A thesis submitted to
The University of Birmingham
for the degree of Doctor of Philosophy

Sequentiality in Real Number Computation

Amin Farjudian[1]
School of Computer Science,
University of Birmingham,
United Kingdom

Prof. Achim Jung
Supervisor

Prof. Abbas Edalat
External Examiner

Dr. Martín Escardó
Internal Examiner

August, 2004

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

Contents

List of Tables

List of Figures

Preface

Although for the mainstream computer scientists real numbers are more or less synonymous with floating point numbers, the unreliability of the latter is a well established fact and the need for investment in *exact real number computation* is out of question.

Yet to bring our orthodox colleagues on-board it was deemed helpful to start off with a very gentle introduction to the topic from very scratch, highlighting the difference between major sets of numbers, i. e. natural, integer, rational on one hand and the real and complex numbers on the other hand. This we do in Chapter 1 which also serves as the de facto introductory material, helping us keep this preface rather short. In particular, the experienced reader can safely skip that chapter.

In Chapter 2 we review the basic material which forms the background of what comes in later chapters.

As time goes on, there is more and more evidence emerging pointing towards the widely speculated fact that in exact real number computation, there is a trade-off between efficiency and data abstraction. In Chapter 3 we put an emphasis on this statement by demonstrating the weaknesses of the sequential — i. e. efficient — fragments of Real-PCF, which otherwise scores high on data abstraction.

Learning from the experience we embark on finding a middle-ground between efficiency and data abstraction. Thus in Chapter 4 we present a framework for computation over real numbers which:

1. is sequential.
2. does not allow multi-valuedness.
3. has a good expressive power.

Although there is an inevitable incremental trend in the flow of material, Chapters 3 and 4 can be read independently.

b

Acknowledgements

My thanks go — first and foremost — to my supervisor Professor Achim Jung. He helped me have the opportunity to do PhD under his supervision in the first place, and I benefited from his expertise throughout all the stages of my studies. I am also grateful for his invaluable comments right from the beginning of the writing of this thesis to the end.

I also thank the other members of my thesis group, Prof. Uday Reddy and Dr. Martín Escardó. I thank Martín specifically regarding Corollary 3.2.9 on page 71 as it came out of a comment of his.

Finally, the thesis could not be regarded complete without the assurance of the approvals of the external and the internal examiners — Prof. Abbas Edalat and Dr. Martín Escardó, respectively. Their careful reading of the thesis and precious suggestions were — and definitely will be — heeded.

Part I

Background

Chapter 1

Introduction

Although we use various mathematical concepts and structures to help us towards our goal, the main object of study is the *set of real numbers*, which we denote by the familiar $\mathbb{R}$. As real numbers are numbers after all, perhaps it is worth taking some time understanding where they fit in the whole universe of (at least commonly used) numbers and what their outstanding characteristics are that make them different from other classes of numbers — needless to say that what follows is only an overview of some elementary (yet fairly standard) observations aimed at weaning the unfamiliar reader from the illusion of identifying real numbers as floating point numbers!

1.1 Classes of Numbers

Human beings — or maybe even a bigger collection including most animals — are naturally familiar with some instances of numbers. Take one of the (say) two kitten off its mother and she immediately recognizes there are not enough kitten to feed. Anyway even if a cat can recognize a few instances of numbers, I strongly doubt it can perform any non-trivial computation on them. But we human beings have (long ago) abstracted away from instances of (what is today considered *natural*) numbers and instead used *symbols* such as $1, 2, 3, \ldots$ to represent them. The number 0 (*zero*) was added later on and today, at least in academia, we consider the collection $0, 1, 2, 3, \ldots$ as the *collection of natural numbers*. Furthermore, we do basic arithmetic on these numbers too.

Though they came up more or less naturally (hence the name!), natural numbers are not the only collection of numbers used today, even by nonmathematically-well-educated people. We use different collections of numbers for different purposes. Most of these collections form *sets*, but mathematically speaking, some others are just too big to be a set, therefore they form *class*es. Examples of the latter cases are *ordinal numbers* and *cardinal numbers* which are in turn special cases of ordinal numbers [1].

Here we are more interested in collections of numbers over which we can perform computation and these collections are all sets. We shall see on different occasions that representation is a crucial issue in making computations over a set of numbers practical, but at a lower level, the structure of the set itself is also an important factor. The set of numbers over which we mostly do computation fall under two general categories: *countable* and *uncountable*.

