{R}$ is *non expansive* if $f(\delta(a)) \leq \delta(a)$ for all $a \in \mathsf{R}$.

Lemma 5.4.3. *Let ϕ be a function from R to R and d and $\varepsilon < 1$ be two positive reals such that, for each* total *element x in R, there is an interval a in $\mathcal{R}_{\mathbb{Q}}$ and some* total *functions $\psi_0, \ldots, \psi_k, \xi_0, \ldots, \xi_k \colon \mathsf{R} \to \mathsf{R}$ satisfying the following conditions:*

$$\phi(x) \sqsupseteq \frac{\varepsilon}{2} \operatorname{bound}_a \left(\bigcup_{0 \leq i \leq k} \psi_i \circ \phi \circ \xi_i(x) \right) \tag{5.4.2a}$$

$$\delta(a) \leq d, \tag{5.4.2b}$$

for all $0 \leq i \leq k$, the function ψ_i is non expansive, and (5.4.2c)

$$\bigcap_{0 \leq i \leq k} \psi_i \circ \phi \circ \xi_i(x) \text{ is not empty.} \tag{5.4.2d}$$

Then ϕ is total.

Notice that the interval a and the functions ψ_i and ξ_i depend not only on ϕ but also on the given total $x \in \mathsf{R}$.

Proof. We prove by induction on n that

$$\text{for all total } x \in \mathsf{R}, \quad \delta\big(\phi(x)\big) \leq \varepsilon^n d, \tag{5.4.3}$$

for all natural numbers n, which entails that $\delta\big(\phi(x)\big) = 0$, and hence that $\phi(x)$ is total for all total elements x, that is ϕ is total.
By inequality 5.4.2a and because $\varepsilon < 1$, we have that $\delta(\phi(x)) \leq \delta(\text{bound}_a(\bot)) \leq \delta(\eta\,(a)) \leq d$, so property 5.4.3 holds for $n = 0$. As induction hypothesis, suppose that $\delta\big(\phi(y)\big) \leq \varepsilon^n d$ for all total $y \in \mathsf{R}$, for some $n \geq 0$. Let $x \in \mathsf{R}$ be total; by 5.4.2a it holds that

$$\phi(x) \sqsupseteq \frac{\varepsilon}{2}\,\text{bound}_a\Big(\bigcup_{0\leq i\leq k} \psi_i \circ \phi \circ\ \xi_i(x)\Big)$$

for some total functions ψ_i and ξ_i satisfying conditions 5.4.2b to 5.4.2d. By induction hypothesis, we have that $\delta\big(\phi \circ \xi_i(x)\big) \leq \varepsilon^n d$ and, because ψ_i is non expansive, that $\delta\big(\psi_i \circ \phi \circ \xi_i(x)\big) \leq \varepsilon^n d$, for each i with $0 \leq i \leq k$. Since by hypothesis 5.4.2d there is a point r_0 such that $\eta\,(r_0)$ is above each $\psi_i \circ \phi \circ \xi_i(x)$, any other point r such that $\eta\,(r)$ is above the union $\bigcup_i \psi_i \circ \phi \circ \xi_i(x)$ is at a distance smaller or equal to $\varepsilon^n d$ from r_0. The diameter of $\bigcup_i \psi_i \circ \phi \circ \xi_i(x)$ is thus bounded by twice that distance: $\delta\big(\bigcup_i \psi_i \circ \phi \circ \xi_i(x)\big) \leq 2\varepsilon^n d$. Since the function bound_a is itself non expansive and

$$\phi(x) \sqsupseteq \frac{\varepsilon}{2}\,\text{bound}_a\Big(\bigcup_{0\leq i\leq k} \psi_i \circ \phi \circ\ \xi_i(x)\Big),$$

it follows that $\delta\big(\phi(x)\big) \leq \varepsilon^{n+1} d$. □

Proof of Theorem 5.4.1. We show that $\text{id}'(\eta\,(r)) = \eta\,(r)$ for all $r \in \mathbb{R}$ by proving that

1. the function id' is below the identity at real numbers, *i.e.* $\forall r \in \mathbb{R} \quad \text{id}'(\eta\,(r)) \sqsubseteq \eta\,(r)$, and
2. id' is total, *i.e.* $\forall r \in \mathbb{R}, \exists r' \in \mathbb{R} \quad \text{id}'(\eta\,(r)) = \eta\,(r')$.

By definition, the function id' is the least fixed point of F defined by

$$\begin{array}{l} F\colon (\mathsf{R} \to \mathsf{R}) \to (\mathsf{R} \to \mathsf{R}) \\ F(f) = \lambda x.\,\text{cases}\ \ x \leq -2 \ \ \to \ \ -2 + f(x+2) \\ \qquad\qquad\qquad\quad\ \ x \geq -3 \ \ \to \ \ \text{cases}\ \ x \geq 2 \ \ \to \ \ 2 + f(x-2) \\ \qquad\qquad\qquad\qquad\qquad\qquad\ \ x \leq 3 \ \ \to \ \ \text{bound}_{[-3,3]}\left(\frac{1}{4}\,(f(4x))\right) \end{array} \tag{5.4.4}$$

1. It is immediate from Equation 5.4.4 that, if a function $f\colon \mathsf{R} \to \mathsf{R}$ agrees with the function $\lambda x.x$ at real numbers, then so does the function $F(f)$:

$$\big(\forall r \in \mathbb{R}, f(\eta\,(r)) = \eta\,(r)\,\big) \implies \big(\forall r \in \mathbb{R}, F(f)(\eta\,(r)) = \eta\,(r)\,\big). \tag{5.4.5}$$

Hence, the fact that id' is below the identity at real numbers is an immediate consequence of Lemma 5.4.2 applied to F and $\lambda x.x$ as f_0.

2. We now show that id' is total as a consequence of Lemma 5.4.3, when taking $\varepsilon = \frac{1}{2}$ and $d = 40$. To apply this lemma, we need to prove that for each *total* $x \in \mathsf{R}$, there exist $a \in \mathcal{R}_{\mathbb{Q}}$ and four total functions $\psi_0, \psi_1, \xi_0, \xi_1 \colon \mathsf{R} \to \mathsf{R}$ such that

$$\mathrm{id}'(x) \sqsupseteq \frac{1}{4}\,\mathrm{bound}_a\Big(\big(\psi_0 \circ \mathrm{id}' \circ\ \xi_0(x)\big) \cup \big(\psi_1 \circ \mathrm{id}' \circ\ \xi_1(x)\big)\Big) \tag{5.4.6a}$$

$$\delta(a) \leq 40, \tag{5.4.6b}$$

$$\psi_0, \psi_1 \text{ are non expansive, and} \tag{5.4.6c}$$

$$\psi_0 \circ \mathrm{id}' \circ\ \xi_0(x) \text{ and } \psi_1 \circ \mathrm{id}' \circ\ \xi_1(x) \text{ overlap.} \tag{5.4.6d}$$

We proceed by cases on x.

Case $-2 < x < 2$. By definition of id', we have that

$$\mathrm{id}'(x) = \mathrm{bound}_{[-3,3]}\left(\frac{\mathrm{id}'(4x)}{4}\right) \tag{5.4.7}$$

$$= \frac{1}{4}\,\mathrm{bound}_{[-12,12]}\big(\mathrm{id}'(4x)\big). \tag{5.4.8}$$

We define $a = [-12, 12]$, $\psi_0 = \psi_1 = \mathrm{id}_{\mathcal{R}}$ and $\xi_0 = \xi_1 = \lambda x.4x$. It is easy to see that properties 5.4.6 are satisfied with those choices of $\psi_0, \psi_1, \xi_0, \xi_1$.

Case $2 \leq x \leq 3$. For all total elements x such that $2 \leq x \leq 3$, we have that

$$\mathrm{id}'(x) = \left(2 + \mathrm{id}'\,(x-2)\right) \cup \left(\ \mathrm{bound}_{[-3,3]}\left(\frac{\mathrm{id}'\,(4x)}{4}\right)\right), \tag{5.4.9}$$

by definition of id'. Since $x-2$ is total and $-2 < x-2 < 2$, we obtain from equality 5.4.8 applied to $x-2$ that

$$\mathrm{id}'(x) = \left(2 + \frac{1}{4}\,\mathrm{bound}_{[-12,12]}\left(\mathrm{id}'\,(4(x-2))\ \right)\right) \cup \left(\mathrm{bound}_{[-3,3]}\left(\frac{\mathrm{id}'\,(4x)}{4}\right)\right) \tag{5.4.10}$$

$$= \left(\frac{1}{4}\,\mathrm{bound}_{[-4,20]}\,(8 + \mathrm{id}'(4(x-2)))\ \right) \cup \left(\frac{1}{4}\,\mathrm{bound}_{[-12,12]}\,(\mathrm{id}'\,(4x))\ \right)$$

$$= \frac{1}{4}\,\mathrm{bound}_{[-12,20]}\left(\ \mathrm{bound}_{[-4,20]}\left(8 + \mathrm{id}'\,\big(4(x-2)\big)\right) \cup \mathrm{bound}_{[-12,12]}\left(\mathrm{id}'\,(4x)\ \right)\right).$$

Define $a = [-12, 20]$, $\psi_0 = \lambda x.\,\mathrm{bound}_{[-4,20]}\,(8 + x)$, $\psi_1 = \mathrm{bound}_{[-12,12]}$, $\xi_0 = \lambda x.\,(4(x-2))$ and $\xi_1 = \lambda x.4x$. It is easy to see that properties 5.4.6 are then satisfied. In particular,

we know from part 1 of the proof that $\mathrm{id}'(4(x-2)) \sqsubseteq 4(x-2)$ and $\mathrm{id}'(4x) \sqsubseteq 4x$, from which it follows that $4x$ is above both $\psi_0 \circ \mathrm{id}' \circ \xi_0(x)$ and $\psi_1 \circ \mathrm{id}' \circ \xi_1(x)$.

Case $-3 \leq x \leq -2$. A similar argument as the previous one leads to

$$\mathrm{id}'(x) = \frac{1}{4}\,\mathrm{bound}_{[-12,20]}\left(\mathrm{bound}_{[-4,20]}\left(-8+\mathrm{id}'\left(4(x+2)\right)\right) \cup \mathrm{bound}_{[-12,12]}\left(\mathrm{id}'\left(4x\right)\right)\right)$$

and hence one can chose $a = [-12, 20]$, $\psi_0 = \lambda x.\,\mathrm{bound}_{[-4,20]}\,(-8+x)$, $\xi_0 = \lambda x.4x+2$, $\psi_1 = \mathrm{bound}_{[-12,12]}$ and $\xi_1 = \lambda x.4x$.

The other cases are similar, but we include them for the sake of completeness.

Other cases. So far, we know that property 5.4.6 is satisfied for all total elements x such that $-3 \leq x \leq 3$. To show that it is satisfied for all total elements x, we prove by induction on n that it is satisfied when $-n \leq x \leq n$, for all $n \in \mathbb{N}$. Suppose as induction hypothesis that property 5.4.6 holds for $-n \leq x \leq n$ with $n \geq 3$ and let us show that it holds for $n < x \leq n+1$ and $-n-1 \leq x < n$. We only proceed for x positive; the negative case is similar.

Let thus x be a total element such that $n < x \leq n+1$ and let a, ψ_0, ψ_1, ξ_0, ξ_1 be such that conditions 5.4.6 are satisfied for $x-2$. Such a, ψ_0, ψ_1, ξ_0, ξ_1 exist by the induction hypothesis, because $n < x-2 \leq n$. They satisfy

$$\mathrm{id}'(x-2) \sqsupseteq \frac{1}{4}\,\mathrm{bound}_a\left(\left(\psi_0 \circ \mathrm{id}' \circ\ \xi_0(x-2)\right) \cup \left(\psi_1 \circ \mathrm{id}' \circ\ \xi_1(x-2)\right)\right) \tag{5.4.11}$$

and, by definition of id', it holds that $\mathrm{id}'(x) = 2 + \mathrm{id}'(x-2)$, so

$$\mathrm{id}'(x) \sqsupseteq 2 + \frac{1}{4}\,\mathrm{bound}_a\left(\left(\psi_0 \circ \mathrm{id}' \circ\ \xi_0(x-2)\right) \cup \left(\psi_1 \circ \mathrm{id}' \circ\ \xi_1(x-2)\right)\right) \tag{5.4.12}$$

$$\sqsupseteq \frac{1}{4}\,\mathrm{bound}_{8+a}\left(\left((8+\psi_0) \circ \mathrm{id}' \circ\ \xi_0(x-2)\right) \cup \left((8+\psi_1) \circ \mathrm{id}' \circ\ \xi_1(x-2)\right)\right) \tag{5.4.13}$$

where $8+\psi_i$ is a notation for the function $\lambda y.8+\psi_i(y)$. Define $\psi_i' = 8+\psi_i$ and $\xi_i' = \lambda x.\xi_i(x-2)$; then we have that

$$\mathrm{id}'(x) \sqsupseteq \frac{1}{4}\,\mathrm{bound}_{8+a}\left(\left(\psi_0' \circ \mathrm{id}' \circ\ \xi_0'(x)\right) \cup \left(\psi_1' \circ \mathrm{id}' \circ\ \xi_1'(x)\right)\right) \tag{5.4.14}$$

Furthermore, because a, ψ_0, ψ_1, ξ_0 and ξ_1 satisfy conditions 5.4.7, it is easy to check that the following requirements are met: $\delta(8+a) \leq 40$, ψ_0' and ψ_1' are total and non expansive, ξ_0' and ξ_1' are total and $\psi_0' \circ \mathrm{id}' \circ\ \xi_0'(x)$ and $\psi_1' \circ \mathrm{id}' \circ \xi_1'(x)$ overlap.

□

5.5 Addition

The definition and proof for addition follow the same pattern as the ones for the identity function in the previous section.

Theorem 5.5.1 (Addition)**.** *The following recursively defined function represents the addition operation:*

$$
\begin{array}{l}
\mathrm{add}\colon (\mathsf{R}\times\mathsf{R}\to\mathsf{R})\\
\mathrm{add}(x,y) =\\
\begin{array}{llll}
\text{cases}\\
x\geq 2 & \to & 2+\mathrm{add}(x-2,y)\\
x\leq 3 & \to & \text{cases}\\
 & & x\leq -2 & \to & -2+\mathrm{add}(x+2,y)\\
 & & x\geq -3 & \to & \text{cases}\\
 & & & & y\geq 2 & \to & 2+\mathrm{add}(x,y-2)\\
 & & & & y\leq 3 & \to & \text{cases}\\
 & & & & & & y\leq -2 & \to & -2+\mathrm{add}(x,y+2)\\
 & & & & & & y\geq -3 & \to & \mathrm{bound}_{[-6,6]}\left(\frac{1}{4}\,\mathrm{add}(4x,4y)\right).
\end{array}
\end{array}
$$

We again use the concepts of diameter and non-expansive functions (definition 5.4.1) and the two corresponding lemmas below. Lemma 5.5.2 is simply a special case of Lemma 5.4.2. The proof of Lemma 5.5.3 is not given as it is essentially the same as for Lemma 5.4.3.

Lemma 5.5.2. *Let C, D, E be dcpo's, $f_0 : C \times D \to E$ a function sending maximal elements to maximal elements and $F\colon (C\times D \to E) \to (C\times D \to E)$ a continuous operator such that, whenever a function $f\colon C \times D \to E$ agrees with f_0 at maximal elements, then so does $F(f)$. Then $\mu(F)(c,d) \sqsubseteq f_0(c,d)$ for all maximal elements (c,d) in $C \times D$.*

Lemma 5.5.3. *Let ϕ be a function from $\mathsf{R}\times\mathsf{R}$ to R and d and $\varepsilon < 1$ two positive reals such that, for each* total *element (x,y) in $\mathsf{R}\times\mathsf{R}$, there is an interval a in $\mathcal{R}_{\mathbb{Q}}$ and some* total *functions $\psi_0,\ldots,\psi_k\colon \mathsf{R}\to\mathsf{R}$ and $\xi_0,\ldots,\xi_k\colon \mathsf{R}\times\mathsf{R}\to\mathsf{R}\times\mathsf{R}$ satisfying the following conditions:*

$$\phi(x,y) \sqsupseteq \frac{\varepsilon}{2}\,\mathrm{bound}_a\left(\bigcup_{0\leq i\leq k}\psi_i\circ\phi\circ\xi_i(x,y)\right) \tag{5.5.1a}$$

$$\delta(a)\leq d, \tag{5.5.1b}$$

for all $0\leq i\leq k$, the function ψ_i is non expansive, and (5.5.1c)

$$\bigcap_{0\leq i\leq k}\psi_i\circ\phi\circ\xi_i(x,y) \text{ is not empty.} \tag{5.5.1d}$$

Then ϕ is total.

Proof of Theorem 5.5.1. We show that $\mathrm{add}(\eta(r), \eta(s)) = \eta(r+s)$ for all $r, s \in \mathbb{R}$ in two steps:

1. $\mathrm{add}(\eta(r), \eta(s)) \sqsubseteq \eta(r+s)$ for all $r, s \in \mathbb{R}$. This is a direct consequence of the definition of add and Lemma 5.5.2, applied with $f_0 = \lambda xy.x + y$.

2. The function add is total. We show this as a consequence of Lemma 5.5.3, specifically: for each *total* elements x, y in R, there is an interval a in $\mathcal{R}_{\mathbb{Q}}$ and some *total* functions $\psi_0, \ldots, \psi_k \colon \mathsf{R} \to \mathsf{R}$ and $\xi_0, \ldots, \xi_k \colon \mathsf{R} \times \mathsf{R} \to \mathsf{R} \times \mathsf{R}$ satisfying the following conditions:

$$\mathrm{add}(x, y) \sqsupseteq \frac{1}{4} \mathrm{bound}_a \left(\bigcup_{0 \leq i \leq k} \psi_i \circ \mathrm{add} \circ \xi_i(x, y) \right), \tag{5.5.2}$$

$\delta(a) \leq 100$,

functions ψ_i are non expansive, for $0 \leq i \leq k$, and

$\bigcap_{0 \leq i \leq k} \psi_i \circ \mathrm{add} \circ \xi_i(x, y)$ is not empty.

We prove this property in two sub-steps:

2a. The property holds for all total elements x and y in R such that $-3 \leq x \leq 3$ and $-3 \leq y \leq 3$. We give some details of a proof by cases on whether $|x| < 2$ and $|y| < 2$ further below.

2b. For other total elements x and y, the element $\mathrm{add}(x, y)$ can be expressed as $u + \mathrm{add}(x + v, y + w)$ with $x + v$ and $y + w$ between -3 and 3, thus reducing this point to point 2a. More precisely, it can be shown by induction on m and n that, for all integers m, n such that $m, n \geq 3$, for all total elements $(x, y) \in \mathsf{R} \times \mathsf{R}$ such that $m < |x| \leq m + 1$ and $n < |y| \leq n + 1$, the following holds:

$$\mathrm{add}(x, y) = u_{x,y} + \mathrm{add}(x + v_{x,y}, y + w_{x,y}).$$

Where $u_{x,y}$, $v_{x,y}$ and $w_{x,y}$ are rational numbers such that $u_{x,y} + v_{x,y} + w_{x,y} = 0$ and $-3 \leq x + v_{x,y} \leq 3$ and $-3 \leq y + w_{x,y} \leq 3$. Then $\mathrm{add}(x + v_{x,y}, y + w_{x,y})$ is total, by point 2a. We omit further details as it is of no difficulty.

It remains to expand point 2*a* a little bit.
2a. Let x and y be total elements in R such that $-3 \leq x \leq 3$ and $-3 \leq y \leq 3$. We proceed by cases on x and y:

- Case $-2 < x < 2$ and $-2 < y < 2$. By definition of add, one has

$$\mathrm{add}(x,y) = \mathrm{bound}_{[-6,6]}\left(\frac{\mathrm{add}\,(4x,4y)}{4}\right) \tag{5.5.3}$$

$$= \frac{1}{4}\,\mathrm{bound}_{[-24,24]}\Big(\,\mathrm{add}\,(4x,4y)\Big). \tag{5.5.4}$$

  Take $a = [-24,24]$, $\psi_0 = \mathrm{id}_{\mathcal{R}}$ and $\xi_0 = \lambda(x,y).(4x,4y)$. It is easy to check that conditions 5.5.2 hold.

- Case $2 \le x \le 3$ and $-2 < y < 2$. An easy calculation using the definition of add gives

$$\mathrm{add}(x,y) = \frac{1}{4}\,\mathrm{bound}_{[-24,32]}\Bigg(\,\mathrm{bound}_{[-16,32]}\Big(8 + \mathrm{add}\,(4(x-2),4y)\Big) \cup \mathrm{bound}_{[-24,24]}\Big(\,\mathrm{add}\,(4x,4y)\Big)\Bigg). \tag{5.5.5}$$

  Hence we can take $a = [-24,32]$, $\psi_0 = \lambda z.\,\mathrm{bound}_{[-16,32]}\,(8+z)$, $\psi_1 = \mathrm{bound}_{[-24,24]}$, $\xi_0 = \lambda(x,y).(4(x-2),4y)$ and $\xi_1 = \lambda(x,y).(4x,4y)$. It follows from step 1 of the proof that $4x+4y$ is above both $\psi_0 \circ \mathrm{add} \circ \xi_0$ and $\psi_1 \circ \mathrm{add} \circ \xi_1$. It is easy to check that conditions 5.5.2 are satisfied by a, ψ_0, and ξ_0.

- Case $-2 < x < 2$ and $2 \le y \le 3$. A similar calculation gives

$$\mathrm{add}(x,y) = \frac{1}{4}\,\mathrm{bound}_{[-24,32]}\Bigg(\,\mathrm{bound}_{[-16,32]}\Big(8 + \mathrm{add}\,(4x,4(y-2))\Big) \cup \mathrm{bound}_{[-24,24]}\Big(\,\mathrm{add}\,(4x,4y)\Big)\Bigg). \tag{5.5.6}$$

  Hence we can take $a = [-24,32]$, $\psi_0 = \lambda z.\,\mathrm{bound}_{[-16,32]}\,(8+z)$, $\psi_1 = \mathrm{bound}_{[-24,24]}$, $\xi_0 = \lambda(x,y).(4x,4(y-2))$ and $\xi_1 = \lambda(x,y).(4x,4y)$ and proceed as in the previous case.

- Case $2 \le x \le 3$ and $2 \le y \le 3$. Now we have

$$\mathrm{add}(x,y) = \frac{1}{4}\,\mathrm{bound}_{[-24,32]}\Bigg(\,\mathrm{bound}_{[-16,32]}\Big(8 + \mathrm{add}\,(4(x-2),4y)\Big) \cup \mathrm{bound}_{[-16,32]}\Big(8 + \mathrm{add}\,(4x,4(y-2))\Big) \cup \mathrm{bound}_{[-24,24]}\Big(\,\mathrm{add}\,(4x,4y)\Big)\Bigg), \tag{5.5.7}$$

and we can then take $a = [-24, 32]$, $\psi_0 = \psi_1 = \lambda z.\, \mathrm{bound}_{[-16,32]}\,(8 + z)$, $\psi_2 = \mathrm{bound}_{[-24,24]}$, $\xi_0 = \lambda(x, y).(4(x - 2), 4y)$, $\xi_1 = \lambda(x, y).(4x, 4(y - 2))$ and $\xi_2 = \lambda(x, y).(4x, 4y)$ and again proceed as in the previous two cases.

- Other cases are treated similarly. These cases are:

  $-3 \leq x \leq -2$ and $-3 \leq y \leq -2$,

  $-2 < x < 2$ and $-3 \leq y \leq -2$,

  $2 \leq x \leq 3$ and $-3 \leq y \leq -2$,

  $-3 \leq x \leq -2$ and $-2 < y < 2$,

  $-3 \leq x \leq -2$ and $2 \leq y \leq 3$,

  $-2 < x < 2$ and $2 \leq y \leq 3$.

□

5.6 Representation of rational numbers and integers

To define operations on $\mathbb{Z}$ and $\mathbb{Q}$, we consider standard encodings of the integers and the rational numbers, but we never work with the encodings explicitly. We use the notation Z or Q in order to indicate the intended idea, but both Z and Q stand for N. For example, addition of an integer to a real is simply defined as an iterate of the basic operators (-1) and $(+1)$.

$$\begin{array}{l} \mathrm{add}_{\mathbb{Z}} \colon \mathsf{R} \times \mathsf{Z} \to \mathsf{R} \\ \mathrm{add}_{\mathbb{Z}}\,(x, n) = \mathrm{if}\, n = 0\, \mathrm{then}\, x\, \mathrm{else\, if}\;\; n > 0 \;\; \mathrm{then} \quad 1 + \mathrm{add}_{\mathbb{Z}}\,(x, n - 1) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathrm{else} \quad -1 + \mathrm{add}_{\mathbb{Z}}\,(x, n + 1) \end{array} \tag{5.6.1}$$

Multiplication of a real by an integer is performed using addition on real numbers defined in Section 5.5.

$$\begin{array}{l} \mathrm{mult}_{\mathbb{Z}} \colon \mathsf{R} \times \mathsf{Z} \to \mathsf{R} \\ \mathrm{mult}_{\mathbb{Z}}(x, n) = \mathit{cases} \;\; n = 0 \quad \to \quad 0 \\ \qquad\qquad\qquad\qquad\quad n < 0 \quad \to \quad \mathrm{mult}_{\mathbb{Z}}(x, -n) \\ \qquad\qquad\qquad\qquad\quad n = 1 \quad \to \quad x \\ \qquad\qquad\qquad\qquad\quad \mathrm{otherwise} \quad \to \quad x + \mathrm{mult}_{\mathbb{Z}}(x, n - 1) \end{array} \tag{5.6.2}$$

The cases function appearing in the definition above is simply a deterministic one (that can be expressed in terms of the conditional if-then-else), where the different conditions don't overlap and don't involve the function ttest.

5.7 Division of a real number by a rational number

We adapt the classical Euclidean algorithm to provide a representation of the division of a real number by an integer. The core function is $\mathrm{div}_{\mathbb{N}}$ that divides a non negative real by a natural number bigger than one, from which a representation of the division of any real by any non-zero rational number is easily derived (*cf.* Equation 5.7.5). The function $\mathrm{div}_{\mathbb{N}}$ is defined in terms of $\mathrm{ttest}_{m,m+1}$ and $\mathrm{bound}_{[m-1,m+1]}$, where m is a natural number parameter, which can be defined as follows:

$$\begin{aligned} &\mathrm{ttest}\colon \mathsf{N} \to \mathsf{R} \to \mathsf{R} \\ &\mathrm{ttest}_{m,m+1}(x) = \mathrm{ttest}_{0,1}\,(x-m) \qquad (5.7.1) \\ &\mathrm{bound}\colon \mathsf{N} \to \mathsf{R} \to \mathsf{R} \\ &\mathrm{bound}_{[m-1,m+1]}(x) = m + \mathrm{bound}_{[-1,1]}\,(x-m)\,. \qquad (5.7.2) \end{aligned}$$

Theorem 5.7.1. *For any non-negative real number x and any natural number $n > 1$,*

$$\mathrm{div}_{\mathbb{N}}\left(\eta_{\mathbb{R}}\,(x)\,,\eta_{\mathbb{N}}\,(n)\,,0,0\right) = \eta_{\mathbb{R}}\left(\frac{x}{n}\right) \qquad (5.7.3)$$

where $\mathrm{div}_{\mathbb{N}}$ *is the function*

$$\begin{aligned} &\mathrm{div}_{\mathbb{N}}\colon \mathsf{R}\times\mathsf{N}\times\mathsf{N}\times\mathsf{N}\to\mathsf{R} \qquad (5.7.4) \\ &\mathrm{div}_{\mathbb{N}}(x,n,m,i) = \ \text{cases}\ \ mn+i \le x \ \to\ \mathrm{div}_{\mathbb{N}}\,(x,n,m',i') \\ &\qquad\qquad\qquad\qquad x \le mn+i+1 \ \to\ \mathrm{bound}_{[m-1,m+1]}\left(\tfrac{1}{4}\times\mathrm{div}_{\mathbb{N}}\,(4x,n,0,0)\right) \\ &\qquad \textit{where} \\ &\qquad i' = \text{if } i = n-1 \text{ then } 0 \text{ else } i+1 \\ &\qquad m' = \text{if } i = n-1 \text{ then } m+1 \text{ else } m. \end{aligned}$$

The function $\mathrm{div}_{\mathbb{N}}$ is based on the following idea. Given a non-negative real number x and a natural number $n > 1$, there exists a unique triple $(m_0, i_0, \varepsilon_0) \in \mathbb{N}\times\mathbb{N}\times\mathbb{R}$ such that $i_0 \le (n-1)$, $\varepsilon_0 \in [0,1)$ and $r = m_0 n + i_0 + \varepsilon_0$. The two natural numbers m_0 and i_0 are respectively the quotient and the remainder of the Euclidean division of the integer part of x by n. Furthermore, the real number $\frac{x}{n}$ is in the interval $[m_0, m_0+1)$. Seen operationally, the function $\mathrm{div}_{\mathbb{N}}$ works by scanning all pairs (m,i) with $i \le n-1$, starting from $(0,0)$, until $mn+i \le x \le mn+i+1$. Pairs (m,i) must be scanned in an appropriate order: the next numbers i' and m' to be considered satisfy $m'n + i' = mn + i + 1$.

With $\mathrm{div}_{\mathbb{N}}$ at hand, division of *any* real number by *any* non-zero integer is then

represented by

$$\begin{aligned}
&\mathrm{div}_{\mathbb{Z}} \colon \mathsf{R} \times \mathsf{Z} \to \mathsf{R} \\
&\mathrm{div}_{\mathbb{Z}}(x,n) = \text{if } n = 1 \text{ then } x \text{ else if } n \leq 0 \text{ then } \mathrm{div}_{\mathbb{Z}}(-x,-n) \text{ else } \phi_1(x,n) \qquad (5.7.5)\\
&\text{where} \\
&\phi_1 \colon \mathsf{R} \times \mathsf{N} \to \mathsf{R} \\
&\phi_1(x,n) = \text{cases } \begin{array}{lcl} x \leq 0 & \to & -\mathrm{div}_{\mathbb{N}}(-x,n,0,0) \\ -1 \leq x & \to & \text{cases } \begin{array}{lcl} x \leq 1 & \to & \mathrm{bound}_{[-1,1]}\left(\frac{1}{4} \times \phi_1(4x,n)\right) \\ 0 \leq x & \to & \mathrm{div}_{\mathbb{N}}(x,n,0,0) \end{array} \end{array} \\
&\mathrm{div}_{\mathbb{Q}} \colon \mathsf{R} \times \mathsf{Q} \to \mathsf{R} \\
&\mathrm{div}_{\mathbb{Q}}(x,q) = \mathrm{numerator}(q) \times \mathrm{div}_{\mathbb{Z}}\left(x, \mathrm{denominator}(q)\right). \qquad (5.7.6)
\end{aligned}$$

For any $q \in \mathbb{Q}$, the functions that return the numerator and denominator of the irreducible fraction with positive denominator that is equal to q are recursive and hence definable in the language as all first-order functions on natural numbers are definable in PCF.

Proof of Theorem 5.7.1. We simplify our notation by identifying numbers with their representations in the model. Let us fix $n > 1$ and define $\phi(x) = \mathrm{div}_{\mathbb{N}}(x,n,0,0)$. We want to show that $\phi(x) = \frac{x}{n}$ for all real numbers x. The outline of the proof is similar to the one for the identity, so we only give details for the specific difficulties of the present case.

1. We will prove in part 4 that for all real x,

$$\phi(x) = \bigcup_{0 \leq j \leq k} \mathrm{bound}_{a_j}\left(\frac{1}{4}\phi(4x)\right) \qquad (5.7.7)$$

for some intervals a_j with length 2 and $\eta\left(a_j\right) \sqsubseteq \frac{x}{n}$. By defining f_0 as the maximal extension of the function $\mathbb{R} \to \mathbb{R}, x \mapsto \frac{x}{n}$, we can deduce from Lemma 5.4.2 that ϕ is below f_0 on total elements:

$$\phi(x) \sqsubseteq \frac{x}{n} \text{ for all total elements } x. \qquad (5.7.8)$$

2. From Lemma 5.4.3 on page 70 and assertions 5.7.7 and 5.7.8, it follows that ϕ is total, by similar arguments as the ones used in the proof for the identity.

3. As a consequence of parts 1 and 2, ϕ represents the the function $\mathbb{R} \to \mathbb{R}, x \mapsto \frac{x}{n}$.

4. We now show that, for each real x, there exist some intervals $a_0, \ldots, a_k$ of length 2 such that

$$\phi(x) = \bigcup_{0 \leq j \leq k} \mathrm{bound}_{a_j}\left(\frac{1}{4}\phi(4x)\right) \qquad (5.7.9)$$

and $\eta\,(a_j) \sqsubseteq \frac{x}{n}$.

For the case where $x = 0$, it is easy to check that

$$\begin{aligned}\mathrm{div}_{\mathbb{N}}(0,n,0,0) &= \mathrm{div}_{\mathbb{N}}(0,n,1,0) \cup \mathrm{bound}_{[-1,1]}\left(\frac{1}{4}\,\mathrm{div}_{\mathbb{N}}(0,n,0,0)\right) \\ &= \mathrm{bound}_{[0,2]}\left(\frac{1}{4}\,\mathrm{div}_{\mathbb{N}}(0,n,0,0)\right) \cup \mathrm{bound}_{[-1,1]}\left(\frac{1}{4}\,\mathrm{div}_{\mathbb{N}}(0,n,0,0)\right)\end{aligned} \tag{5.7.10}$$

and $\frac{0}{n}$ belongs to both $[0,2]$ and $[-1,1]$.

Let now x be a *positive* real number. By expressing the Euclidean division of the integer part $E(x)$ of x by n as $E(x) = m_0 n + i_0$, one easily shows that there exists a unique triple $(m_0, i_0, \varepsilon_0) \in \mathbb{N} \times \mathbb{N} \times \mathbb{R}$ such that $i_0 \leq (n-1)$, $\varepsilon_0 \in [0,1)$ and $x = m_0 n + i_0 + \varepsilon_0$. Furthermore, the real number $\frac{x}{n}$ is in $[m_0, m_0 + 1)$.

We order the set of pairs $P_n = \{(m,i) \mid m, i \in \mathbb{N}, 0 \leq i \leq n-1\}$ by stating that

$$(l,j) < (m,i) \text{ if and only if } ln + j < mn + i. \tag{5.7.11}$$

This gives a discrete, total order over P_n and the successor of (m,i) is given by (m',i') as in the definition of $\mathrm{div}_{\mathbb{N}}$:

$$(m',i') = \begin{cases}(m, i+1) & \text{if } i < n-1 \\ (m+1, 0) & \text{otherwise.}\end{cases} \tag{5.7.12}$$

Furthermore $m'n + i' = mn + i + 1$ for all pairs (m,i) in P_n.

Let then (m_1, i_1) be the smallest pair such that $x \leq m_1 n + i_1 + 1$, *i.e.* such that the second condition in the definition of $\mathrm{div}_{\mathbb{N}}$ is satisfied. It is not difficult to see that $\mathrm{div}_{\mathbb{N}}(x,n,0,0) = \mathrm{div}_{\mathbb{N}}(x,n,m_1,i_1)$, so it remains to prove that $\mathrm{div}_{\mathbb{N}}\,(x,n,m_1,i_1) = \frac{x}{n}$. We proceed according to whether x is an integer or not.

Let us first suppose that x is a boundary of cases, which is equivalent to saying that $\varepsilon_0 = 0$. Then we have that

$$(m_0 n + i_0 - 1) + 1 = m_0 n + i_0 = x \tag{5.7.13}$$

and hence $(m_0, i_0) = (m_1', i_1')$, *i.e.* (m_1, i_1) is the predecessor of (m_0, i_0) and

$$x = m_1 n + i_1 + 1 = m_0 n + i_0. \tag{5.7.14}$$

Thus, by definition of $\mathrm{div}_{\mathbb{N}}$,

$$\begin{aligned}
\mathrm{div}_{\mathbb{N}}(x,n,m_1,i_1) = {} & \mathrm{div}_{\mathbb{N}}(x,n,m_1',i_1') && (5.7.15)\\
& \cup\big(\,\mathrm{bound}_{[m_1-1,m_1+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)\\
= {} & \mathrm{div}_{\mathbb{N}}(x,n,m_0,i_0) && (5.7.16)\\
& \cup\big(\,\mathrm{bound}_{[m_1-1,m_1+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)\\
= {} & \mathrm{div}_{\mathbb{N}}(x,n,m_0',i_0') && (5.7.17)\\
& \cup\big(\,\mathrm{bound}_{[m_0-1,m_0+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)\\
& \cup\big(\,\mathrm{bound}_{[m_1-1,m_1+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)\\
= {} & \big(\,\mathrm{bound}_{[m_0'-1,m_0'+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big) && (5.7.18)\\
& \cup\big(\,\mathrm{bound}_{[m_0-1,m_0+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)\\
& \cup\big(\,\mathrm{bound}_{[m_1-1,m_1+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)
\end{aligned}$$

and $\frac{x}{n}$ belongs to the three intervals $[m_0'-1,m_0'+1]$, $[m_0-1,m_0+1]$ and $[m_1'-1,m_1'+1]$ because $x = m_1n+i_1+1 = m_0n+i_0 = m_0'n+i_0'-1$.