1.1.1 Countable Sets of Numbers: $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$

Take the set of *natural numbers*:

$$\mathbb{N} = \{0, 1, 2, \dots\}$$

and you can easily observe that its elements can be *counted* in an ascending order. It is impossible to count all the elements in finite time, but if one fixes an element beforehand, and starts the counting process, only a finite number of elements have to be passed — in this case, the numbers less than the fixed natural number — before one gets to the fixed element. Based on this intuition, we call $\mathbb{N}$ a *countable set*. It has to be said that being countable has got nothing to do with any structure imposed on the set, like the ascending order we normally impose over $\mathbb{N}$. Even if $\mathbb{N}$ is shuffled to a new arrangement of elements, say:

$$\mathbb{N}' = \{1977, 1356, \dots, 0, \dots, 2, \dots\}$$

it remains a countable set. The reason is simple, we call a set countable if its elements *can be* ordered similar to $\mathbb{N}$ together with the usual ascending order on its elements. The proposed order is only a means via which one can prove the underlying set to be countable. Hence, as one can unshuffle $\mathbb{N}'$ back to $\mathbb{N}$, $\mathbb{N}'$ is countable.

Essentially a countable set is "of the same size" as the set of natural numbers. There are countable sets which to the untrained eye might seem to have a size

[1]For precise definitions of ordinal and cardinal numbers, see any book on (elementary) set theory, e.g. [Enderton, 1977]

different than that of $\mathbb{N}$, simply because there are some elements missing, or extra. A simple example can be presented by the set of integers we describe soon.

Adding or multiplying any two natural numbers results in another natural number, on which account it is said that *the set of natural numbers is closed under addition and multiplication*. But this is not so with subtraction. Taking away 2 from 1 should logically result in something less than 0, which we do not have in $\mathbb{N}$. Therefore we consider the set of *integers*:

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

which *is closed under addition, multiplication and subtraction*.

Any natural number can be found inside $\mathbb{Z}$. On the other hand there are negative numbers present in $\mathbb{Z}$ you do not find in $\mathbb{N}$. But as neither of the size of $\mathbb{N}$ nor that of $\mathbb{Z}$ is finite, they do not need to follow the same rules that finite sets do. In particular, they can be of the same size even though one is a proper subset of the other.

To get an intuitive account of how $\mathbb{Z}$ can be seen to be countable, consider the order on the integers presented in the following figure and the way in which this new order is similar to the ascending order on natural numbers:

Figure 1.1 $\mathbb{Z}$ is countable

$$\begin{array}{ccccccccccccc}
\mathbb{Z} & = & \{ & 0, & -1, & 1, & -2, & 2, & -3, & 3, & \ldots & k, & -(k+1), & \ldots & \} \\
 & & & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & & \uparrow & \uparrow & & \\
\mathbb{N} & = & \{ & 0, & 1, & 2, & 3, & 4, & 5, & 6, & \ldots & 2k, & 2k+1, & \ldots & \}
\end{array}$$

The picture readily indicates why $\mathbb{Z}$ is countable.

The same is true with the set $\mathbb{Q}$ of *rational numbers* consisting of *fractions* of integers. $\mathbb{Q}$ has $\mathbb{Z}$ as one of its proper subsets, and the reason we extend our universe of numbers from $\mathbb{Z}$ to $\mathbb{Q}$ is — similar to the case of going from $\mathbb{N}$ to $\mathbb{Z}$ — the set of integers *not being closed under a much used operation, i. e. division*. $\mathbb{Q}$ is closed under *addition, multiplication, subtraction and division* — excluding, of course, division by zero.

The countability of $\mathbb{Q}$ is not that immediate to see, and a simple picture as in the case of $\mathbb{Z}$ cannot be presented. However, we can try a friendly approach to induce the intuition.

If $p \neq 0$ is an integer, then for any fraction $m/n \in \mathbb{Q}$ we have:

$$\frac{m}{n} = \frac{pm}{pn}$$

For example:

$$\frac{1}{2} = \frac{2}{4} = \frac{3}{6} = \cdots$$

In that regard, we can *normalize* any given fraction m/n to a fraction p/q where:

1. $m/n = p/q$ as rational numbers.

2. p and q are *relatively prime*, i. e. they have no *common divisor* other than 1.

3. $q > 0$.

So, the normalized form of $\frac{3}{6}$ will be $\frac{1}{2}$, and that of $\frac{3}{-6}$ will, of course, be $\frac{-1}{2}$. In the following discussion, by a rational number we mean a *rational number in its normal form*.