If x is not an integer, then $\varepsilon_0 > 0$ and the following holds:

$$m_0n+i_0 < m_0n+i_0+\varepsilon_0 = x < m_0n+i_0+1 = m_0'n+i_0'. \tag{5.7.19}$$

In particular, it is easy to see that $(m_0,i_0) = (m_1,i_1)$. So, by definition of $\mathrm{div}_{\mathbb{N}}$,

$$\begin{aligned}
\mathrm{div}_{\mathbb{N}}(x,n,m_1,i_1) = {} & \mathrm{div}_{\mathbb{N}}(x,n,m_0',i_0') && (5.7.20)\\
& \cup\big(\,\mathrm{bound}_{[m_0-1,m_0+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)\\
= {} & \big(\,\mathrm{bound}_{[m_0'-1,m_0'+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big) && (5.7.21)\\
& \cup\big(\,\mathrm{bound}_{[m_0-1,m_0+1]}\big(\tfrac{1}{4}\,\mathrm{div}_{\mathbb{N}}(4x,n,0,0)\big)\,\big)
\end{aligned}$$

and $\frac{x}{n}$ belongs to the two intervals $[m_0'-1,m_0'+1]$ and $[m_0-1,m_0+1]$ because $x = m_0n+i_0 = m_0'n+i_0'-1$.

□

5.8 Parametrized basic operators

We can now easily define parametrized versions bound', ttest', $+'$ and $\times'$ of the functions $\mathrm{bound}_{[p,q]}$, $\mathrm{ttest}_{p,q}$, $p+$ and $p\times$ where p and q are taken as rational parameters.

$$(\times') \colon \mathsf{Q} \times \mathsf{R} \to \mathsf{R}$$
$$q \times' x = \mathrm{div}_{\mathbb{Z}} \left(\mathrm{mult}_{\mathbb{Z}} \left(x, \mathrm{numerator}(q) \right), \mathrm{denominator} \left(q \right) \right) \tag{5.8.1}$$

$$(+') \colon \mathsf{Q} \times \mathsf{R} \to \mathsf{R}$$
$$q +' x = \mathrm{div}_{\mathbb{Z}} \left(\mathrm{add}_{\mathbb{Z}} \left(\mathrm{mult}_{\mathbb{Z}} \left(\mathrm{denominator}(q), x \right), \mathrm{numerator}(q) \right), \mathrm{denominator}(q) \right) \tag{5.8.2}$$

$$\mathrm{bound}' \colon \mathsf{Q} \times \mathsf{Q} \times \mathsf{R} \to \mathsf{R}$$
$$\mathrm{bound}'_{[p,q]}(x) = p +' (q - p) \times' \mathrm{bound}_{[0,1]} \left(\frac{(-p) +' x}{q - p} \right) \tag{5.8.3}$$

$$\mathrm{ttest}' \colon \mathsf{Q} \times \mathsf{Q} \times \mathsf{R} \to \mathsf{R}$$
$$\mathrm{ttest}'_{[p,q]}(x) = \mathrm{ttest}_{[0,1]} \left(\frac{(-p) +' x}{q - p} \right) \tag{5.8.4}$$

We remark that, for each p and q, the functions $\mathrm{bound}'_{[p,q]}$, $\mathrm{ttest}'_{p,q}$, $p+'$ and $p\times'$ are not maximal whereas the denotations of the corresponding language constants are. However, as long as the distinction is not important, we will use the same notation and drop the prime superscript on the parametrized versions.

5.9 Realizers for real numbers

The following definition is reminiscent of the characterization of real numbers via Dedekind cuts and will allow us to find representations of the square root function in a straightforward way, in Section 5.10.

Definition 5.9.1. Let $\alpha \colon \mathsf{Q} \times \mathsf{Q} \to \mathsf{B}$ be a multi-valued continuous function, that is $\alpha(p,q) \sqsupseteq \eta\,(true) \cup \eta\,(false)$ for all total elements p and q. We say that α realizes a total element $x \in \mathsf{R}$, and write $\alpha \Vdash x$, if for all $(p,q) \in \mathbb{Q} \times \mathbb{Q}$ such that $p < q$,

$$\begin{aligned} \alpha(p,q) \sqsubseteq \eta\,(\mathrm{true}) &\implies x \leq q, \\ \alpha(p,q) \sqsubseteq \eta\,(\mathrm{false}) &\implies p \leq x. \end{aligned} \tag{5.9.1}$$

(One peculiarity that distinguishes realizers from Dedekind cuts is the use of inequalities $x \leq q$ and $p \leq x$ rather than strict ones, $x < q$ and $p < x$.)

For example, given $x \in \mathsf{R}$ total, the function $\tilde{x} \colon \mathsf{Q} \times \mathsf{Q} \to \mathsf{B}$ defined by

$$\tilde{x}(p,q) = \mathrm{ttest}'_{p,q}(x)$$

is a realizer of x, where the function $\mathrm{ttest}'_{p,q}$ is the parametrized one defined in Section 5.8. A realizer for x can be seen as a black box which tells which tests x passes. Given p and q it indicates at least one test that x passes among $p \leq x$ and $x \leq q$. If x satisfies both tests $p \leq x$ and $x \leq q$, the above specification of a realizer for x only stipulates that it should indicate one of them, possibly both, but not necessarily both. In other words, if $\alpha \Vdash x$ and $p < q$ then

$$x < p \implies \alpha(p,q) = \eta(\mathrm{true}) \tag{5.9.2}$$

$$x > q \implies \alpha(p,q) = \eta(\mathrm{false}) \tag{5.9.3}$$

$$p \leq x \leq q \implies \alpha(p,q) \in \{\eta(\mathrm{true}), \eta(\mathrm{false}), \eta(\mathrm{true}) \cup \eta(\mathrm{false})\}\,. \tag{5.9.4}$$

It is easy to see that a realizer realizes a unique total element. Suppose α realizes two distinct total elements x and y, with $x < y$ and choose p and q strictly between x and y. Then $\alpha(p,q) = \eta\,(\mathrm{true})$ because $\alpha \Vdash x$, which contradicts $\alpha \Vdash y$.

Let us call $\widetilde{\mathcal{R}}$ the set of continuous functions from $\mathsf{Q} \times \mathsf{Q}$ to B. With this notation, we have

$$\begin{aligned} \widetilde{\cdot}\colon \mathsf{R} &\longrightarrow \widetilde{\mathcal{R}} \\ x &\longmapsto \lambda pq.\, \mathrm{ttest}'_{p,q}(x), \end{aligned} \tag{5.9.5}$$

which provides a realizer of any total element $x \in \mathsf{R}$. In the other direction, the function ζ below allows one to retrieve a real number from any of its realizers. Notice the parallel with the identity function given in section 5.4.

$$\zeta\colon \widetilde{\mathcal{R}} \to \mathsf{R} \tag{5.9.6}$$

$$\begin{array}{l} \zeta(\alpha) = \text{cases} \\ \qquad \begin{array}{lcl} \alpha\,(-3,-2) = \mathrm{true} & \to & -2 + \zeta(\alpha \mathbin{\widetilde{+}} 2) \\ \alpha\,(-3,-2) = \mathrm{false} & \to & \text{cases} \\ \qquad\qquad \begin{array}{lcl} \alpha\,(2,3) = \mathrm{false} & \to & 2 + \zeta(\alpha \mathbin{\widetilde{-}} 2) \\ \alpha\,(2,3) = \mathrm{true} & \to & \mathrm{bound}_{[-3,3]}\left(\frac{1}{4}\left(\zeta(4 \mathbin{\widetilde{\times}} \alpha)\right)\right) \end{array} \end{array} \end{array}$$

where

$\alpha \mathbin{\widetilde{+}} 2 = \lambda(p,q).\alpha(p-2, q-2)$,

$\alpha \mathbin{\widetilde{-}} 2 = \lambda(p,q).\alpha(p+2, q+2)$ and

$4 \mathbin{\widetilde{\times}} \alpha = \lambda(p,q).\alpha(\frac{1}{4} \times p, \frac{1}{4} \times q)$.

Theorem 5.9.1. *For any total element x in R and any $\alpha \in \widetilde{\mathcal{R}}$, if $\alpha \Vdash x$ then $\zeta(\alpha) = x$.*

Proof. For any total element x and realizer α such that $\alpha \Vdash x$, it is easy to see that

$$\begin{aligned} &\alpha \mathbin{\widetilde{+}} 2 \Vdash x + 2, \\ &\alpha \mathbin{\widetilde{-}} 2 \Vdash x - 2 \text{ and} \\ &4 \mathbin{\widetilde{\times}} \alpha \Vdash 4 \times x. \end{aligned} \tag{5.9.7}$$

By definition, the function ζ is the least fixed point of F defined by

$$F(f)(\alpha) = \begin{array}{lll} \text{if } \alpha(-3,2) \text{ then} & -2 + f\left(\alpha \mathbin{\widetilde{+}} 2\right) & \\ \text{else if } \alpha(2,3) \text{ then} & 2 + f\left(\alpha \mathbin{\widetilde{-}} 2\right) & \\ \text{else} & \text{bound}_{[-3,3]}\left(\frac{1}{4}\left(f\left(4 \mathbin{\widetilde{\times}} \alpha\right)\right)\right) & \end{array} \tag{5.9.8}$$

By definition of $\Vdash$, it holds that, whenever $\alpha \Vdash x$,

$$\alpha(p,q) \sqsupseteq \text{ttest}_{p,q}(x), \tag{5.9.9}$$

So by monotonicity and Equality 5.9.8 we obtain that, for any total x and any realizer α of x,

$$F(f)(\alpha) \sqsupseteq \begin{array}{lll} \text{if } \text{ttest}_{-3,-2}(x) \text{ then} & -2 + f\left(\alpha \mathbin{\widetilde{+}} 2\right) & \\ \text{else if } \text{ttest}_{2,3}(x) \text{ then} & 2 + f\left(\alpha \mathbin{\widetilde{-}} 2\right) & \\ \text{else} & \text{bound}_{[-3,3]}\left(\frac{1}{4}\left(f\left(4 \mathbin{\widetilde{\times}} \alpha\right)\right)\right). & \end{array} \tag{5.9.10}$$

It can be shown from this, in the same way as we did for the identity function, that for all total realizers α, the following holds for the particular case $m = 1$:

$$\begin{array}{l} \exists n_\alpha \in \mathbb{N}, \exists k \in \mathbb{N}, \exists a_0, \ldots, a_k, \exists \beta_0, \ldots, \beta_k, \exists r_0, \ldots, r_k \in \mathbb{Q}, \text{ such that} \\ F^{n_\alpha}(f)(\alpha) \sqsupseteq \dfrac{1}{4^m} \bigcup_{0 \le i \le k} \text{bound}_{a_i}\left(r_i + f\left(\beta_i\right)\right) \\ \text{where for each } i \text{ between 0 and } k, \\ \text{the length of } a_i \text{ is less than 30,} \\ \eta\left(a_i\right) \sqsubseteq 4^m x, \text{ where } x \in \mathsf{R} \text{ is the unique total element such that } \alpha \Vdash x, \\ \beta_i \text{ is a realizer and} \\ r_i + y_i = 4^m x \text{ where } y_i \in \mathsf{R} \text{ is the unique total element such that } \beta_i \Vdash y_i. \end{array} \tag{5.9.11}$$

Let us refer to Property 5.9.11 as $P(\alpha, m)$. We have just said that $P(\alpha, 1)$ holds for all realizers α. We now show by induction on m that $P(\alpha, m)$ holds for all integer $m \ge 1$ and all realizers α, from which Theorem 5.9.1 easily follows.

Suppose as induction hypothesis that $P(\alpha, m)$ is satisfied for all realizers α and some $m \ge 1$. Let α be a realizer and let us show that $P(\alpha, m+1)$. By induction hypothesis, $P(\alpha, m)$ holds, so let $n_\alpha \in \mathbb{N}$, $k \in \mathbb{N}$, $a_0, \ldots, a_k \in \mathcal{R}_\mathbb{Q}$, $\beta_0, \ldots, \beta_k$ and $r_0, \ldots, r_k \in \mathbb{Q}$ be

such that

$$F^{n_\alpha}(f)(\alpha) \sqsupseteq \frac{1}{4^m} \bigcup_{0\le i\le k} \mathrm{bound}_{a_i}\left(r_i + f\left(\beta_i\right)\right) \tag{5.9.12}$$

where for each i, $0 \le i \le k$,
the length of a_i is less than 30,
$\eta\left(a_i\right) \sqsubseteq 4^m x,$ where $x \in \mathsf{R}$ is the unique total element such that $\alpha \Vdash x$,
β_i is a realizer and
$r_i + y_i = 4^m x$ where $y_i \in \mathsf{R}$ is the unique total element such that $\beta_i \Vdash y_i$.

For each β_i, the property $P(\beta_i, 1)$ is satisfied. Let then, for each i between 0 and k, choose $n_i \in \mathbb{N}$, $l_i \in \mathbb{N}$, $b_{i,0}, \ldots, b_{i,l}$, $\gamma_{i,0}, \ldots, \gamma_{i,l_i}$, $s_{i,0}, \ldots, s_{i,l_i} \in \mathbb{Q}$, such that

$$F^{n_i}(f)(\beta_i) \sqsupseteq \frac{1}{4} \bigcup_{0\le j\le l_i} \mathrm{bound}_{b_{i,j}}\left(s_{i,j} + f\left(\gamma_{i,j}\right)\right) \tag{5.9.13}$$

where for each j, $0 \le j \le l_i$,
the length of $b_{i,j}$ is less than 30,
$\eta\left(b_{i,j}\right) \sqsubseteq 4y_i$,
$\gamma_{i,j}$ is a realizer,
$s_{i,j} + z_{i,j} = 4y_i,$ where $z_{i,j} \in \mathsf{R}$ is the unique total element such that $\gamma_{i,j} \Vdash z_{i,j}$.

Let n be the greatest integer among $n_0, \ldots, n_k$. Then $F^{n_\alpha+n}(f)(\alpha)$, which is equal to $F^{n_\alpha}\left(F^n(f)\right)(\alpha)$, satisfies

$$F^{n_\alpha+n}(f)(\alpha) \sqsupseteq \frac{1}{4^m} \bigcup_{0\le i\le k} \mathrm{bound}_{a_i}\left(r_i + \left(\frac{1}{4} \bigcup_{0\le j\le l_i} \mathrm{bound}_{b_{i,j}}\left(s_{i,j} + f(\gamma_{i,j})\right)\right)\right), \tag{5.9.14}$$

from which we obtain

$$F^{n_\alpha+n}(f)(\alpha) \sqsupseteq \frac{1}{4^{m+1}} \bigcup_{0\le i\le k} \bigcup_{0\le j\le l_i} \mathrm{bound}_{4a_i}\left(\mathrm{bound}_{4r_i+b_{i,j}}\left(4r_i + s_{i,j} + f(\gamma_{i,j})\right)\right) \tag{5.9.15}$$

and

$$F^{n_\alpha+n}(f)(\alpha) \sqsupseteq \frac{1}{4^{m+1}} \bigcup_{\substack{0\le i\le k\\ 0\le j\le l_i}} \mathrm{bound}_{c_{i,j}}\left(t_{i,j} + f(\gamma_{i,j})\right) \tag{5.9.16}$$

where for all i such that $0 \leq i \leq k$ and all j such that $0 \leq j \leq l_i$,

$$c_{i,j} \text{ is by } c_{i,j} = \text{bound}_{4a_i}\left(4r_i + b_{i,j}\right), \tag{5.9.17}$$
$$c_{i,j} \text{ has length less than 30 because } b_{i,j} \text{ has,} \tag{5.9.18}$$
$$\eta\left(c_{i,j}\right) \sqsubseteq 4^{m+1}x \text{ because } \eta\left(a_i\right) \sqsubseteq 4^m x \text{ and } \eta\left(4r_i + b_{i,j}\right) \sqsubseteq 4\left(r_i + y_i\right) = 4^{m+1}x,$$
$$\gamma_{i,j} \Vdash z_{i,j},$$
$$t_{i,j} \text{ is defined to be } 4r_i + s_{i,j} \text{ and satisfies}$$
$$t_{i,j} + z_{i,j} = 4^{m+1}x \text{ because } 4r_i + s_{i,j} + z_{i,j} = 4r_i + 4y_i = 4(4^m x),$$

which shows that $P(\alpha, m+1)$ is satisfied. □

5.10 Square root

For $p, q, x \geq 0$, one has that $p \leq x \leq q$ if and only if $p^2 \leq \sqrt{x} \leq q^2$, which allows to easily find a realizer for $\sqrt{x}$:

$$\begin{aligned} &\text{sqrt} : \mathsf{R} \to \mathsf{R} \\ &\text{sqrt}\left(x\right) = \zeta\left(\lambda\left(p,q\right).\,\text{if } p \leq 0 \,\text{then false else ttest}'_{p^2,q^2}(x)\right) \end{aligned} \tag{5.10.1}$$

where ttest′ is the parametrized version of ttest given in section 5.8 on page 81. As an application of Theorem 5.9.1 of Section 5.9, we obtain the following theorem.

Theorem 5.10.1. *For any non-negative real number r,*

$$\text{sqrt}\left(\eta\left(r\right)\right) = \eta\left(\sqrt{r}\right). \tag{5.10.2}$$

5.11 Embedding the rational numbers into the real numbers

We have defined the function $\lambda x.\tilde{x}$ which sends a total element $x \in \mathsf{R}$ to a particular realizer of x in $\widetilde{\mathcal{R}}$, namely $\lambda pq.\,\text{ttest}'_{p,q}\left(x\right)$. Based on the same idea, we define

$$\begin{aligned} &\widetilde{\iota_{\mathbb{Q},\mathbb{R}}} : \mathsf{Q} \to \widetilde{\mathcal{R}} \\ &\widetilde{\iota_{\mathbb{Q},\mathbb{R}}}(s) = \lambda pq.\,\text{if } \; s \leq q \;\text{ then } \; \text{true} \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{else } \; \text{if } \; p \leq s \;\text{ then } \; \text{false} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{else } \; \text{true}\,. \end{aligned} \tag{5.11.1}$$

As an immediate consequence of Theorem 5.9.1, we obtain

Theorem 5.11.1. *For any rational number s, it holds that*

$$\zeta\Big(\widetilde{\iota_{\mathbb{Q},\mathbb{R}}}\left(\eta_{\mathbb{Q}}\left(s\right)\right)\Big) = \eta_{\mathbb{R}}\left(s\right)$$

where $\eta_{\mathbb{Q}}$ is the embedding that sends any code $n \in \mathbb{N}$ for q to the corresponding total element in N. (Recall our convention to write Q instead N when the intention is to work with codes of rational numbers).

One can similarly define the function

$$\widetilde{\iota_{\mathbb{N},\mathbb{R}}} \colon \mathsf{N} \to \widetilde{\mathcal{R}} \tag{5.11.2}$$

$$\widetilde{\iota_{\mathbb{N},\mathbb{R}}}(n) = n \leq q \text{ or } n \leq p \tag{5.11.3}$$

that sends a representation of any natural number to a corresponding realizer and then obtain

Theorem 5.11.2. *For any natural number n, it holds that*

$$\zeta\Big(\widetilde{\iota_{\mathbb{N},\mathbb{R}}}\left(\eta_{\mathbb{N}}\left(n\right)\right)\Big) = \eta_{\mathbb{R}}\left(n\right).$$

Definition 5.11.1. We use the following notation:

$$\iota_{\mathbb{Q},\mathbb{R}} \colon \mathsf{Q} \to \mathsf{R} \tag{5.11.4}$$
$$\iota_{\mathbb{Q},\mathbb{R}} = \zeta \circ \widetilde{\iota_{\mathbb{Q},\mathbb{R}}}$$

and

$$\iota_{\mathbb{N},\mathbb{R}} \colon \mathsf{N} \to \mathsf{R} \tag{5.11.5}$$
$$\iota_{\mathbb{N},\mathbb{R}} = \zeta \circ \widetilde{\iota_{\mathbb{N},\mathbb{R}}}\,.$$

5.12 Partial $x <_{\mathrm{p}} 0$ test

It is possible to define the partial test $x <_{\mathrm{p}} 0$ on real numbers which is undefined for $x = 0$.

Lemma 5.12.1. *Define*

$$\cdot <_p 0 \colon \mathsf{R} \to \mathsf{B} \tag{5.12.1}$$

$$\begin{array}{lllllll} x <_p 0 = \text{cases} & x \leq 0 & \to & \text{true} & & & \\ & -1 \leq x & \to & \text{cases} & 0 \leq x & \to & \text{false} \\ & & & & x \leq 1 & \to & (2x) <_p 0. \end{array}$$

Then

$$\begin{aligned} \eta(0) <_p 0 &= \bot, \\ x <_p 0 &= \{\text{true}\} \ \textit{if } x < 0 \textit{ and} \\ x <_p 0 &= \{\text{false}\} \ \textit{if } 0 < x. \end{aligned} \tag{5.12.2}$$

Proof. We have that $\eta\,(0) <_{\mathrm{p}} 0 = \{\text{true}\} \cup \{\text{false}\} \cup \{\eta\,(0) <_{\mathrm{p}} 0\}$, so $\eta\,(0) <_{\mathrm{p}} 0$ is the least fixed point of $\lambda b.\, \{\text{false}, \text{true}, b\}$, that is $\{\text{true}, \text{false}, \bot_{\mathbb{B}_\bot}\} = \bot$.

If $x < 0$ then $2^n x < 0$ and it is easy to see that $x <_{\mathrm{p}} 0 = \{\text{true}, 2^n x <_{\mathrm{p}} 0\}$, for all $n \in \mathbb{N}$. For some n big enough, $2^n x <_{\mathrm{p}} 0 = \text{true}$. Hence $x <_{\mathrm{p}} 0 = \{\text{true}\}$. Similarly for $x > 0$. □

5.13 Parametric bound functions

We have already defined a parametric function $\text{bound}' \colon \mathsf{Q} \times \mathsf{Q} \times \mathsf{R} \to \mathsf{R}$ which takes rational parameters and represents the function $\text{bound} \colon \mathbb{Q} \times \mathbb{Q} \times \mathbb{R} \to \mathbb{R}$ (Section 5.8, p. 81). However, the behaviour of bound' at partial elements is difficult to describe. We now define another operator $\text{bound}'' \colon \mathsf{R} \times \mathsf{R} \times \mathsf{R} \to \mathsf{R}$ with the following properties.

1. It accepts real parameters rather than rational ones.
2. For all real intervals $[a, b]$, the function $\text{bound}''(\eta(a), \eta(b), -) \colon \mathsf{R} \to \mathsf{R}$ represents a function which agrees at $[a, b]$ with the function $\text{bound}_{[a,b]} \colon \mathbb{R} \to \mathbb{R}$, and,
3. even if the third argument is undefined, $\text{bound}''\,(\eta\,(a)\,, \eta\,(b)\,, \bot)$ provides an approximation of the interval $[a, b]$ which is above the interval $[a - 3(b - a), b + 3(b - a)]$.

The bound'' operator will be used in recursive definitions to ensure that some information is outputted at each unfolding. For example, we will use it to define operators that find the root of certain functions (Section 5.15), compute the limit of a sequence (Section 5.17) and, in the next chapter, to prove that all computable, first-order total functions are

definable. We define the function bound″ as follows.

$$\text{bound}'' \colon \mathsf{R} \times \mathsf{R} \times \mathsf{R} \to \mathsf{R} \tag{5.13.1}$$

$$\begin{aligned}
\text{bound}''(a,b,x) = \text{cases}\ \ & \tfrac{3}{4} \le c \ \ \to \ \ 1 + \text{bound}''(a-1, b-1, x-1) \\
& c \le 1 \ \ \to \ \ \text{cases}\ \ c \le -\tfrac{3}{4} \ \ \to \ \ -1 + \text{bound}''(a+1, b+1, x+1) \\
& \qquad\qquad -1 \le c \ \ \to \ \ \phi(a,b,x)
\end{aligned}$$

$$\begin{aligned}
\phi(a,b,x) = \text{cases}\ \ & r \le 2.1 \ \ \to \ \ \text{bound}_{[-6,6]}\left(\tfrac{1}{2}\,\text{bound}''(2a, 2b, 2x)\right) \\
& 2 \le r \ \ \to \ \ \psi(a,b,x)
\end{aligned}$$

$$\begin{aligned}
\psi(a,b,x) = \text{cases}\ \ & 4.9 \le r \ \ \to \ \ 2 \times \psi\left(\tfrac{1}{2}a, \tfrac{1}{2}b, \tfrac{1}{2}x\right) \\
& r \le 5 \ \ \to \ \ \text{bound}_{[-6,6]}(x)
\end{aligned}$$

where

$$c = \frac{a+b}{2} \tag{5.13.2}$$

$$r = \frac{b-a}{2} \tag{5.13.3}$$

Lemma 5.13.1. *For all real numbers a and b such that $a \le b$ and all $x \in \mathsf{R}$, there exists finitely many intervals $[c_0, d_0], \ldots, [c_K, d_K]$ such that*

$$\text{bound}''(\eta(a), \eta(b), x) = \bigcup_{0 \le k \le K} \text{bound}_{[c_k, d_k]}(x) \tag{5.13.4}$$

and

$$[a - 3(b-a), b + 3(b-a)] \sqsubseteq [c_k, d_k] \sqsubseteq [a, b] \tag{5.13.5}$$

for each k.

In particular, if $r \in [a, b]$, one has that

$$\text{bound}''(\eta(a), \eta(b), \eta(r)) = \eta(r) = \eta\left(\text{bound}_{[a,b]}(r)\right). \tag{5.13.6}$$

Proof. We identify real numbers with their representations in R. In particular, we write a for both the real number a and the element $\eta(a)$ of R, and similarly for b. Let $h \colon \mathbb{R} \to \mathbb{R}$ be a linear map; we use the same name for the continuous function $h \colon \mathsf{R} \to \mathsf{R}$ arising from

the Smyth power construction. A continuous function $h\colon \mathsf{R} \to \mathsf{R}$ is said to be a strictly increasing linear function if it arises from a strictly increasing linear map $h\colon \mathbb{R} \to \mathbb{R}$.

The proof consists in showing that the function bound$''$ satisfies

$$\text{bound}''(a,b,x) = \bigcup_{0\leq i\leq K} h_i^{-1}\left(\text{bound}_{[-6,6]}\left(h_i\left(x\right)\right)\right) \tag{5.13.7}$$

for some strictly increasing linear maps h_i such that, by taking $c_i = h^{-1}(-6)$ and $d_i = h^{-1}(6)$, one retrieves the claim of the Lemma. Notice that one has $h^{-1}\left(\text{bound}_{[p,q]}\left(h(x)\right)\right) = \text{bound}_{[h^{-1}(p),h^{-1}(q)]}\left(x\right)$ for all strictly increasing linear maps h, all $p, q \in \mathbb{R}$ such that $p \leq q$ and all $x \in \mathsf{R}$.

We only sketch a few main steps of the proof.

1. By definition of bound$''$, for all a, $b \in \mathbb{R}$ such that $a \leq b$ and all $x \in \mathsf{R}$, there exist some integers $m_0, \ldots, m_k$ such that

$$\text{bound}''\left(a,b,x\right) = \bigcup_{0\leq i\leq k} m_i + \phi\left(a-m_i, b-m_i, x-m_i\right) \tag{5.13.8}$$

and such that $m_i + \dfrac{a+b}{2} \in [-1,1]$, for each m_i.

In other words,

$$\text{bound}''\left(a,b,x\right) = \bigcup h_i^{-1}\phi\left(h_i(a), h_i(b), h_i(c)\right) \tag{5.13.9}$$

where each $h_i\colon \mathcal{R} \to \mathcal{R}$ is a linear transformation—here a translation—which sends the centre of the interval $[a,b]$ to $[-1,1]$ and preserves the radius of $[a,b]$.

Remark. When saying that h_i preserves the radius of $[a,b]$, we mean that, whenever $a, b \in \mathbb{R}$ with $a \leq b$, there exist $a', b' \in \mathbb{R}$, such that $h_i(\eta\left(a\right)) = \eta\left(a'\right)$, $h_i(\eta\left(b\right)) = \eta\left(b'\right)$ and $b' - a' = b - a$. We omit similar remarks in the remainder of this proof.

2. We first suppose $a = b$. For all real numbers a, by definition of ϕ and the previous point, there exist integers m_i such that

$$\begin{aligned}
&\text{bound}''\left(a,a,x\right) \qquad\qquad (5.13.10)\\
&\quad= \bigcup_{0\leq i\leq k} m_i + \phi\left(a-m_i, a-m_i, x-m_i\right)\\
&\quad= \bigcup_{0\leq i\leq k} m_i + \left(\text{bound}_{[-6,6]}\left(\frac{1}{2}\Big(\text{bound}''\left(2(a-m_i), 2(a-m_i), 2(x-m_i)\right)\Big)\right)\right)\\
&\quad= \bigcup_{0\leq i\leq k} \text{bound}_{[m_i-6,m_i+6]}\left(m_i + \frac{1}{2}\Big(\text{bound}''\left(2(a-m_i), 2(a-m_i), 2(x-m_i)\right)\Big)\right)
\end{aligned}$$

and $a \in [m_i + 6, m_i - 6]$ for each m_i. It is not difficult to deduce from there that $\text{bound}''(a, a, x) = a = \text{bound}_{[a,a]}(x)$. A detailed proof would be similar to the one for the identity operator (Section 5.4).

3. We now suppose that $b - a > 2$. From the definition of ϕ and point 1, we have that

$$\text{bound}''(a, b, x) = \bigcup_{0 \leq i \leq k} h_i^{-1} \psi\left(h_i(a), h_i(b), h_i(c)\right) \tag{5.13.11}$$

where each h_i is a strictly increasing linear transformation such that one has both $\frac{h_i(a)+h_i(b)}{2} \in [-1, 1]$ and $\frac{h_i(b)-h_i(a)}{2} > 2$. By definition of ψ, for each h_i there are strictly increasing linear transformations $g_{i,j}$ such that

$$\psi(h_i(a), h_i(b), h_i(x)) = \bigcup_{0 \leq j \leq k_i} g_{i,j}^{-1}\left(\text{bound}_{[-6,6]}\left(g_{i,j}(x)\right)\right) \tag{5.13.12}$$

and such that

i. $g_{i,j}(a), g_{i,j}(b) \in [-6, 6]$,

ii. $\dfrac{g_{i,j}(a) + g_{i,j}(b)}{2} \in [-1, 1]$ and

iii. $2 \leq \dfrac{g_{i,j}(b) - g_{i,j}(a)}{2} \leq 5$.

Notice that point i follows from points ii and iii.

Putting equations 5.13.11 and 5.13.12 together and after renaming the maps $g_{i,j}$, we conclude that there exist strictly increasing linear transformations g_i such that

$$\begin{aligned} \text{bound}''(a, b, x) &= \bigcup_{0 \leq i \leq K} g_i^{-1}\left(\text{bound}_{[-6,6]}\left(g_i(x)\right)\right) \\ &= \bigcup_{0 \leq i \leq K} \left(\text{bound}_{[g_i^{-1}(-6), g_i^{-1}(6)]}\left(x\right)\right) \end{aligned}$$

and such that, for each i,

i. $a, b \in [g_i^{-1}(-6), g_i^{-1}(6)]$,

ii. $\dfrac{a + b}{2} \in [g_i^{-1}(-1), g_i^{-1}(1)]$ and

iii. $2 \leq \dfrac{g_i(b) - g_i(a)}{2} \leq 5$.

By taking $c_i = g_i^{-1}(-6)$ and $d_i = g_i^{-1}(6)$, it is easy to check that the lemma is satisfied.

4. The remaining cases are technically more tedious but essentially similar.

□

5.14 Finding a positive number among two given numbers

We define a function that indicates which of its arguments in R represents a positive real number, provided that both arguments are total and at least one represents a positive real number. This will be useful for the definition of an operator that finds the root of a function, in Section 5.15.

Lemma 5.14.1. *Let* dtest *be the following function*

$$\begin{array}{l} \mathrm{dtest}\colon \mathsf{R}\times\mathsf{R}\to\mathsf{B} \\ \mathrm{dtest}(x,y) = \mathrm{cases}\ \ 1\le x \ \to\ \mathrm{true} \\ \qquad\qquad\qquad\qquad\quad x\le 2 \ \to\ \mathrm{cases}\ \ 1\le y \ \to\ \mathrm{false} \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad y\le 2 \ \to\ \mathrm{dtest}\,(2x,2y) \end{array} \tag{5.14.1}$$

For any total r and s in $\mathbb{R}$ such that $r>0$ or $s>0$,

$$\begin{array}{l} \eta\,(\mathrm{true})\cup\eta\,(\mathrm{false})\sqsubseteq \mathrm{dtest}\,(\eta\,(r),\eta\,(s))\,, \\ \mathrm{dtest}\,(\eta\,(r),\eta\,(s))\sqsubseteq\eta\,(\mathrm{true}) \implies r>0 \text{ and} \\ \mathrm{dtest}\,(\eta\,(r),\eta\,(s))\sqsubseteq\eta\,(\mathrm{false}) \implies s>0. \end{array} \tag{5.14.2}$$

Proof. We prove by induction on n that

$$\begin{array}{lcl} \left(r>2^{1-n} \text{ or } s>2^{1-n}\right) & \implies & \eta\,(\mathrm{true})\cup\eta\,(\mathrm{false})\sqsubseteq \mathrm{dtest}\,(\eta\,(r),\eta\,(s))\,, \\ & & \mathrm{dtest}\,(\eta\,(r),\eta\,(s))\sqsubseteq\eta\,(\mathrm{true}) \implies r>0 \text{ and} \\ & & \mathrm{dtest}\,(\eta\,(r),\eta\,(s))\sqsubseteq\eta\,(\mathrm{false}) \implies s>0. \end{array} \tag{5.14.3}$$

It is immediate from the definition of dtest that it holds for $n=0$ (*i.e.* $r>2$ or $s>2$). Suppose as induction hypothesis that it holds for some $n\ge 0$ and let r and s be such that that $r>2^{1-(n+1)}$ or $s>2^{1-(n+1)}$ and neither $r>2^{1-n}$ nor $s>2^{1-n}$. We have that $r\le 2$ and $s\le 2$. According to whether $r<1$ and whether $s<1$, we can express $\mathrm{dtest}\,(\eta\,(r),\eta\,(s))$ as follows, by definition of dtest.

$$\begin{array}{l} \text{If } r<1 \text{ and } s<1 \text{ then } \mathrm{dtest}\,(\eta\,(r),\eta\,(s)) = \mathrm{dtest}\,(\eta\,(2r),\eta\,(2s)), \\ \text{if } 1\le r \text{ and } s<1 \text{ then } \mathrm{dtest}\,(\eta\,(r),\eta\,(s)) = \eta\,(\mathrm{true})\cup\mathrm{dtest}\,(\eta\,(2r),\eta\,(2s)), \\ \text{if } r<1 \text{ and } 1\le s \text{ then } \mathrm{dtest}\,(\eta\,(r),\eta\,(s)) = \eta\,(\mathrm{false})\cup\mathrm{dtest}\,(\eta\,(2r),\eta\,(2s)) \text{ and} \\ \text{if } 1\le r \text{ and } 1\le s \text{ then } \mathrm{dtest}\,(\eta\,(r),\eta\,(s)) = \eta\,(\mathrm{true})\cup\eta\,(\mathrm{false})\cup\mathrm{dtest}\,(\eta\,(2r),\eta\,(2s)). \end{array} \tag{5.14.4}$$

Since by induction hypothesis

$$\begin{array}{l} \eta\,(\mathrm{true})\cup\eta\,(\mathrm{false})\sqsubseteq \mathrm{dtest}\,(\eta\,(2r),\eta\,(2s))\,, \\ \mathrm{dtest}\,(\eta\,(2r),\eta\,(2s))\sqsubseteq\eta\,(\mathrm{true}) \implies 2r>0 \text{ and} \\ \mathrm{dtest}\,(\eta\,(2r),\eta\,(2s))\sqsubseteq\eta\,(\mathrm{false}) \implies 2s>0, \end{array} \tag{5.14.5}$$

Property 5.14.3 holds for $n+1$.

□

Finding a natural number at which a function is positive. We sketch the definition of a more general, multi-valued operator witness: $(\mathsf{R} \to \mathsf{R}) \to \mathsf{R}$ that satisfies the following property. If $f\colon \mathsf{R} \to \mathsf{R}$ is a total function for which there exists a total $x \in \mathsf{R}$ satisfying $f(x) > \eta(0)$, then

$$\text{witness}(f) \subseteq \{\eta(r) \mid r \in \mathbb{R} \text{ and } f(\eta(r)) > \eta(0)\}. \tag{5.14.6}$$

It is based on the following result.