Suppose $\mathbb{Q}_n$, $(n \in \mathbb{N})$, is the (finite) set of rational numbers the sum of the absolute value of whose numerator and denominator equals n. As each $\mathbb{Q}_n$ is finite, its elements can be ordered in an ascending order. Now, write down the elements of $\mathbb{Q}_1$ in order, then the elements of $\mathbb{Q}_2$ in order, ..., then the elements of $\mathbb{Q}_n$ in order, ... You will eventually get the following picture:

Figure 1.2 $\mathbb{Q}$ **is countable**

$\mathbb{Q}_1$	$\mathbb{Q}_2$		$\mathbb{Q}_3$				$\cdots$	$\mathbb{Q}_{n+1}$						$\cdots$
$\frac{0}{1}$	$\frac{-1}{1}$	$\frac{1}{1}$	$\frac{-2}{1}$	$\frac{-1}{2}$	$\frac{1}{2}$	$\frac{2}{1}$	$\cdots$	$\frac{-n}{1}$	$\cdots$	$\frac{-1}{n}$	$\frac{1}{n}$	$\cdots$	$\frac{n}{1}$	$\cdots$

Any (normalized) rational number p/q falls in the set $\mathbb{Q}_{|p|+q}$, where $|p|$ is the absolute value of p. Therefore all the rational numbers are listed. Furthermore, as we considered the normal forms, no rational number is repeated in the list. Finally, as each $\mathbb{Q}_n$ is finite, the list does actually resemble natural numbers in ascending order. Perhaps now it is more believable that $\mathbb{Q}$ is in fact countable.

1.1.2 Uncountable Sets of Numbers: $\mathbb{R}$ and $\mathbb{C}$

One might get the impression that as the set $\mathbb{Q}$ of rational numbers is closed under the four basic arithmetic operations — addition, multiplication, subtraction and division — then that is all we need to get on with our lives. But much to our disappointment[2], rational numbers can easily prove to be less than adequate in relatively simple situations. Take a square whose sides are of length 1 meter, and suppose you want to measure the length of its diagonal. In this case rational numbers will not do you any good as according to a simple ancient formula due to the Greek geometrician Pythagoras[3], if d is the length of the diameter in meters, then:

$$d^2 = 2$$

It can easily be shown that no such d exists in the realm of rational numbers. Intuitively one might like to think of rational numbers as having holes between them, and as it happens, the number d above falls exactly on one of these holes.

Perhaps the creative mind of the reader already suggests a system of numbers that includes rational numbers and takes account of the holes, i. e. has no holes in it. This brings up the picture of a line which is — as opposed to that of $\mathbb{Q}$ — continuous. In that respect, one might like to call such line *continuum*. In fact, the standard process of going up from the set $\mathbb{Q}$ of rational numbers to the set $\mathbb{R}$ of *real numbers* is exactly that of *filling the holes*. $\mathbb{R}$ does indeed solve the above mentioned problem, i. e. there exists a real number d such that $d^2 = 2$. Moreover, $\mathbb{R}$ has $\mathbb{Q}$ as one of its proper subsets, and is closed under addition, multiplication, subtraction and division. Accordingly, one can easily be temped to say that $\mathbb{R}$ has all the good qualities of $\mathbb{Q}$, plus some extra, a claim that might seem legitimate through the eyes of a mathematical analyst, but *not so through the eyes of a computer scientist studying real number computation at this point in time*. The reason can simply be expressed as:

> *At this point in time, computation on rational numbers*
> *is much easier than that on real numbers.*

For now, this seems more like a vague feeling rather than a well-established fact, but throughout this thesis, we will have a more in-depth discussion on how a variety of frameworks have failed in one way or another in presenting a theoretically well-behaved and, at the same time, practically efficient method of real number

[2] Greeks were the first to be disappointed by this!

[3] 569–475 BC

computation. This is ultimately a matter of feeling and personal judgement and anyway should be postponed to where all our technical discussions have finished. So let us get back to the very set $\mathbb{R}$ of real numbers.

As mentioned earlier, the set of real numbers can be thought of as representing the points on the *geometric straight line* which forms the intuition behind our informal discussion, assuming some common familiarity with real numbers from (say) high school maths. What is aimed at in this (sub-)section is to highlight one major difference between $\mathbb{R}$ and the other previously mentioned sets of numbers — $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$ — i. e.

The set $\mathbb{R}$ of real numbers is uncountable.

To establish the fact we need to show that real numbers cannot be written down similar to natural numbers in their ascending order. Thus we fix a representation which for our purposes could well be the one known as the *decimal representation* of real numbers. In this setting, each number can be represented as a (possibly infinite) stream of digits, say:

$$d_n \dots d_1 d_0 \,.\, d_{-1} \dots d_{-m} \dots$$

where for all $i \leq n$: $d_i \in \{0, 1, \dots, 9\}$. If the real value of such a stream is denoted by $[\![d_n \dots d_1 d_0 \,.\, d_{-1} \dots d_{-m} \dots]\!]$, that value can be expressed by the following sum:

$$[\![d_n \dots d_1 d_0 \,.\, d_{-1} \dots d_{-m} \dots]\!] = \sum_{i \leq n} d_i \times 10^i \tag{1.1}$$