Theorem 5.14.2. *Define*

$$\begin{array}{l} \text{witness}'\colon (\mathsf{N} \to \mathsf{R}) \to \mathsf{R} \qquad (5.14.7)\\ \text{witness}'(f) = \ \phi(1,0,0)\\ \qquad\qquad \textit{where}\\ \qquad\qquad \phi(\varepsilon,i,m) = \text{cases}\\ \qquad\qquad\qquad\qquad f(i) \geq \dfrac{\varepsilon}{2} \ \ \to \ \ i\\ \qquad\qquad\qquad\qquad f(i) \leq \varepsilon \ \ \to \ \ \text{if } i < m \text{ then } \phi\left(\frac{\varepsilon}{2}, i+1, m\right)\\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{else } \ \phi\left(\frac{\varepsilon}{2}, 0, m+1\right). \end{array}$$

Then, for any total, continuous function $f\colon \mathsf{N} \to \mathsf{R}$ *such that* $f\big(\eta(n)\big) > 0$ *for some* $n \in \mathbb{N}$*, one has that*

$$\text{witness}'(f) \subseteq \{n \mid n \in \mathbb{N} \textit{ and } f(\eta(n)) > 0\}. \tag{5.14.8}$$

To define the operator witness, let $e\colon \mathsf{N} \to \mathsf{R}$ be an enumeration of the rational numbers in R. Such a function can be defined from a representation $e'\colon \mathsf{N} \to \mathsf{Q}$ of some enumeration of the codes for rational numbers by $e = \iota_{\mathbb{Q},\mathbb{R}} \circ e'$ (where $\iota_{\mathbb{N},\mathbb{R}}$ is defined in Section 5.11 on page 86). Suppose now that $g\colon \mathsf{R} \to \mathsf{R}$ is a total function satisfying $g\big(\eta(r)\big) > 0$ for some $r \in \mathbb{R}$. Because the numerical function that g represents is continuous and $\mathbb{Q}$ is dense in $\mathbb{R}$, there must exist a *rational number* r such that $g(\eta(r)) \neq \eta(0)$. So $\text{witness}'(g \circ e) \subseteq \{[r,r] \mid r \in \mathbb{Q} \text{ and } g(\eta(r)) > (0)\}$. Thus, we define $\text{witness} = \lambda f.\, \text{witness}'(f \circ e)$.

5.15 Zero of a function

We define an operator root: $\mathsf{R} \times \mathsf{R} \times (\mathsf{R} \to \mathsf{R}) \to \mathsf{R}$ such that, if a total function $f\colon \mathsf{R} \to \mathsf{R}$ represents a monotone function with a unique root r in a given interval $[a,b]$ with $a < b$,

then $\mathrm{root}\left(\eta\left(a\right),\eta\left(b\right),f\right)=\eta\left(r\right)$. We employ a technique known as the *trisection of intervals*, used for instance in [5]. It is an adaptation of the classical dichotomy search for the root of a function which is based on the observation that, if a monotone function $f\colon[a,b]\to\mathbb{R}$ has a unique root r and $a\leq a'<b'\leq b$ then $f(a)f(b')<0$ or $f(a')f(b)<0$. Finding which holds of $f(a)f(b')<0$ or $f(a')f(b)<0$ is computable because at least one among a' and b' is guaranteed not to be the root. This allows to deduce in which of the two overlapping intervals $[a,b']$ and $[a',b]$ the root is. The dichotomy method, which performs either test with $a'=b'$ would fail as it could inadvertently pick $a'=b'=r$ and either test $f(a)f(r)<0$ or $f(r)f(b)<0$ would fail.

Recall that $(x<_{\mathrm{p}}0)$ represents the partial test on real numbers which is undefined at 0 (Lemma 5.12.1 on page 87).

Theorem 5.15.1. *Let $f\colon\mathsf{R}\to\mathsf{R}$ be a total function that represents a* monotone*, numerical function with a* unique *root r in a given interval $[a,b]$, with $a<b$. Then one has that* $\mathrm{root}\left(\eta\left(a\right),\eta\left(b\right),f\right)=\eta\left(r\right)$*, where*

$$\begin{aligned}
&\mathrm{root}\colon\mathsf{R}\times\mathsf{R}\times(\mathsf{R}\to\mathsf{R})\to\mathsf{R} && (5.15.1)\\
&\mathrm{root}\left(a,b,f\right)=\text{if }\ w\ \text{ then }\ \text{if }\ f(a)f(b')<_p 0\ \text{ then }\ \mathrm{bound}''\left(a,b',\mathrm{root}\left(a,b',f\right)\right)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{else }\ \mathrm{bound}''\left(a',b,\mathrm{root}\left(a',b,f\right)\right)\\
&\qquad\qquad\qquad\qquad\text{else }\ \text{if }\ f(a')f(b)<_p 0\ \text{ then }\ \mathrm{bound}''\left(a',b,\mathrm{root}\left(a',b,f\right)\right)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{else }\ \mathrm{bound}''\left(a,b',\mathrm{root}\left(a,b',f\right)\right),\\
&b'=\frac{1}{3}a+\frac{2}{3}b,\\
&a'=\frac{2}{3}a+\frac{1}{3}b\ \textit{and}\\
&w=\mathrm{dtest}\left(\mathrm{abs}\left(f(a)f(b')\right),\mathrm{abs}\left(f(a')f(b)\right)\right).
\end{aligned}$$

Notice that by construction of dtest (Lemma 5.14.1), appearing in the definition of w, the elements $f(a)f(b')$ or $f(a')f(b)$ on which the tests $(\cdot<_{\mathrm{p}}0)$ are performed cannot be zero and so the inequality tests always succeed.

Proof. We identify real numbers with their representations in R. By definition, the function $\mathrm{root}\left(a,b,f\right)$ is the least fixed point of G defined by

$$\begin{aligned}
G(g)(a,b,f)=\text{if }\ w\ \text{ then }\ \text{if }\ f(a)f(b')<0\ \text{ then }\ &\mathrm{bound}''\left(a,b',g\left(a,b',f\right)\right) && (5.15.2)\\
\text{else }\ &\mathrm{bound}''\left(a',b,g\left(a',b,f\right)\right)\\
\text{else }\ \text{if }\ f(a')f(b)<0\ \text{ then }\ &\mathrm{bound}''\left(a',b,g\left(a',b,f\right)\right)\\
\text{else }\ &\mathrm{bound}''\left(a,b',g\left(a,b',f\right)\right)\\
& && (5.15.3)
\end{aligned}$$

where $b'=\frac{1}{3}a+\frac{2}{3}b$, $a'=\frac{2}{3}a+\frac{1}{3}b$ and $w=\mathrm{dtest}\left(\mathrm{abs}\left(f(a)f(b')\right),\mathrm{abs}\left(f(a')f(b)\right)\right)$. From Lemma 5.14.1, it follows that there exist some real numbers $a_0,\ldots,a_k$, $b_0,\ldots,b_k$ such

that

$$G(g)(a,b,f) \sqsupseteq \bigcup_{0\leq i\leq k} \text{bound}''\left(a_i, b_i, g\left(a_i, b_i, f\right)\right) \tag{5.15.4}$$

and $a \leq a_i \leq r \leq b_i \leq b$, $a_i < b_i$ and $0 \leq b_i - a_i \leq \frac{2}{3}(b-a)$, for each i. By applying Lemma 5.13.1 on page 89, we obtain that there exist some real numbers $a_0, \ldots, a_k$, $b_0, \ldots, b_k$ such that

$$G(g)(a,b,f) \sqsupseteq \bigcup_{0\leq i\leq k} \text{bound}_{[a_i-3(b_i-a_i),b_i+3(b_i-a_i)]}\left(g\left(a_i, b_i, f\right)\right) \tag{5.15.5}$$

and $a \leq a_i \leq r \leq b_i \leq b$,$a_i < b_i$ and $0 \leq b_i - a_i \leq \frac{2}{3}(b-a)$, for each i. It can be then shown by induction on n that for all $n \geq 1$, there exist some real numbers $a_0, \ldots, a_k$, $b_0, \ldots, b_k$ such that

$$G^n(g)(a,b,f) \sqsupseteq \bigcup_{0\leq i\leq k} \text{bound}_{[a_i-3(b_i-a_i),b_i+3(b_i-a_i)]}\left(g\left(a_i, b_i, f\right)\right) \tag{5.15.6}$$

and $a \leq a_i \leq r \leq b_i \leq b$,$a_i < b_i$ and $0 \leq b_i - a_i \leq \left(\frac{2}{3}\right)^n (b-a)$, for each i. Theorem 5.15.1 follows from this. □

5.16 Multiplication and inverse function

We can now easily give representations of the square function

$$\begin{aligned} \cdot^2 &\colon \mathbb{R} \to \mathbb{R} \qquad (5.16.1)\\ r &\mapsto r^2, \end{aligned}$$

using the root operator defined in Section 5.15 and the square root function defined in Section 5.10. For any real number $r \in [-1,1]$, the function

$$\begin{aligned} \phi &\colon [0,1] \to [0,1] \qquad (5.16.2)\\ s &\mapsto |r| - \sqrt{s} \qquad (5.16.3) \end{aligned}$$

is strictly decreasing and admits r^2 as its unique root. Thus the following is a representation of the square function.

$$\begin{aligned} &-^2 \colon \mathsf{R} \to \mathsf{R} \qquad (5.16.4)\\ &x^2 = \text{cases } \tfrac{1}{2} \leq \text{abs}\,(x) \;\to\; 4 \times \left(\text{sqrt}\left(\tfrac{1}{2}x\right)\right)^2\\ &\qquad\qquad \text{abs}\,(x) \leq 1 \;\to\; \text{root}\left(0, 1, \lambda y.\Big(\text{abs}\,(x) - \text{sqrt}\,(y)\Big)\right) \end{aligned}$$

In practice, one would prefer a direct and more efficient implementation of the square function. However, our main concern is on definability in principle.

The multiplication of real numbers can be represented by

$$\times\colon \mathsf{R}\times\mathsf{R}\to\mathsf{R} \tag{5.16.5}$$
$$x\times y=\frac{1}{2}\Big((x+y)^2-x^2-y^2\Big).$$

To represent the inverse function, we observe that for any real number $r\in[1,6]$, the linear function

$$\phi\colon [\tfrac{1}{6},1]\to[-\tfrac{5}{6},1] \tag{5.16.6}$$
$$s\mapsto sr-1 \tag{5.16.7}$$

is strictly increasing and admits $\frac{1}{r}$ as its unique root. The inverse function can then be represented by

$$\tfrac{1}{\cdot}\colon \mathsf{R}\to\mathsf{R} \tag{5.16.8}$$
$$\begin{array}{lllllllll}\frac{1}{x}= \text{if } x<0 \text{ then } & -\frac{1}{-x} & & & & & & & \\ \text{else} & \text{cases} & x\le 2 & \to & 2\times\frac{1}{2x} & & & & \\ & & 1\le x & \to & \text{cases} & 5\le x & \to & \frac{1}{2}\times\frac{1}{\frac{1}{2}\times x} \\ & & & & & x\le 6 & \to & \text{root}\big(\frac{1}{6},1,\lambda y.xy-1\big)\end{array}$$

where $x<0$ is total if is x not $\eta(0)$ (*cf.* Equation 5.12.1 on page 87). At this stage we omit the proofs as they are very similar to previous ones for other functions.

5.17 Limits

The operator that takes the limit of any convergent sequence of real numbers is not computable [28, 3]. However, its restriction to certain sequences is known to be computable. A sequence α_i is *fast converging* if it satisfies

$$|\alpha_n-\alpha_{n+1}|\le\frac{1}{2^n},\text{ for all } n\in\mathbb{N}.$$

We represent sequences of real numbers by elements of the domain $(\mathsf{N}\to\mathsf{R})$, extending the Definition 5.2.1 in the obvious way.

Theorem 5.17.1. *Let*

$$\lim\colon(\mathsf{N}\to\mathsf{R})\to\mathsf{R}$$
$$\lim(\alpha)=\phi(0,1,\alpha) \tag{5.17.1}$$

where we recursively define

$$\begin{aligned}&\phi\colon \mathsf{N}\times\mathsf{R}\times(\mathsf{N}\to\mathsf{R})\to\mathsf{R}\\&\phi\left(n,\varepsilon,\alpha\right)=\text{bound}''\left(\alpha_n-\varepsilon,\alpha_n+\varepsilon,\phi\left(n+1,\frac{\varepsilon}{2},\alpha\right)\right).\end{aligned}$$

If $\alpha\in(\mathsf{N}\to\mathsf{R})$ *represents a fast convergent sequence* $r_n\in\mathbb{R}$ *then* $\lim(\alpha)=\eta\left(\lim_{n\to\infty}\left(r_n\right)\right)$.

Proof. As a consequence of Lemma 5.13.1 on page 89 we have that, for all $n\in\mathbb{N}$,

$$\phi\left(\eta_{\mathbb{N}}\left(n\right),\eta_{\mathbb{R}}\left(\frac{1}{2^n}\right),\alpha\right)\sqsupseteq\text{bound}_{[r_n-\frac{7}{2^n},r_n+\frac{7}{2^n}]}\left(\phi\left(\eta_{\mathbb{N}}\left(n+1\right),\eta_{\mathbb{R}}\left(\frac{1}{2^{n+1}}\right),\alpha\right)\right)\tag{5.17.2}$$

and

$$\lim_{n\to\infty}r_n\in[r_n-\tfrac{1}{2^n},r_n+\tfrac{1}{2^n}]\subset[r_n-\tfrac{7}{2^n},r_n+\tfrac{7}{2^n}],$$

from which the theorem follows. □

5.17.1 Power series

Many common numerical functions, such as the exponential, the logarithm and other trigonometric functions, can be expressed as power series, *i.e.* are of the form

$$f\colon\mathbb{R}\to\mathbb{R}\tag{5.17.3}$$

$$r\mapsto\sum_{n=0}^{\infty}a_n\left(r-c\right)^n,\tag{5.17.4}$$

where $c\in\mathbb{R}$ and $\left(a_n\right)_{n\in\mathbb{N}}$ is a sequence of real numbers. For simplicity, we only consider power series with an infinite radius of convergence, that is the domain of f is $\mathbb{R}$, and where $c=0$. For example

$$e^r=\sum_{n=0}^{\infty}\frac{r^n}{n!}=1+r+\frac{r^2}{2!}+\frac{r^3}{3!}+\cdots\tag{5.17.5}$$

and

$$\sin(r)=\sum_{n=0}^{\infty}\frac{(-1)^nr^{2n+1}}{(2n+1)!}=r-\frac{r^3}{3!}+\frac{r^5}{5!}-\frac{r^7}{7!}+\cdots\tag{5.17.6}$$

For many functions of interest, the sequence (a_n) can be represented by a function $\mathsf{N} \to \mathsf{R}$ which is definable in the language, using the basic functions previously defined. Furthermore, there exists a computable function $\phi\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that

$$\forall m \in \mathbb{N}, r \in \mathbb{R}, \qquad |r| \leq k \implies \left| \sum_{n=0}^{\phi(m,k)} a_n r^n - f(r) \right| \leq \frac{1}{2^m}. \tag{5.17.7}$$

That is, the function $\phi(m,k)$ tells how many terms of the power series it is enough to add to get an approximation with precision $\frac{1}{2^m}$ of $f(r)$ for $|r| \leq k$. Because ϕ is computable, it can be defined by a PCF term and hence in the language considered here. The restriction of f to $[-k,k]$ is represented by

$$\begin{aligned} &f_k\colon \mathsf{R} \to \mathsf{R} \\ &f_k(r) = \lim \left(\lambda m.\ \sum_{n=0}^{\phi(m,k)} a_n \times (r-c)^n \right), \end{aligned} \tag{5.17.8}$$

and f is represented by

$$\begin{aligned} &f\colon \mathsf{R} \to \mathsf{R} \\ &f(r) = \psi(r,1) \end{aligned} \tag{5.17.9}$$

where

$$\psi(r,k) = \text{cases} \begin{array}{ll} |r| \leq k & \to \quad f_k(r) \\ k-1 \leq |r| & \to \quad \psi\left(r, k+1\right). \end{array}$$

Example: the exponential function

To represent the exponential function

$$\begin{aligned} &e\colon \mathbb{R} \to \mathbb{R} \\ &e^r = \sum_{n=0}^{\infty} \frac{r^n}{n!} = 1 + r + \frac{r^2}{2!} + \frac{r^3}{3!} + \cdots \end{aligned} \tag{5.17.10}$$

we remark that for all $k, n \in \mathbb{N}$ such that $n \geq 2k$ and all $r \in [-k, k]$, one has

$$\begin{aligned}
\left| e^r - \sum_{i=0}^{n-1} \frac{r^i}{i!} \right| &\leq \sum_{i=n}^{\infty} \frac{|r^i|}{i!} \\
&\leq \sum_{i=n}^{\infty} \frac{k^i}{i!} \\
&\leq \frac{k^n}{n!} \sum_{i=0}^{\infty} \left(\frac{k}{n} \right)^i \\
&\leq \frac{k^n}{n!} \sum_{i=0}^{\infty} \left(\frac{k}{2k} \right)^i \qquad \text{(because } n \geq 2k\text{)} \\
&\leq \frac{2k^n}{n!}
\end{aligned}$$

Hence, we can take

$$\phi(k, m) = 2k + m$$

to ensure that

$$\left| e^r - \sum_{i=0}^{\phi(k,m)} \frac{r^i}{i!} \right| \leq \frac{1}{2^m} \tag{5.17.11}$$

for all $k, m \in \mathbb{N}$ and $r \in [-k, k]$. The exponential function can then be defined as follows.

$$\begin{aligned}
&e \colon \mathsf{R} \to \mathsf{R} \\
&e(x) = \psi_e(x, 1)
\end{aligned} \tag{5.17.12}$$

where

$$\begin{aligned}
\psi_e(x, k) = \text{cases} \;\; \text{abs}\,(x) \leq k \qquad &\to \quad \lim \left(\lambda m. \sum_{n=0}^{2k+m} \frac{x^n}{n!} \right) \\
k - 1 \leq \text{abs}\,(x) \quad &\to \quad \psi\,(x, k+1)\,.
\end{aligned}$$

This example illustrates a general manner of representing functions given as power series. There are of course much better algorithms to compute the exponential function.

Chapter 6

Universality at first order

We prove that any total, computable function $f\colon \mathbb{R}^n \to \mathbb{R}$ is definable in the language, in the sense that there exists a program of type $\texttt{real}^{(n)} \to \texttt{real}$ whose denotation, of type $\mathsf{R}^{(n)} \to \mathsf{R}$, represents f (*cf.* Definition 5.2.1). This result shows not only that any first-order, computable total function on the reals is computed by some program but also that our semantics is "close enough" to the operational semantics to account for this universality. In fact, we establish a slightly more general result: all computable functions $A \to \mathbb{R}$ with $A \subseteq \mathbb{R}^n$ recursively open are definable.

Notation. In order to lighten our notation, we often identify real numbers, their representations in $\mathcal{R}$ and their representations in R, thus writing r for $r \in \mathbb{R}$, $\eta(r) \in \mathcal{R}$ and $\eta(r) \in \mathsf{R}$. We use similar conventions for other types.

6.1 Signed-digit representation

In order to establish the above results, we work with signed-digit representation of real numbers [11].

Definition 6.1.1. Let 3 be the set $\{-1, 0, 1\}$. The function

$$\begin{aligned} &\rho\colon 3^{\mathbb{N}} \times \mathbb{N} \to \mathbb{R} \\ &\rho(w, n) = 2^n \sum_{i=0}^{\infty} w(i) 2^{-i-1} \end{aligned} \tag{6.1.1}$$

is a surjection, called the *signed digit representation* of real numbers. Each element (w, n) of $3^{\mathbb{N}} \times \mathbb{N}$ is also called a signed digit representation of $\rho(w, n)$.

In the context of our language and semantics, a *signed digit representation* is an ordered pair (v, m) in

$$\widetilde{\mathsf{R}} = [(\mathsf{N} \to \mathsf{Z}) \times \mathsf{N}]$$

that agrees with some $(w, n) \in 3^{\mathbb{N}} \times \mathbb{N}$ in the sense that $m = \eta\,(n)$ and $v\,(\eta\,(i)) = w(i)$ for all $i \in \mathbb{N}$. Let us define the *partial* surjection

$$\begin{aligned} &e\colon \widetilde{\mathsf{R}} \to 3^{\mathbb{N}} \times \mathbb{N} \\ &e(v, n) = \text{the unique } (w, n) \text{ that agrees with } (v, n), \end{aligned} \tag{6.1.2}$$

defined at each (v, n) such that $v(\eta(i)) \in \{\eta(-1), \eta(0), \eta(1)\}$ for all $i \in \mathbb{N}$ and such that $n = \eta(m)$ for some $m \in \mathbb{N}$. It follows from the work of the previous chapter that there exists a continuous function $\rho^*\colon \widetilde{\mathsf{R}} \to \mathsf{R}$, definable in the language, such that the following diagram commutes on the domain of e:

$$\begin{array}{ccc} 3^{\mathbb{N}} \times \mathbb{N} & \xrightarrow{\ \rho\ } & \mathbb{R} \\ {\scriptstyle e}\uparrow & & \downarrow{\scriptstyle \eta} \\ \widetilde{\mathsf{R}} & \xrightarrow{\ \rho^*\ } & \mathsf{R}. \end{array} \tag{6.1.3}$$

This will be proved as Lemma 6.1.1 below, but we first make a remark about the type which is interpreted by $\widetilde{\mathsf{R}}$. Although we do not have product types in the language, the product of types $(\mathtt{nat} \to \mathtt{nat})$ and $\mathtt{nat}$ can easily be encoded. We choose to define a type $\widetilde{\mathtt{real}}$ as

$$\widetilde{\mathtt{real}} = (\mathtt{nat} \to \mathtt{nat}) \tag{6.1.4}$$

along with programs

$$\begin{aligned} &\text{pairing}\colon (\mathtt{nat} \to \mathtt{nat}) \to \mathtt{nat} \to \widetilde{\mathtt{real}}, \\ &\text{pairing}(v)(m) = \lambda i.\, \mathtt{if}\ i = 0\ \mathtt{then}\ m\ \mathtt{else}\ v(i-1) \end{aligned} \tag{6.1.5}$$

$$\begin{aligned} &\pi_{\text{left}}\colon \widetilde{\mathtt{real}} \to (\mathtt{nat} \to \mathtt{nat}) \\ &\pi_{\text{left}}(\alpha) = \lambda i.\alpha(i+1) \end{aligned}$$

and

$$\begin{aligned} &\pi_{\text{right}}\colon \widetilde{\mathtt{real}} \to \mathtt{nat} \\ &\pi_{\text{right}}(\alpha) = \alpha(0). \end{aligned}$$

We thus consider $\widetilde{\mathsf{R}}$ as the denotation of $\widetilde{\mathtt{real}}$.

Definition 6.1.2. A continuous function $f\colon \widetilde{\mathsf{R}} \to \widetilde{\mathsf{R}}$ *represents* a partial function $g\colon \mathbb{R} \to \mathbb{R}$, if for all $r \in \mathrm{dom}(g)$ and all $\alpha \in \widetilde{\mathsf{R}}$ representing r, one has that $f(\alpha)$ represents $g(r)$.

If furthermore $f\colon \widetilde{\mathsf{R}} \to \widetilde{\mathsf{R}}$ is the denotation of some term of type $\widetilde{\texttt{real}} \to \widetilde{\texttt{real}}$ belonging to the PCF fragment of the language, the function g is said to be *definable* in PCF (using the signed digit representation).

Similarly, a continuous function $f\colon \widetilde{\mathsf{R}} \to \mathsf{R}$ *represents* a function $g\colon \mathbb{R} \to \mathbb{R}$ if for all $r \in \mathrm{dom}(g)$ and $\tilde{r} \in \widetilde{\mathsf{R}}$ satisfying $\rho(e(\tilde{r})) = r$, it holds that $f(\tilde{r}) = \eta(g(r))$.

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{\;g\;} & \mathbb{R} \\ {\scriptstyle \rho\circ e}\uparrow & & \downarrow{\scriptstyle \eta} \\ \widetilde{\mathsf{R}} & \xrightarrow{\;f\;} & \mathsf{R}. \end{array} \tag{6.1.6}$$

If furthermore the function f is the denotation of some program of type $\widetilde{\texttt{real}} \to \texttt{real}$, the function g is said to be *definable* by a program of type $\widetilde{\texttt{real}} \to \texttt{real}$. We extend these notions of representation to functions $f\colon \widetilde{\texttt{real}}^{(n)} \to \widetilde{\texttt{real}}$ and $f\colon \widetilde{\texttt{real}}^{(n)} \to \texttt{real}$ in an obvious way (*cf.* Definition 5.2.1).

Notice that it is ambiguous to say that a function $g\colon \mathbb{R} \to \mathbb{R}$ is definable without specifying whether we mean by a program of type $\widetilde{\texttt{real}} \to \widetilde{\texttt{real}}$, of type $\widetilde{\texttt{real}} \to \texttt{real}$ or of type $\texttt{real} \to \texttt{real}$. However, it will follow from the results of this chapter that these notions are equivalent for g belonging to a large class of partial functions.

Lemma 6.1.1. *The signed digit representation* $\rho\colon 3^{\mathbb{N}} \times \mathbb{N} \to \mathbb{R}$ *is definable by a program of type* $\widetilde{\texttt{real}} \to \texttt{real}$.

Proof. We have to show that there exists a program whose denotation $\rho_*\colon \widetilde{\mathsf{R}} \to \mathsf{R}$ extends the signed digit representation ρ in the following sense: for all $(v, m) \in \widetilde{\mathsf{R}}$ such that $m = \eta(n)$ for some $n \in \mathbb{N}$ and $v(\eta(i)) = w(i)$ for some $w \in 3^{\mathbb{N}}$ and all $i \in \mathbb{N}$, it holds that $\rho_*(v, m) = \eta(\rho(w, n))$. Let $f\colon \mathbb{Z}^{\mathbb{N}} \times \mathbb{N} \to \mathbb{R}$ be the function defined by

$$f(w, n) = 2^n \times \lim_{k\to\infty} (u_k)$$

where the sequence $(u_k)_{k\in\mathbb{N}}$ is defined by

$$u_k = \sum_{i=0}^{k} w'(i) \times 2^{-i-1}$$

and

$$w'(i) = \max(-1, \min(w(i), 1)).$$

Notice that $w'(i) \in \{-1, 0, 1\}$ and that $w'(i) = w(i)$ if $w(i) \in \{-1, 0, 1\}$. Hence the sequence u_k is always convergent and $f(w, n) = \rho(w, n)$ whenever $w \in 3^{\mathbb{N}}$. It is easy to see that $|u_k - \lim_{k\to\infty}(u_k)| \leq \frac{1}{2^k}$, for all $k \in \mathbb{N}$ and all $w \in \mathbb{Z}^{\mathbb{N}}$. Hence we know from Theorem 5.17.1 that f is definable in the language. Let M_f be a program defining f. We choose $\rho_* = [\![M_f]\!]$. It follows from the definition of f that ρ_* satisfies the conditions of the lemma. □

6.2 Computability

The signed digit representation is commonly used to perform real number computation as well as to provide a definition of computable numerical functions. There are many definitions of computability for real numbers and numerical functions in the literature , possibly starting with the one given by Turing [27]. However, most of these definitions are equivalent to the one we consider here.

Definition 6.2.1. A function $\mathbb{R}^n \to \mathbb{R}$ is *computable* if it is computable by a Turing machine with alphabet $\Sigma = \{-1, 0, 1, ., \ldots\}$, using the signed digit representation on its input and output tapes (for more details, see [28, 4, 13]).

It is well known that this is equivalent to PCF definability using the signed-digit representation.

6.3 Universality

Our strategy is to obtain a definition $\mathtt{real}^{(n)} \to \mathtt{real}$ of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ from a PCF definition $\widetilde{\mathtt{real}}^{(n)} \to \widetilde{\mathtt{real}}$ of f. We split the proof in two parts. We start by considering the case of total functions $\mathbb{R} \to \mathbb{R}$, as it includes the essential points of more general results. In a second step, we briefly indicate how to obtain these more general results, which encompass all first-order, computable partial functions with recursively open domains of definition.

6.3.1 Computable functions from $\mathbb{R}$ to $\mathbb{R}$ are definable

We use the fact that the supremum and infimum of any computable total function from $[-1, 1]$ to $[-1, 1]$ can be computed in PCF, using the signed-digit representation [25]. We proceed in two steps. Given a partial function $f\colon \mathbb{R} \to \mathbb{R}$, our first step is to show that the functions $\sup_f, \inf_f\colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{R}$ that compute the supremum and infimum of f at *any* interval $[p, q] \subseteq \mathrm{dom}\,(f)$ are definable (Lemma 6.3.1). In a second step, we define

a continuous function $\phi\colon \mathsf{R} \to \mathsf{R}$ which represents f (Theorem 6.3.2). The idea is to approximate $\phi(x)$ with intervals

$$\left[\inf_f (q-\varepsilon, q+\varepsilon), \sup_f (q-\varepsilon, q+\varepsilon)\right]$$

where q and ε are rational numbers such that $|x-q| \leq \varepsilon$. The following lemma is a consequence of Simpson's result [25].

Lemma 6.3.1. *Let $f\colon \mathbb{R} \to \mathbb{R}$ be a computable function. Then the functions*

$$\begin{aligned} &\sup_f\colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \\ &\sup_f (p,q) = \sup_{r \in [p,q]} f(r) \end{aligned} \tag{6.3.1}$$

and

$$\begin{aligned} &\inf_f\colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \\ &\inf_f (p,q) = \inf_{r \in [p,q]} f(r), \end{aligned} \tag{6.3.2}$$

with domain of definition

$$\operatorname{dom}\left(\sup_f\right) = \operatorname{dom}\left(\inf_f\right) = \left\{(p,q) \in \mathbb{Q}^2 \mid p \leq q \text{ and } [p,q] \subseteq \operatorname{dom}(f)\right\},$$

are definable by PCF programs of type $\mathtt{nat} \to \mathtt{nat} \to \widetilde{\mathtt{real}}$.

Proof. We only consider the case of the supremum operator; the case of the infimum operator is dual.

Simpson proved that the operator

$$\begin{aligned} &\sup_{[-1,1]}\colon \big([-1,1] \to [-1,1]\big) \to [-1,1] \\ &\sup_{[-1,1]}(g) = \sup\{g(r) \mid r \in [-1,1]\} \end{aligned} \tag{6.3.3}$$

with domain

$$\operatorname{dom}\left(\sup_{[-1,1]}\right) = \left\{ g \in [-1,1]^{[-1,1]} \;\middle|\; g \text{ is continuous and total} \right\}$$

is definable in PCF, using the signed digit representation [25]. Notice that the supremum of any computable total function $g\colon [-1,1] \to \mathbb{R}$ exists because a computable total function is continuous and all continuous functions from a compact interval into the reals have a supremum.

Given an interval $[p, q]$ and a continuous, total function $g' \colon [p, q] \to \mathbb{R}$, the function

$$\begin{aligned} &g \colon [-1, 1] \to [-1, 1] \\ &g(r) = g'\left(p + \frac{q-p}{2}(r+1)\right) \end{aligned} \tag{6.3.4}$$

is also continuous and total. Furthermore, its supremum over $[-1, 1]$ is the same as the supremum of g' over $[p, q]$. It is easy to see that the operator

$$\begin{aligned} &\sup' \colon (\mathbb{R} \to [-1, 1]) \times \mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \\ &\sup'(g, p, q) = \sup_{[-1,1]}\left(\lambda r.g\left(p + \frac{q-p}{2}(r+1)\right)\right), \end{aligned} \tag{6.3.5}$$

which takes the supremum at $[p, q]$ of computable total functions from $\mathbb{R}$ to $[-1, 1]$, can be defined by some PCF program, with respect to signed binary representation, using a definition of $\sup_{[-1,1]}$ (and the fact that arithmetic operations on real numbers are definable with respect to the signed digit representation). Let us call such a PCF program, of type $(\widetilde{\texttt{real}} \to \widetilde{\texttt{real}}) \times \texttt{nat} \times \texttt{nat} \to \widetilde{\texttt{real}}$, by the same name $\sup'$.

In order to now compute the supremum of functions whose range is not restricted to $[-1, 1]$, we remark that, for any continuous function $h \colon [p, q] \to \mathbb{R}$, one has that

1. $\sup_{[p,q]}(h) = 2\sup_{[p,q]}\left(\frac{1}{2}h\right)$ and
2. if $\sup_{[p,q]}\left(\lambda r. \max(-1, \min(h(r), 1))\right) \in (-1, 1)$ then $\sup_{[p,q]}(h) \in (-1, 1)$.

Based on these remarks, we define the PCF program

$$\begin{aligned} &\sup'' \colon (\widetilde{\texttt{real}} \to \widetilde{\texttt{real}}) \times \texttt{nat} \times \texttt{nat} \to \widetilde{\texttt{real}} \\ &\sup''(h, p, q) = \text{if } \text{ firstdigit}\,(2 \times s) = 0 \text{ then } s \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{else } 2 \times \sup''\left(\tfrac{1}{2}h, p, q\right) \end{aligned} \tag{6.3.6}$$

where

$$\begin{aligned} s &= \sup'(g_h, p, q) \\ g_h &= \lambda x. \max(-1, \min(h(x), 1)) \end{aligned}$$

and where $\text{firstdigit} \colon \widetilde{\texttt{real}} \to \texttt{nat}$ is some program that takes a signed-digit representation s of a real number and must satisfy the following: it evaluates to 0 if s is of the form $2^m \times 0.d_0d_1d_3\ldots$ with $m < 0$ or $d_0 = d_1 \cdots = d_m = 0$; otherwise, for inputs s representing a real number but not of that form, $\text{firstdigit}(s)$ evaluates to a number different from 0. The point is that, if $\text{firstdigit}(2 \times s) = 0$, then s represents a real number belonging to

the interval $(-1, 1)$. The program $\sup''$ works on the fact that, for $n \in \mathbb{N}$ large enough, the supremum over $[p, q]$ of the function $\frac{1}{2^n}h$ is small enough to warrant that all of its signed-digit representations s satisfy $\mathrm{firstdigit}(s) = 0$.

Since the function $f\colon \mathbb{R} \to \mathbb{R}$ is computable, it is defined by a PCF program $\tilde{f}\colon \widetilde{\texttt{real}} \to \widetilde{\texttt{real}}$. The PCF program

$$\begin{aligned} &\sup_f\colon \texttt{nat} \times \texttt{nat} \to \widetilde{\texttt{real}} \\ &\sup_f(p, q) = \sup''(\tilde{f}, p, q) \end{aligned} \tag{6.3.7}$$

defines the function $\sup_f\colon \mathbb{Q} \times \mathbb{Q} \to \mathbb{R}$. □

Theorem 6.3.2. *Any total computable function $f\colon \mathbb{R} \to \mathbb{R}$ is definable by a program of type* $\texttt{real} \to \texttt{real}$.