Next we observe that the set $\mathbb{R}$ can in fact be *partitioned* into two sets $\mathbb{R}_1$ and $\mathbb{R}_2$, such that $\mathbb{R}_1$ consists of those real numbers that have exactly *one* decimal representation, whereas $\mathbb{R}_2$ consists of the ones with exactly *two* different decimal representations. As an example of the latter, take the number "one". There are two ways to represent number "one" as a decimal expansion: either by $1.000\dots$ or by $0.999\dots$ which is simply written as $1.\overline{0}$ and $0.\overline{9}$, respectively. This can be easily verified using Equation (1.1) above.

Notation 1.1.1. *For any character — or in general any pattern of characters — d, $\overline{d}$ means "an infinite stream of the pattern d". In particular, for every digit $d \in \{0, 1, \dots, 9\}$, $\overline{d}$ can be thought of as an abbreviation for $ddd \dots d \dots$.*

In fact, any two different representations of the same element of $\mathbb{R}_2$ follow a similar pattern, i. e. they are a pair of the forms:

- $d_n \dots d_1 d_0 \,.\, d_{-1} \dots d_{-i} d\overline{0}$
- $d_n \dots d_1 d_0 \,.\, d_{-1} \dots d_{-i} (d-1)\overline{9}$

where $i \geq 0$ and $d \in \{1, 2, \dots, 9\}$.

As we are in the middle of an argument regarding a property of the set $\mathbb{R}$, and not the set of the representations of real numbers, we should consider only *one* representation per number. Thus we are faced with an arbitrary choice between the two representations. Let us pick the latter one, i. e. the one ending in an infinite stream of 9's. This way, we have a unique representation for each real number and the following discussion becomes easier.

In order to demonstrate that *the set* $\mathbb{R}$ *is of a strictly greater size than that of* $\mathbb{N}$ *— i. e.* $\mathbb{R}$ *is uncountable* — it suffices to show that the set of real numbers between 0 and 1 — known as the interval $[0, 1]$ — has a greater size than $\mathbb{N}$, as after all $[0, 1]$ is a proper subset of $\mathbb{R}$. Thus we show that *no enumeration of* $[0, 1]$ *by natural numbers exists which exhausts the whole set*. By the choice we made earlier in representation, each real number inside $[0, 1]$ has a representation of the form $0 \,.\, d_{-1} \dots d_{-n} \dots$, i. e. all the digits to the left of the decimal point are zero. Now take any possible enumeration $a_1, a_2, \dots, a_n, \dots$ of the elements of $[0, 1]$ and put them in a vertical arrangement as follows:

$$\begin{aligned}
a_1 &= [\![0 \,.\, d^1_{-1} d^1_{-2} \dots d^1_{-i} \dots]\!] \\
a_2 &= [\![0 \,.\, d^2_{-1} d^2_{-2} \dots d^2_{-i} \dots]\!] \\
&\vdots \\
a_i &= [\![0 \,.\, d^i_{-1} d^i_{-2} \dots d^i_{-i} \dots]\!] \\
&\vdots
\end{aligned}$$

Based on that, define the sequence $(x_{-n})_{n \geq 1}$ as follows:

1. $x_{-n} = 5$ if and only if $d^n_{-n} \neq 5$.
2. $x_{-n} = 2$ if and only if $d^n_{-n} = 5$.

The stream $0 \,.\, x_{-1} x_{-2} \dots$ does not end in an infinite sequence of 0's therefore it represents some real number $x \in [0, 1]$. Yet this stream differs from all a_i's. The reason is simple:

$$\forall n \geq 1 : x_{-n} \neq d^n_{-n}$$

hence for all $n \geq 1$: $x \neq a_n$.

Notice that the argument applies to *any* enumeration as the one we manipulated in our proof was arbitrary. We just showed that $[0, 1]$ is not countable, hence as a result, the whole of $\mathbb{R}$ is not countable.

Similar to what we saw earlier on page 7 regarding the need to extend rational numbers to real numbers, certain situations arise where real numbers are not enough to solve problems expressed in terms of real numbers themselves! Take the following equation for instance:

$$x^2 + 1 = 0$$

It is obvious that for any real number x, we have $x^2 > 0$, therefore $x^2 + 1 > 0$. Thus we need to go beyond the realm of real numbers. Hopefully there exists a system of numbers which:

1. includes real numbers.

2. is closed under the solution to all equations expressed in terms of its own elements.

This new one is known as the set $\mathbb{C}$ of *complex* numbers. $\mathbb{C}$ can naively be described as the set of *pairs of real numbers*, i. e.

$$\mathbb{C} = \{(x, y) \mid x, y \in \mathbb{R}\}$$

containing the following proper subset:

$$\mathbb{C}_{\mathbb{R}} = \{(x, 0) \mid x \in \mathbb{R}\}$$

which "behaves" exactly like $\mathbb{R}$. The counterpart of any real number r in complex numbers is the pair $(r, 0)$, which is of course an element of $\mathbb{C}_{\mathbb{R}}$. For simplicity, we refer to the complex number $(x, 0)$ as x.

Addition over complex numbers is defined pointwise in terms of addition over real components, i. e.

$$(x_1, y_1) + (x_2, y_2) \coloneqq (x_1 + x_2, y_1 + y_2)$$

where + on the right-hand side is addition on real numbers.

Multiplication does not look so natural in the first glance. It is defined as:

$$(x_1, y_1) \,.\, (x_2, y_2) \coloneqq (x_1 x_2 - y_1 y_2, x_1 y_2 + y_1 x_2)$$

where again all the operations on the right-hand side are over real numbers. To get a better intuition, take the complex number $\mathrm{i} = (0, 1)$ and consider i^2:

$$\mathrm{i}^2 = (0, 1) \,.\, (0, 1) = (-1, 0)$$

As $(-1, 0)$ is in fact the real number -1, we have:

$$\mathrm{i}^2 + 1 = 0 \tag{1.2}$$

Each complex number can be re-written as:

$$(x, y) = (x, 0) + (0, y) = (x, 0) + (0, 1)(y, 0) = x + \mathrm{i}y \tag{1.3}$$

Based on Equations (1.2) and (1.3) above one can rewrite multiplication over complex numbers in terms of the new representation as:

$$\begin{aligned}
(x_1, y_1) \,.\, (x_2, y_2) &= (x_1 + \mathrm{i}y_1) \,.\, (x_2 + \mathrm{i}y_2) \\
&= (x_1x_2) \;+\; (x_1\mathrm{i}y_2) \;+\; (\mathrm{i}y_1x_2) \;+\; (\mathrm{i}^2y_1y_2) \\
&= (x_1x_2 - y_1y_2) + \mathrm{i}(x_1y_2 + y_1x_2) \\
&= (x_1x_2 - y_1y_2, x_1y_2 + y_1x_2)
\end{aligned}$$

We do not go into any further details on complex numbers and suffice it to say that *arithmetic operations on complex numbers reduce to those on real numbers*. We will not explicitly mention anything about complex numbers, nevertheless by the tight connection between complex and real number computation our improvements in the latter directly affect that of the former. Finally, it is easy to see that $\mathbb{C}$ is also uncountable, because it has a set the size of $\mathbb{R}$ as one of its proper subsets.

1.1.3 Summary

So far we have tried to present a gentle introduction to familiar sets of numbers in general, and highlight the uncountability of real numbers. Of course, one needs to study real numbers much further than just that of their uncountability in order to get a better grip on what goes on in their computation. Nonetheless even the very humble uncountability fact already excludes the majority of real numbers from being computable as any set of computable objects has to be countable. This need not necessarily worry us however, as the much used real numbers — such as e and π — are computable.

A final word on the subject of the sizes of different sets — otherwise known as their *cardinalities* — we have mentioned so far. We just about managed to show that $\mathbb{R}$ is strictly bigger than $\mathbb{N}$. The exact relation between the cardinalities of $\mathbb{R}$ and $\mathbb{N}$ — the *continuum hypothesis* — was an open problem for a long while, until after the publication of the mathematically-elegant works of Gödel [1938, 1940a,b] and Cohen [1963, 1964] it was established that in fact the relation is independent of the axioms of mathematics — otherwise known as Zermelo-Fränkel axioms of set theory — we often use. But as far as we are concerned, under any setting we work in, real numbers are "uncountable".

1.2 Computation on Countable Sets of Numbers

Consider the following two scenarios:

1. You are about to start teaching a course and you are given a list of names of those who are going to attend. In as much as preparations are concerned, perhaps the number of students is enough. There is usually more information — such as students' ID's — that can give you an indication of, say, how many of them are taking the course for the second time, etc. Anyway, *the real people, i. e. students, are there, living in the world*. You have to wait to see them, and probably wait for a few sessions to pass, before you get a good grip of what to say and how to adjust the speed and the amount of material in your lectures. But apart from routine human interaction and the normal impression we have on each other, there is not much you can do to change those real people according to your expectations. You are not able to manufacture a number of students towards your own goals.