Proof. We define a continuous function $\phi\colon \mathsf{R} \to \mathsf{R}$ that represents f and is the denotation of a program of type $\texttt{real} \to \texttt{real}$. For each real number x, the function ϕ works by finding some rational number q such that $|x - q| \leq \varepsilon$ and by approximating $f(x)$ with some interval close to the interval $[\inf_f(q - \varepsilon, q + \varepsilon), \sup_f(q - \varepsilon, q + \varepsilon)]$, for ε smaller and smaller, using the function $\mathrm{bound}''\colon \mathsf{R} \times \mathsf{R} \times \mathsf{R} \to \mathsf{R}$ defined in Section 5.13 (before Lemma 5.13.1, which formulates and proves its specification). We fix a computable enumeration $(q_i)_{i \in \mathbb{N}}$ of $\mathbb{Q}$ and also call $q\colon \mathsf{N} \to \mathsf{Q}$ some corresponding representation in the model. The function ϕ is defined as

$$\begin{aligned} &\phi\colon \mathsf{R} \to \mathsf{R} \\ &\phi(x) = \psi(x, 1, 0) \end{aligned} \tag{6.3.8}$$

where

$$\begin{aligned} &\psi\colon \mathsf{R} \times \mathsf{Q} \times \mathsf{N} \to \mathsf{R} \\ &\psi(x, \varepsilon, i) = \text{cases} \\ &\qquad \mathrm{abs}\,(x - q_i) \leq \varepsilon \quad \to \quad \mathrm{bound}''\left(a, b, \psi\left(x, \frac{\varepsilon}{2}, 0\right)\right) \\ &\qquad \mathrm{abs}\,(x - q_i) \geq \frac{\varepsilon}{2} \quad \to \quad \psi(x, \varepsilon, i + 1) \end{aligned}$$

with

$$\begin{aligned} &a = \inf_f(q_i - \varepsilon, q_i + \varepsilon) \text{ and} \\ &b = \sup_f(q_i - \varepsilon, q_i + \varepsilon). \end{aligned}$$

Let r be a real number and let us convince ourselves that $\phi(r) = f(r)$. Let ε be a positive real number and define

$$a_\varepsilon = \inf\{f(t) \mid t \in [r-2\varepsilon, r+2\varepsilon]\}$$
$$b_\varepsilon = \sup\{f(t) \mid t \in [r-2\varepsilon, r+2\varepsilon]\}.$$

Let i_ε be the smallest natural number such that $|r - q_{i_\varepsilon}| < \frac{\varepsilon}{2}$. By definition of $\psi(r, \varepsilon, 0)$, there exist $a_0, b_0, \ldots, a_J, b_J \in \mathbb{R}$ satisfying

$$a_\varepsilon \leq a_j \leq f(r) \leq b_j \leq b_\varepsilon \tag{6.3.9}$$

for each j, and such that

$$\bigcup_{0\leq j\leq J} \text{bound}''\left(a_j, b_j, \psi\left(r, \frac{\varepsilon}{2}, 0\right)\right) \sqsubseteq \psi^{i_\varepsilon}(r, \varepsilon, 0). \tag{6.3.10}$$

By Lemma 5.13.1, for each j, there exist finitely many intervals $[c_{j,0}, d_{j,0}], \ldots, [c_{j,K_j}, d_{j,K_j}]$, such that

$$\text{bound}''\left(a_j, b_j, \psi\left(r, \frac{\varepsilon}{2}, 0\right)\right) = \bigcup_{k\in K_j} \text{bound}_{[c_{j,k}, d_{j,k}]}\left(\psi^{i_\varepsilon}(r, \varepsilon, 0)\right)$$

and

$$[a_j - 3(b_j - a_j), b_j + 3(b_j - a_j)] \sqsubseteq [c_{j,k}, d_{j,k}] \sqsubseteq [a_j, b_j] \sqsubseteq [f(r), f(r)].$$

From this and equations 6.3.9 and 6.3.10, we obtain that there exist some intervals $[c_0, d_0], \ldots, [c_K, d_K]$ such that

$$\bigcup_{0\leq k\leq K} \text{bound}_{[c_k, d_k]}\left(\psi\left(r, \frac{\varepsilon}{2}, 0\right)\right) \sqsubseteq \psi^{i_\varepsilon}(r, \varepsilon, 0)$$

with

$$[a_\varepsilon - 3(b_\varepsilon - a_\varepsilon), b_\varepsilon + 3(b_\varepsilon - a_\varepsilon)] \sqsubseteq [c_k, d_k] \sqsubseteq [f(r), f(r)].$$

It easy to prove from there that $\psi(r, 1, 0) = f(r)$, that is $\phi(r) = f(r)$. □

6.3.2 Generalizations

The purpose of this section is two-fold. Firstly, we generalize the result of the previous section to functions of several arguments. Secondly, and at the same time, we take into account computable partial functions that are defined on an open set, such as the division or the logarithm operations. The aim is not to characterize definable partial functions at the most general level but to show that Theorem 6.3.2 extends to some common partial functions in a relatively easy way.

Lemma 6.3.3. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a computable function. Then the functions*

$$\begin{aligned} &\sup_f\colon (\mathbb{Q}\times\mathbb{Q})^n \to \mathbb{R} \\ &\sup_f\big((p_1,q_1),\ldots,(p_n,q_n)\big) = \sup\{f(r) \mid r \in [p_1,q_1]\times\ldots\times[p_n,q_n]\} \end{aligned} \tag{6.3.11}$$

and

$$\begin{aligned} &\inf_f\colon (\mathbb{Q}\times\mathbb{Q})^n \to \mathbb{R} \\ &\inf_f\big((p_1,q_1),\ldots,(p_n,q_n)\big) = \inf\{f(r) \mid r \in [p_1,q_1]\times\ldots\times[p_n,q_n]\} \end{aligned} \tag{6.3.12}$$

with domain of definition

$$\begin{aligned} \mathrm{dom}\big(\sup_f\big) = \mathrm{dom}\,(\inf_f) = \big\{\big((p_1,q_1),\ldots,(p_n,q_n)\big) \in (\mathbb{Q}\times\mathbb{Q})^n \\ \mid p_1 \le q_1,\ldots,p_n \le q_n \\ \textit{and } [p_1,q_1]\times\ldots\times[p_n,q_n] \subseteq \mathrm{dom}\,(f)\big\}, \end{aligned} \tag{6.3.13}$$

are definable by programs of type $(\mathtt{nat}\times\mathtt{nat})^n \to \widetilde{\mathtt{real}}$.

Proof. This follows from Lemma 6.3.1 and the fact that, over any n-dimensional compact ball $K = [p_1,q_1]\times\ldots\times[p_n,q_n]$, we have that

$$\sup_K(f) = \sup_{[p_1,q_1]}\left(\lambda r_1.\ \sup_{[p_2,q_2]}\left(\lambda r_2.\ldots\ \sup_{[p_n,q_n]}\big(\lambda r_n.f(r_1,\ldots,r_n)\big)\right)\right) \tag{6.3.14}$$

and similarly for $\inf_K(f)$. □

In order to apply the method of proof of Theorem 6.3.2 to partial functions $f\colon \mathbb{R}^n \to \mathbb{R}$, we need to be able to find smaller and smaller compact neighbourhoods that will correspond to the intervals $[q-\varepsilon, q+\varepsilon]$. A sufficient property to proceed in this manner is that $\mathrm{dom}(f)$ be recursively open in the sense of the following definition.

Definition 6.3.1. Let $q_1,\ldots,q_n$ and $\varepsilon > 0$ be rational numbers. The *closed rational ball* of $\mathbb{R}^n$ of centre $(q_1,\ldots,q_n)$ and radius ε is the set

$$B\,(q_1,\ldots,q_n,\varepsilon) = \big\{(r_1,\ldots,r_n) \in \mathbb{R}^n \ \mid\ \max\big(|r_1-q_1|,\ldots,|r_n-q_n|\big) \le \varepsilon\big\}.$$

Let S be a subset of $\mathbb{R}^n$. We say that S is *recursively open* if there exists a computable function

$$\nu\colon \mathbb{N} \to \mathbb{Q}^n\times\mathbb{Q}$$

such that

1. for all $k \in \mathbb{N}$, we have $B(\nu(k)) \subseteq S$,

2. for all real $s \in S$ and all $\varepsilon > 0$, there exists $k \in \mathbb{N}$ such that $B\big(\nu(k)\big)$ contains s and has a radius smaller than ε.

We will use the following characterization of recursively open subsets of $\mathbb{R}^n$.

Lemma 6.3.4. *A subset S of $\mathbb{R}^n$ is recursively open if and only if there exists a computable function*

$$\nu \colon \mathbb{N} \times \mathbb{Q} \to \mathbb{Q}^n \times \mathbb{Q}$$

satisfying the following property. For any $s \in S$ and $\varepsilon > 0$

1. *The radius of $B\left(\nu(k,\varepsilon)\right)$ is smaller than ε for all $k \in \mathbb{N}$,*
2. *the closed ball $B\left(\nu(k,\varepsilon)\right)$ is included in S for all $k \in \mathbb{N}$ and*
3. *there exists $k \in \mathbb{N}$ such that s is inside the ball of same centre as $B\left(\nu(k,\varepsilon)\right)$ and with radius $\varepsilon'/8$, where ε' is the radius of $B(\nu(k,\varepsilon))$.*

Proof. We only show the existence of the function $\nu \colon \mathbb{N} \times \mathbb{Q} \to \mathbb{Q}^n \times \mathbb{Q}$ for any recursively open set. Let S be a recursively open subset of $\mathbb{R}^n$ and let the function $\nu' \colon \mathbb{N} \to \mathbb{Q}^n \times \mathbb{Q}$ be as in Definition 6.3.1. Let $q \colon \mathbb{N} \to \mathbb{Q}^n$ be a computable enumeration of $\mathbb{Q}^n$. We give an algorithm for a function $\nu \colon \mathbb{N} \times \mathbb{Q} \to \mathbb{Q}^n \times \mathbb{Q}$ that satisfies the requirements of the lemma:

$$\begin{array}{lll} \nu(\langle i,j\rangle,\varepsilon) = & \text{if} & q(j) \in B(\nu'(i)) \text{ and } \operatorname{radius}\big(B(\nu'(i))\big) < \varepsilon \\ & \text{then} & (q(j),\varepsilon') \\ & \text{else} & \text{some } \nu'(k) \text{ such that } \operatorname{radius}\big(B(\nu'(k))\big) < \varepsilon \end{array}$$

where

$$\varepsilon' = \operatorname{radius}\Big(B(\nu'(i))\Big) - \operatorname{distance}\Big(q(j), \operatorname{centre}\big(B(\nu'(i))\big)\Big).$$

and the natural number $\langle i,j\rangle$ is a code for the ordered pair $(i,j) \in \mathbb{N}^2$. The distance between two points in $\mathbb{R}^n$ is given by the maximum difference between corresponding coordinates.

To see that ν satisfies the desired property, let us take $s \in S$ and $\varepsilon > 0$. Parts 1 and 2 of the property are clearly satisfied. Let us show part 3. Let $i \in \mathbb{N}$ be such that the ball $B(\nu'(i))$ has s as a member and has a radius smaller than ε (such an number i exists by definition of ν'). Call d the distance from s to the edge of $B(\nu'(i))$ and let $j \in \mathbb{N}$ be such that $q(j)$ is at a distance less than $d/16$ from s. It is not hard to check that $s \in \nu\left(\langle i,j\rangle,\varepsilon\right)$. ∎

Main result of this chapter

The following generalizes Theorem 6.3.2.

Theorem 6.3.5. *Let f be a computable partial function from $\mathbb{R}^n$ to $\mathbb{R}$ such that* $\mathrm{dom}(f)$ *is recursively open. Then f is definable.*

Proof. The proof is very similar to the one of Theorem 6.3.2. We define a function $\psi\colon \mathsf{R}^n \times \mathsf{Q} \times \mathsf{N} \to \mathsf{R}$ that approximates $f(x_1, \ldots, x_n)$ with

$$\left[\inf_{r\in K}\big(f(r)\big), \sup_{r\in K}\big(f(r)\big)\right]$$

where $K = [q_1 - \varepsilon, q_1 + \varepsilon] \times \ldots \times [q_n - \varepsilon, q_n + \varepsilon] =$ for some rational points $(q_1, \ldots, q_n)$ such that $|x_1 - q_1|, \ldots, |x_n - q_n| \leq \varepsilon$. We apply Lemma 6.3.4 to easily find the sets K while retaining a definition of ψ very similar to the one given in the proof of Theorem 6.3.2.

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a computable function such that $\mathrm{dom}(f)$ is recursively open and let $\nu_f\colon \mathsf{N} \times \mathsf{Q} \to \mathsf{Q}^n \times \mathsf{Q}$ be a representation of the corresponding function ν given by Lemma 6.3.4. We define

$$\begin{aligned} &\phi\colon \mathsf{R}^n \to \mathsf{R} \\ &\phi(x) = \psi(x, 1, 0) \end{aligned} \tag{6.3.15}$$

where

$$\begin{aligned} &\psi\colon \mathsf{R}^n \times \mathsf{Q} \times \mathsf{N} \to \mathsf{R} \\ &\psi(x, \varepsilon, i) = \text{cases} \\ &\qquad \delta(x, q) \leq \frac{\varepsilon'}{2} \quad \to \quad \mathrm{bound}''\left(a, b, \psi\left(x, \frac{\varepsilon}{2}, 0\right)\right) \\ &\qquad \delta(x, q) \geq \frac{\varepsilon'}{4} \quad \to \quad \psi(x, \varepsilon, i+1) \end{aligned}$$

where

$$\begin{aligned} &(q, \varepsilon') = \nu_f(i, \varepsilon), \\ &(x_1, \ldots, x_n) = x, \\ &(q_1, \ldots, q_n) = q, \\ &\delta(x, q) = \max\left(|x_1 - q_1|, \ldots, |x_n - q_n|\right), \\ &a = \inf_f\left(\left(q_1 - \varepsilon', q_1 + \varepsilon'\right), \ldots, \left(q_n - \varepsilon', q_n + \varepsilon'\right)\right) \text{ and} \\ &b = \sup_f\left(\left(q_1 - \varepsilon', q_1 + \varepsilon'\right), \ldots, \left(q_n - \varepsilon', q_n + \varepsilon'\right)\right). \end{aligned}$$

We omit further details as Lemma 6.3.4 ensures that the proof of Theorem 6.3.2 carries through in its essential points. □

6.4 Concluding remarks

To prove first-order universality of the language, we based our approach on one specific characterization of computable functions (Definition 6.2.1). There are however at least two other characterizations which might have seemed a more natural choice in our context. We briefly explain why they are not applicable, at least directly, in the context of our approximate semantics.

Brattka [3] characterized computable real-valued functions as those belonging to a certain set of *recursive relations*. This set of recursive relations is the smallest set of relations between finite products of $\mathbb{N}$ and $\mathbb{R}$ which

1. contains a few basic functions corresponding to the field structure of $\mathbb{R}$ and Peano arithmetic on $\mathbb{N}$,
2. is closed under finite projections,
3. is closed under certain "recursion operators"—called juxtaposition, composition, iteration, minimization and normed limitation operators— and
4. contains the relation $\mathrm{Ord}_{\mathbb{R}} := \{(x, 0) \mid x < 0\} \cup \{(x, 1) \mid x + 1 > 1\} \subseteq \mathbb{R} \times \mathbb{N}$.

From the results of the previous chapter, it is clear that the basic functions, the projections and the recursion operators are definable in our context. However, the relation $\mathrm{Ord}_{\mathbb{R}}$ cannot be represented in our semantic model, as this is equivalent to representing rtest.

The other characterization, by Brattka and Hertling [4], makes use of "Feasible Real Random Access Machine" manipulating registers for real numbers and natural numbers with some basic operations. However an operator equivalent to rtest is again required among the basic operations, which makes it impossible to simulate those RAM machines directly in our semantic model.

What would certainly be possible, however, is to use either of these two characterizations to show that all computable functions are computed by some program of the language, in the sense of Definition 5.2.1. But the possibility is open, in principle, that such a program has a denotation strictly below its operational behaviour. Thus, without either inspecting or modifying the constructions of the above two papers, it is not possible to apply their results to obtain our definability results.

As we discussed in Chapter 5, there are programs that compute total functions but whose denotation does not represent these functions. Although it wasn't *a priori* known that all computable total functions, and a large class of computable partial functions, are definable, Theorem 6.3.5 establishes that, in fact, this is the case. It is in this sense that our denotational semantics is "close enough" to the operational semantics.

Chapter 7

Sequentiality in continuous domains

This chapter, which diverges a bit from the main flow of the previous ones, provides some explanation as to why modelling sequential, real number computation using domains is inherently difficult.

Escardó, Hofmann and Streicher investigated the possibility of sequential computation on the real line via its interval-domain environment [9]. Their main result is that sequential computation on the reals via the interval domain is extremely restrictive, to the extent that not even a basic operation such as addition is sequential. The argument in [9] has two main steps: (1) no extension of the addition operation on the real numbers to the interval domain is Vuillemin sequential (see Definition 7.1.1 below), and (2) under natural assumptions for a sequential programming language, the weak parallel-or operator is definable from any function that fails to be Vuillemin sequential. Escardó, Hofmann and Streicher asked whether this limitation is due to some intrinsic property of the interval domain, or whether it holds for any domain within a certain class as well.

A main result of the present work is that the first step generalizes to any continuous domain environment for the real line (Theorem 7.2.4 and its corollary). Thus, we can say that the limitation is not specific to the interval domain, but is rather due to a property of the real line. A second contribution of the present work is to identify this property. We show that any continuous domain environment for any *connected* topological space exhibits the parallel effect. More precisely, we prove that only very restricted binary operations on such a space can be extended to Vuillemin sequential operations on a continuous domain environment (Theorem 7.2.3). Step (2) is discussed in the concluding section 7.3.

7.1 Vuillemin sequentiality

The following definition is taken from [9] and comes from a similar one originally proposed for products of flat domains (See [21]). The intuition is that if a binary function is

sequential, then it must look at one of its arguments first, and if there is no progress at this argument, then the value of the function itself cannot make any progress either.

Definition 7.1.1. A continuous function $f\colon D \times E \to F$ of domains is called *Vuillemin sequential* if, for any d in D and e in E,

1. $f(d, e') = f(d, e)$ for all $e' \sqsupseteq e$, or
2. $f(d', e) = f(d, e)$ for all $d' \sqsupseteq d$.

Equivalently, the function $f(-, e)$ is constant at $\uparrow d$ or the function $f(d, -)$ is constant at $\uparrow e$.

Definition 7.1.2. A *domain environment* for a topological space X is a continuous domain E containing X as a subspace in the relative Scott topology.

Notice that we don't require a domain environment to be the subspace of maximal elements, thus being less restrictive than in [12].

Definition 7.1.3. If E is a domain environment for a topological space X, we say that an operation $X \times X \to X$ is *Vuillemin sequential* if it has at least one Vuillemin sequential extension $E \times E \to E$.

7.2 Vuillemin sequentiality on connected spaces

We show that for any continuous domain environment of a connected topological space, only very restricted binary operations on the space can be Vuillemin sequential. We split the argument in two steps. Lemma 7.2.1 doesn't assume connectedness, but item (1) of its conclusion is closely related to connectedness, as it becomes apparent in Lemma 7.2.2.