2. Now consider another story. You want to have a computer to help you do some of your work. Let us assume you are the kind of person who first makes a list of the components — either on a paper, or at least in your mind — consisting of essential parts, such as motherboard, CPU, etc. Then you might add a few non-essential parts, such as scanner, modem, etc. When you are making your own list, you might think of specific brands as well, but in any case, *the parts will be chosen later on, according to your needs, and perhaps some restrictions as well*, say, price, availability, etc.

When dealing with numbers, there are more or less two general approaches similar to the above scenarios, that are of interest to us here. The first approach, usually referred to as *Platonic* approach, considers symbols $0, 1, 2, \ldots$ as names referring to objects that do exist by themselves, somewhere there, and it is our job to find them out. This resembles the scenario 1 on the preceding page.

Another way of dealing with symbols $0, 1, 2, \ldots$ is to actually think of them as names waiting for objects to refer to, much like scenario 2 on the facing page. These objects are going to be manufactured according to our needs, the specific framework we work in, and — perhaps most important — *a list of axioms (that comes with any set of numbers) they have to satisfy*.

We tend to have the non-Platonic mentality thus frequently build up *models* for different sets of numbers which suit our purposes as folks doing computer science.

1.2.1 Natural Numbers

Let us begin with $\mathbb{N}$ as a set of symbols. As discussed in subsection 1.1.1, the set of natural numbers stands more or less as the standard reference countable set. Another characteristic property is that *it consists exactly of the cardinalities (the number of elements) of finite sets*. In other words, we use natural numbers in counting the number of objects in finite collections, a fact which necessitates the existence of a *unit* out of which natural numbers are made up. This is easily verified by taking number 1 as the unit, and then observing that any natural number n can be thought of as "n number of units".

Natural numbers $0, 1, 2, \ldots$ as a collection of symbols cannot serve everybody's purposes and there are times when one needs to "construct" one's own model of natural numbers. Regardless of the setting in which we work, in order to construct a model of $\mathbb{N}$ — which for now we write as $[\![\mathbb{N}]\!]$ — and generate the natural numbers (say) one-by-one, all we need is the following two:

1. A place to start, i. e. *zero*.

2. An operation which given a natural number n, increments it one unit and returns $n + 1$, i. e. the *successor* operation.

Even in the rigorous set theoretical modelling of $\mathbb{N}$, we usually tend to use the same idea. There we know that the universe consists entirely of sets — which in itself can be viewed as a kind of restriction — hence $[\![\mathbb{N}]\!]$ as well as each of its elements must turn out to be sets which after all, have to satisfy a number of axioms characterizing natural numbers. One can take the empty set $\emptyset$ as zero, and then define the successor operation S defined over the universe of all sets as:

$$S(x) = x \cup \{x\}$$

Finally we stipulate that $[\![\mathbb{N}]\!]$ is the smallest set containing zero — in this case, $\emptyset$ — which is closed under the successor operation — in this case, S.

In computer science, more specifically in the design of programming languages, people have implemented different ideas regarding how to model various sets of numbers. Here the actual building blocks are not necessarily pure sets. But as an example, the only one which gives the best (potential) model of $\mathbb{N}$, uses the same basic idea as in set theory, i. e. the zero and successor approach.

In the functional programming language Haskell, there are two data types used for integers: Int and Integer. The former is modelled according to tradition. Therefore, Int essentially works like the Int data type in Java, C, Pascal, etc. One has to bear in mind that although Int is supposed to be used as a surrogate for $\mathbb{Z}$, mathematically speaking, it does not even come close to being a true representative of the set of integers.

Notation 1.2.1. *For any syntactic term M, we denote its mathematical meaning by $[\![M]\!]$.*

Whereas Int is supposed to simulate $\mathbb{Z}$, $[\![\mathsf{Int}]\!]$ is merely a finite set consisting of fixed length binary strings of 0's and 1's. Of course, looking on the bright side, the computation is fast. So if all the numbers involved throughout a computation process fall within the legitimate range, there seems to be no problem. In fact, this kind of implementation has been used for a long time. The reason is that elements of data type Int are not merely used for heavy computations. While writing a program in (say) Java, one might like to use a variable as a switch, i. e. a variable whose value alternates between two fixed numbers — say zero and one, or one might like to index an array. But once you want to get involved in computations where you cannot even predict the range of values that might arise during the process, a limited set like $[\![\mathsf{Int}]\!]$ can cause irritating errors.

The data type Integer is implemented in a different way. $[\![\mathsf{Integer}]\!]$ is meant to resemble the set of integers, at least as far as the computer resources allow. To get a better idea, let us start with a data type Nat for natural numbers implemented in Haskell as follows[4]:

[4]For a more comprehensive discussion of implementing Nat and Integer inside Haskell, see [Bird, 1998, Chapter 3].

```
data Nat = Zero | Succ Nat
```

We do not really want to get involved with the details of the above definition and what kind of object Nat is in a technical sense. Instead we emphasize the presence of Zero and Succ. If we postulate $[\![\mathbb{N}]\!] := \mathsf{Nat}$, then we can have:

$$
\begin{aligned}
[\![0]\!] &= \mathsf{Zero} \\
[\![1]\!] &= \mathsf{Succ\ Zero} \\
[\![2]\!] &= \mathsf{Succ\ (Succ\ Zero)} \\
&\vdots
\end{aligned}
$$

This implementation does not have the problem of boundedness[5] and visibly resembles natural numbers. Arithmetic operations are easy to implement using pattern matching. As an example, consider addition:

```
(+) :: Nat -> Nat -> Nat
m + Zero   = m
m + Succ n = Succ (m + n)
```

Perhaps it is now easier to understand the following code (taken exactly as it appears in [Bird, 1998, page 75]) implementing the data type Integer in Haskell, based on more or less the same idea as that of implementing Nat. Here Integer is partitioned into three subsets: positive integers, negative integers and zero. We need to implement the positive ones (much as the way we did with Nat) and define the negatives in terms of the positive ones:

```
data Integer  = Neg Positive | Zero | Pos Positive
data Positive = One | Succ Positive
```

There is a dichotomy present between Int and Integer that can even suggest a more general statement in much of the work we do throughout the thesis. Integer arithmetic is exact (correct), whereas that of Int is not. On the other hand, Int uses the machine's fast built-in arithmetic unit, against the Integer's (potentially) slow symbolic arithmetic. All in all, this is a special case of *the trade-off between efficiency and correctness*.

[5] worth mentioning again, *as far as computer resources allow*.

1.2.2 Rational Numbers

In case one wants to implement rational numbers symbolically, the definition and arithmetic operations reduce to that of integers.

Notation 1.2.2 (gcd). *By* $\mathrm{gcd}(m, n)$ *we mean the* greatest common divisor *of m and n, where (m, n) is a pair of integers other than* $(0, 0)$.

Definition 1.2.3 (rational numbers: $\mathbb{Q}$). *The set $\mathbb{Q}$ of rational numbers is defined by:*

$$\mathbb{Q} := \{m/n \mid m, n \in \mathbb{Z},\ n > 0,\ \mathrm{gcd}(m, n) = 1\}$$

Rational numbers as defined above are said to be in *normal* form. Of course, this is a slightly strict condition we impose on the rational numbers to keep things neat. We bear in mind that the result of any computation must in turn be in normal form. Hence, to facilitate the discussion, we define:

Notation 1.2.4 (relative complement (set difference): $A \setminus B$). *For any two sets A and B we denote the* relative complement of B in A *by $A \setminus B$, i. e.*

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

Definition 1.2.5 (rational normalization operator: RNO). *The rational normalization operator* $\mathrm{RNO}\colon \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \to \mathbb{N}$ *is defined by:*

$$\mathrm{RNO}(m, n) := (r, s)$$

where:

1. $s > 0$
2. $rn = sm$
3. $\mathrm{gcd}(r, s) = 1$

Using RNO, addition $+_{\mathbb{Q}} : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$ on rational numbers can be defined as follows:

$$\forall\ (m/n),\ (p/q) \in \mathbb{Q}\colon (m/n) +_{\mathbb{Q}} (p/q) = \mathrm{RNO}((mq + np)/(nq))$$

Other arithmetic operations over rational numbers reduce to those of integers too — in our case, followed by a normalization. This is more or less the way rational numbers have been implemented as the data type Rational in Haskell. Unfortunately, in most other languages one cannot see a proper data type for rational numbers. Programmers tend to use types such as Float or Double instead. These types are normally implemented on the basis of the machine's built-in *floating point arithmetic unit* (to be discussed in the following section) which results in (relatively) *fast but error-prone* computations.

1.3 Computation on Real Numbers

1.3.1 Floating Point Representation

Measuring the diagonal of a square — especially that with sides of length 1 meter [6] — might not fit into everyone's daily timetable. Nevertheless, real numbers turn out to be an inevitable set of numbers, at least for mathematicians, physicists and many engineers. For that matter, programming language designers had long ago included at least one data type for real numbers in most commonly known languages such as Pascal, C, Java, etc. One major difference between the real number data types and that of the integers is that the former has almost *always* [7] been implemented based on the *machine's built-in floating point arithmetic unit*, even in Java and Haskell where we have a (more or less) symbolic (next-to-perfect) implementation of integers. It is helpful to mention in passing that some machines have not come with a physical built-in floating point arithmetic unit as such, nevertheless a kind of software simulation has done the job quite efficiently.