Lemma 7.2.1. *If $f\colon D \times E \to F$ is a Vuillemin sequential function on continuous domains and R is a subspace of E, then for every d in D,*

1. *$f(d, -)$ is constant at each open set of some open cover of R, or*
2. *$f(-, r)$ is constant at $\uparrow d$ for some r in R.*

Proof. Assume that (2) doesn't hold. To prove (1), let r be in R and Y the set of elements way below r. Because $f(-, r)$ is not constant at $\uparrow d$, there are d_0 and d_1 in $\uparrow d$ such that

$$f(d_0, r) \neq f(d_1, r). \tag{7.2.1}$$

Since f is Scott continuous and r is the supremum of the directed set Y, we have both

$$f(d_0, r) = \bigsqcup_{y \in Y} f(d_0, y) \text{ and } f(d_1, r) = \bigsqcup_{y \in Y} f(d_1, y). \tag{7.2.2}$$

From assertions (7.2.1) and (7.2.2), one concludes that, for some y_0 in Y,

$$f(d_0, y_0) \neq f(d_1, y_0)$$

as, otherwise, one would have $\bigsqcup_{y \in Y} f(d_0, y) = \bigsqcup_{y \in Y} f(d_1, y)$ by continuity of f, and hence $f(d_0, r) = f(d_1, r)$, contradicting (7.2.1). In other words, the function $f(-, y_0)$ is not constant at $\uparrow d$. So, because f is Vuillemin sequential, the function $f(d, -)$ is constant at $\uparrow y_0$. And because $\uparrow y_0$ is a neighbourhood of r, there is an open subset U_r of $\uparrow y_0$ such that U_r has r as a member. Then the collection $\{U_r\}_{r \in R}$ is an open cover of R such that $f(d, -)$ is constant at each U_r, as required. □

Recall that a topological space R is called *connected* if whenever two open sets V and W are such that $V \cap W = \emptyset$ and $V \cup W = R$, one of the sets V and W is empty. Equivalently, R is connected if R and $\emptyset$ are the only clopens (both closed and open). Typical examples of connected spaces are the real intervals, including $\mathbb{R}$ itself. We will use the characterization given by Lemma 7.2.2(3) below, which is a reformulation of the well known characterization (2).

Lemma 7.2.2. *The following are equivalent for any non-empty space R:*

1. *R is connected.*
2. *Any continuous function from R to a discrete space is constant.*
3. *Any function defined on R which is constant at each open set of some open cover of R is constant.*

Although this result is standard, we include a proof for the sake of completeness.

Proof. (1) $\implies$ (2). Suppose X is a discrete space and $f \colon R \to X$ is continuous. Let $r \in R$. The set $f^{-1}(\{f(r)\})$ is not empty (because it contains r) and is a clopen (because $\{f(r)\}$ is a clopen and f is continuous), hence it is the whole space R (because R is connected), which means that f is constant.

(2) $\implies$ (3). Let $f \colon R \to X$ be a function defined on R and constant at each open set of some open cover $\{U_i\}$ of R. Endow X with the discrete topology. Then f is continuous, because the singletons form a base of X, and, for any element $\{x\}$ of that base, the set $f^{-1}(\{x\})$ is open as it is a union of open sets, namely those U_i whose direct image is $\{x\}$. So, by (2), f is constant.

(3) $\implies$ (1). If U is a clopen in R, then U and its complement form an open cover of R and, by (3), the characteristic function of U has to be constant, so U has to be $\emptyset$ or R. □

The following is an immediate consequence of Lemmas 7.2.1 and 7.2.2.

Lemma 7.2.3. *If $f\colon D\times E \to F$ is a Vuillemin sequential function on continuous domains and R is a connected subspace of E, then for every $d \in D$,*

1. *the function $f(d,-)$ is constant at R , or*
2. *the function $f(-,r)$ is constant at $\uparrow d$ for some $r \in R$.*

We say that a function is *locally constant at a point* if it is constant at some neighbourhood of the point. Given property p and a point y, we say that $p(y')$ *holds for some y' as close to y as one wishes* if for every neighbourhood V of y there is $y' \in V$ such that $p(y')$ holds. Recall that a space is locally connected if every point has a neighbourhood base of connected sets.

Theorem 7.2.4. *If X is a locally connected space and a function $g\colon X \times X \to X$ is Vuillemin sequential with respect to some continuous domain environment, then for any x and y in X,*

1. *the function $g(x,-)$ is locally constant at y, or*
2. *the function $g(-,y')$ is locally constant at x, for some y' as close to y as one wishes.*

Proof. Let E be a domain environment for X and $f\colon E \times E \to E$ a Vuillemin sequential extension of g. Let x and y be in X and assume (2) doesn't hold. This means that for some neighbourhood V of y and all $y' \in V$, the function $g(-,y')$ fails to be locally constant at x, and so, for every d way below x, its extension $f(-,y')$ is not constant at $\uparrow d$. By local connectedness, we may assume that V is connected, and, by Lemma 7.2.3 with $R = V$, we conclude that for every d way below x, the function $f(d,-)$ is constant at V. Hence, by continuity of f, the function $f(x,-)$, and hence $g(x,-)$ is constant at V, i.e. (1) holds. □

As already discussed, it was shown in [9] that when E is the interval domain and X is the Euclidean real line embedded into E by the singleton map $x \mapsto \{x\}$, there is no Vuillemin sequential map $E \times E \to E$ extending the addition operation. Our main result is that this holds for all domain environments and many operations.

Corollary 7.2.5. *Addition and multiplication fail to be Vuillemin sequential, for any continuous domain environment for the real line.*

Note that condition (2) in Theorem 7.2.4 doesn't imply that $g(-,y)$ is locally constant at x. For example, let X be the real line and define g as $g(x,y) = \max(|x|,y)$. For all y greater than 0, the function $g(-,y)$ is locally constant at 0. So, for $x = 0$ and $y = 0$, the function g satisfies condition (2). But the function $g(-,0)$ is not locally constant at 0. However, this example fails to be Vuillemin sequential (consider $x = y = 1$ in the theorem).

Remark By just skipping the definition of Y in the proof of Lemma 7.2.1, one obtains a proof of the following, slightly more general lemma.

Lemma 7.2.6. *Let $f\colon D \times E \to F$ be a Vuillemin sequential function on dcpo's and R a subset of E. Suppose that, for every r in R, there exists a directed subset Y of E such that r is the supremum of Y and, for each y in Y, the upper set $\uparrow y$ is a neighbourhood of r. Then, for every d in D,*

1. *$f(d,-)$ is constant at each open set of some open cover of R, or*
2. *$f(-,r)$ is constant at $\uparrow d$ for some r in R.*

One example where this generalization is of interest is a particular quotient of the domain 3^∞, where 3^∞ is the set of finite and infinite sequences of digits -1, 0 or 1, with the prefix ordering. The maximal elements in 3^∞, that is the infinite sequences, are identified with real numbers in the interval $[-1,1]$ via the map $s \mapsto \sum_{i\geq 0} \frac{s_i}{2^{i+1}}$. For any s in 3^∞, the set of real numbers above s is an interval. We define an equivalence relation $\sim$ on 3^∞ by stipulating that $s \sim s'$ if and only if s and s' define the same interval. The quotient $3^\infty/\sim$ is a non-continuous dcpo. In fact, it is easy to verify that, although every maximal element is the directed join of elements way below it, a non-maximal element doesn't have any element way below it other than bottom. Nevertheless, Lemma 7.2.6 allows us to immediately extend Theorem 7.2.3 and Theorem 7.2.4, with the same proofs, to the case where $\mathbb{R}$ is embedded into $3^\infty/\sim$. It is not too hard to check that this dcpo is quasicontinuous [12, p. 226], and it is natural to ask whether our results generalize to such dcpo's. This is left for future work.

7.3 Concluding remarks

Our result generalizes only one part of the results by Escardó, Hofmann and Streicher. It was shown in [9] that the wpor operator is definable in a programming language for real number computation whose semantics is based on the interval domain, which we haven't generalized to continuous domains. The argument in [9] is as follows. Let $\Sigma = \{\bot, \top\}$ be the Sierpinski domain, in which $\bot \sqsubseteq \top$. Let $\mathcal{L}_\mathbb{R}$ be a programming language with a type for real numbers interpreted as the interval domain $\mathcal{R}$ and assume that

1. for any $a, b \in \mathcal{R}$ with $a \sqsubseteq b$ and $a \neq b$, there is a continuous, $\mathcal{L}_\mathbb{R}$-definable function $\mathrm{test}_{a,b}\colon \mathcal{R} \to \Sigma$ such that $\mathrm{test}_{a,b}(a) = \bot$ and $\mathrm{test}_{a,b}(b) = \top$,
2. for any $a, b \in \mathcal{R}$ with $a \sqsubseteq b$, there is a continuous, $\mathcal{L}_\mathbb{R}$-definable function $\mathrm{path}_{a,b}\colon \Sigma \to \mathcal{R}$ such that $\mathrm{path}_{a,b}(\bot) = a$ and $\mathrm{path}_{a,b}(\top) = b$ and
3. for any rational number p, the unary operators $x \mapsto p + x$ and $x \mapsto p \times x$ from $\mathcal{R}$ to $\mathcal{R}$ are $\mathcal{L}_\mathbb{R}$-definable.

Those operators are regarded as obviously sequential and their $\mathcal{L}_{\mathbb{R}}$-definability as a reasonable requirement for any language for real number computation. Given a function $f\colon \mathcal{R}^m \to \mathcal{R}$, a function $h\colon \mathcal{R}^n \to \mathcal{R}$ is said to be f-definable if it can be obtained from these operators, from f and from $\top$ and $\bot$, using projections and substitution. It was proved that, from any extension f of the addition, the wpor operator is f-definable; for this reason, any extension of the addition to $\mathcal{R}$ is called *inherently parallel*. More precisely, it was proved that

1. if $f\colon \mathcal{R}\times\mathcal{R} \to \mathcal{R}$ is an extension of the addition, then f is not Vuillemin sequential.
2. if $f\colon \mathcal{R}\times\mathcal{R} \to \mathcal{R}$ is not Vuillemin sequential, that is if there exists $(u,v) \sqsubseteq (u',v')$ in $\mathcal{R}\times\mathcal{R}$ such that $f(u,v) \neq f(u',v)$ and $f(u,v) \neq f(u,v')$, then for such u, v, u' and v', there exists an f-definable function $h\colon \mathcal{R} \to \mathcal{R}$ such that $h(a \sqcap b) \neq h(a) \sqcap h(b)$, where $a = f(u',v)$ and $b = f(u,v')$, and

$$\mathrm{wpor}(x,y) = \mathrm{test}_{h(a\sqcap b),h(a)\sqcap h(b)}\left(f\left(\mathrm{path}_{u,u'}(x), \mathrm{path}_{v,v'}(y)\right)\right). \tag{7.3.1}$$

We have generalized the first step to continuous domains. However, we were unable to generalize the construction of the function h in the second step. The argument in [9] uses at least three properties of the interval domain: let a and b be compatible in $\mathcal{R}$; then

1. the meet $a \sqcap b$ of a and b exists,

and if $a \sqcap b \neq a$ and $a \sqcap b \neq b$ then

2. for all $r \in \mathbb{R}$, $a \sqcap b \sqsubseteq \eta(r) \implies a \sqsubseteq \eta(r)$ or $b \sqsubseteq \eta(r)$ and
3. there exists $s \in \mathbb{R}$ such that $a \sqsubseteq \eta(s)$ and $b \not\sqsubseteq \eta(s)$.

Those properties are not satisfied by continuous domains in general, nor by those having finite meets of compatible elements.

Chapter 8

Conclusion

8.1 Summary

The undecidability of (in)equality tests on real numbers from a constructive point of view creates problems in the design of control-flow mechanisms for programming languages for exact real-number computation, as discussed in the introduction. A good operational solution, based on constructive analysis, has been proposed by Boehm and Cartwright [2] and developed by Marcial-Romero and Escardó [17] to some extent. A main contribution of [17] is that the Hoare power domain of the interval domain can be used to reason about partial correctness in a natural way, combining real analysis and domain theory. A main problem left open in the same work is the development of a denotational semantics that would allow for total correctness proofs. Marcial-Romero and Escardó ruled out the Smyth and Plotkin power domains for such a semantics, by proving that there is no continuous interpretation of Boehm and Cartwright's `rtest` construction in those power domains.

In this work, we have shown that the best continuous approximation of `rtest` in the Smyth power domain allows one to develop total correctness proofs based on real analysis and denotational semantics, in a natural way, avoiding termination proofs based on operational semantics. Technically, our main contribution is the formulation and proof of a suitable computational adequacy theorem for the approximate semantics, which we expressed as $[\![M]\!] \sqsubseteq [M]$, for any program M of ground type. We then gave several examples of programs for various numerical functions and used the adequacy result to show those programs correct. The thrust of our arguments is that, for such programs M, one has that their denotational meaning $[\![M]\!]$ in our approximate semantics is total. Moreover, it does not seem hard in practice to find programs with total denotations. However, it is also easy to construct programs that are operationally total but whose denotational approximate semantics is not total, although such examples tend to be contrived or artificial. Thus, a natural question arises to assess the utility of our approximate semantics: Which

computable functions are definable relatively to our model? That is: Which computable functions f can be represented by a continuous function ϕ in the model such that ϕ is the denotation of some program. We established that all computable total functions $\mathbb{R}^n \to \mathbb{R}$ are definable and that a large class of computable partial functions $\mathbb{R}^n \to \mathbb{R}$ are definable.

8.2 Open questions

8.2.1 Other semantics

An obvious question that comes to the mind is whether there exists a natural domain-based denotational semantics which is both fully adequate ($[\![M]\!] = [M]$ for all programs M of ground types) *and* accounts for the total correctness of programs. For example, it might be possible to combine Marcial-Romero Hoare semantics with our Smyth semantics to achieve this. A simple-minded idea would be to let the denotation of a program to be the pair of these two semantics. A better idea would be to impose additional structure to such a pair, but at present it is not clear to us what such a structure might be.

Another possibility is that a satisfactory solution might require to modify the programming language considered here. As long as the language remains expressive, sequential, and offers a sufficient level of abstraction to work with real numbers, this would also be a satisfactory approach.

It is natural to wonder whether a semantics based on Plotkin power domains would be possible, and whether it would offer further advantages. While this might be possible, we think that this is unlikely. In fact, for both the Hoare semantics and the Smyth semantics, one is implicitly relying on the fact that the elements $\eta(\text{true})$, $\eta(\text{false})$ and $\eta(\text{true}) \cup \eta(\text{false})$ form a connected subset of the corresponding power domain in the Scott topology. In the Hoare power domain, the element $\eta(\text{true}) \cup \eta(\text{false})$ is above $\eta(\text{true})$ and $\eta(\text{false})$ in the information order, and in the Smyth power domain the element $\eta(\text{true}) \cup \eta(\text{false})$ is below $\eta(\text{true})$ and $\eta(\text{false})$ in the information order. However, in the Plotkin power domain, the set $\{\eta(\text{true}), \eta(\text{false}), \eta(\text{true}) \cup \eta(\text{false})\}$ is discretely ordered and hence its relative Scott topology is discrete. Hence any continuous function with values on this three-element set must be constant, and to obtain a non-trivial semantics of rtest in the Plotkin power domain, one has to allow it to have values outside this set, which amounts to allowing it to have a value bottom among its possible outputs, even for total inputs.

8.2.2 Expressivity at higher types

We have given a few examples of programs for higher-type functions (a limit operator and operators to find arguments at which a function is 0 or at which it is positive). But apart from these examples, we have not considered higher-type functions. Two particular second-order operators of interest are the supremum and integration functionals [25] but

at this stage we don't know whether they are definable in our language with respect to our denotational semantics. But notice that the supremum operator with respect to signed-digit representation of real numbers is definable, and that we have used this to obtain the first-order definability results of Chapter 6.

Bibliography

[1] 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.

[2] H.J. Boehm and R. Cartwright. Exact real arithmetic: formulating real numbers as functions. In *Research topics in functional programming*, pages 43–64. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[3] V. Brattka. Recursive characterization of computable real-valued functions and relations. *Theoretical Computer Science*, 162:4577, 1996.

[4] V. Brattka and P. Hertling. Feasible real random access machines. *Journal of Complexity*, 14(4):490–526, 1998.

[5] A. Edalat and F. Rico. Two algorithms for root finding in exact real arithmetic, 1998.

[6] M.H. Escardó. PCF extended with real numbers. *Theoretical Computer Science*, 162(1):79–115, 1996.

[7] M.H. Escardó. *PCF extended with real numbers: a domain-theoretic approach to higher-order exact real number computation.* PhD thesis, Department of Computing, Imperial College, University of London, 1996.

[8] M.H. Escardó. Mathematical foundations of functional programming with real numbers. Four-lecture course in the Midlands Graduate School in the Foundations of Computer Science, Leicester. `http://www.cs.bham.ac.uk/ mhe/papers/mgs.pdf`, 2003.

[9] M.H. Escardó, M. Hofmann, and T. Streicher. On the non-sequential nature of the interval-domain model of real-number computation. *Mathematical Structures in Computer Science*, 14(6):803–814, 2004.

[10] A. Farjudian. *Sequentiality in Real Number Computation*. PhD thesis, School of Computer Science, University of Birmingham, 2004.

[11] P. Di Gianantonio. *A Functional Approach to Computability on Real Numbers*. PhD thesis, University of Pisa, Udine, 1993.

[12] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, and D.S. Scott. *Continuous Lattices and Domains*, volume 93 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2003.

[13] A. Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.

[14] C.A. Gunter and D.S. Scott. Semantic domains. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 633–674. 1990.

[15] R. Heckmann. Power domain constructions. *Science of Computer Programming*, 17(1-3):77–117, 1991.

[16] J.R. Marcial-Romero. *Semantics of a Sequential Language for Exact Real-Number Computation*. PhD thesis, School of Computer Science, University of Birmingham U.K., 2004.

[17] J.R. Marcial-Romero and M.H. Escardó. Semantics of a sequential language for exact real-number computation. *Theoretical Computer Science*, 379(1-2):120–141, June 2007.

[18] E. Moggi. Computational lambda-calculus and monads. In *Proceedings 4th Annual IEEE Symp. on Logic in Computer Science, LICS'89, Pacific Grove, CA, USA, 5–8 June 1989*, pages 14–23. IEEE Computer Society Press, Washington, DC, 1989.

[19] G.D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):225–255, 1977.

[20] G.D. Plotkin. Post-graduate lecture notes in advanced domain theory (incorporating the 'pisa notes'). 1981.

[21] R.Amadio and P.L. Curien. Domains and lambda calculi. In *Cambridge Tracts in Theoretical Computer Science*, volume 46. Cambridge University Press, 1998.

[22] D.S. Scott. Outline of a mathematical theory on computation. In *Proceeding 4th Princeton Conference on Information Science*, 1970.

[23] D.S. Scott. Lattice theory, data type and semantics. In Randall Rustin, editor, *Formal Semantics of Algorithmic Languages*, pages 65–106. Prentice Hall, 1972.

[24] D.S. Scott. Data types as lattices. *SIAM Journal on Computing*, 5(3):522–587, 1976.

[25] A.K. Simpson. *Lazy functional algorithms for exact real functionals*, volume 1450 of *Lecture Notes in Computer Science*, pages 456–464. Springer, Berlin, Heidelber, 1998.

[26] T. Streicher. *Domain-Theoretic Foundations of Functional Programming*. Imperial College Press, London, 2006.

[27] A.M. Turing. On computable numbers, with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society*, 42:230–266, 1936. A correction appears in 43:544–546, same journal and year.

[28] K. Weihrauch. *Computable analysis: an introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.