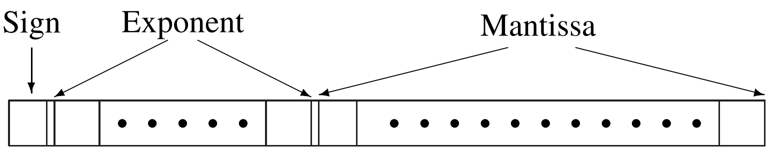
Anyway, the point is that these implementations are always *inaccurate*. We let $[\![\mathbb{R}]\!]_{\mathrm{f}}$ — subscript f for **f**loating point — be the model of real numbers used in a typical implementation of this kind. On the bright side, computation on $[\![\mathbb{R}]\!]_{\mathrm{f}}$ is much faster than any symbolic computation on real numbers, at least as of today. On the other hand, $[\![\mathbb{R}]\!]_{\mathrm{f}}$ is so far away from being similar to $\mathbb{R}$ that it easily leads to wrong computations. The reason is that $[\![\mathbb{R}]\!]_{\mathrm{f}}$ is in fact a *finite subset of rational numbers*, represented by a finite set of binary strings of fixed length — typically $16, 32, 64, \ldots$. Any such string is (theoretically) partitioned into three groups of bits, as in Figure 1.3 on the following page. The actual (rational) value represented by one such string can be calculated as:

$$v = s \times m \times 2^{e}$$

[6] see page 7

[7] at least to my knowledge...

Figure 1.3 A Sketch of Hardware Floating-point representation

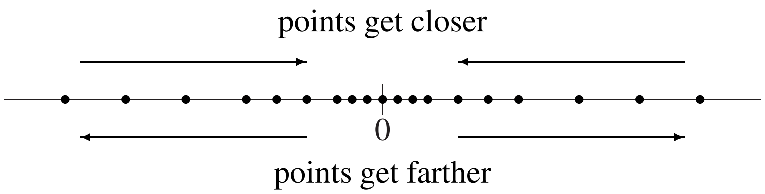


where:

- s is the content value of Sign — either 1 or -1. [8]
- m is the content value of Mantissa.
- e is the content value of Exponent.

As the length of all strings are fixed and equal, the maximum number of significant digits of the represented numbers is a fixed finite number. Therefore the way $[\![\mathbb{R}]\!]_{\mathrm{f}}$ is distributed over the real line looks more or less like the following figure. The

Figure 1.4 Distribution of Floating-points over the Real-line

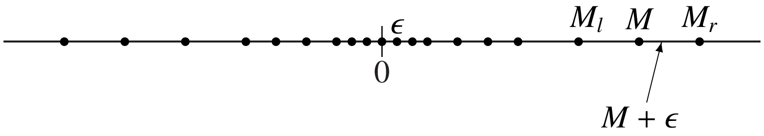


points get further apart as one moves away from 0, which causes a (somewhat) fundamental mistake in computations. For instance, take a big number $M > 0$ and a relatively small one $\epsilon > 0$ from $[\![\mathbb{R}]\!]_{\mathrm{f}}$, such that the distances between M and its immediate left and write neighbours — say, M_l and M_r — are both more than ϵ (Figure 1.5 on the facing page). This leads to:

$$M + \epsilon = M$$

[8] The Sign bit is either 1 or 0, but for our purposes interpreted as 1 or -1.

Figure 1.5 Floating-point representation is unreliable



which is wrong as $\epsilon \neq 0$. Hence in a sense, *the unit of addition is not unique*. For instance, the division

$$\frac{\epsilon}{(M+\epsilon)-M}$$

would lead to a *division by zero*, whereas the innocent programmer is waiting for a simple 1. Of course this is not the only problem that can arise, otherwise one could argue that:

1. If the range of the computation is somehow known beforehand, we adjust the precision accordingly — i. e. resort to longer strings if necessary — to approximate real numbers by elements of $[\![\mathbb{R}]\!]_{\mathrm{f}}$.

2. If the intermediate computations lead to minor errors, we can still use $[\![\mathbb{R}]\!]_{\mathrm{f}}$ on account of its fast computations.

Unfortunately the so-called *round-off* errors — the difference between the ideal real number and the floating point approximating it — can accumulate in unexpected ways, as Ménissier-Morain [1996] demonstrated in detail via two "striking" examples.

Being unsatisfied with this sort of intrinsic *unreliability* of the floating-point representation, new ways of implementing real number computation have been put forward. Skipping further cumbersome discussions, we explicitly demand *effectiveness* as well, a property that (for now) can informally be thought of as *recursive computability*. Accordingly the realm of *exact real number computation* has been opened up based on the following two major criteria:

1. It must be reliable, i. e. the output produced must be guaranteed to be correct.

2. The results must be *effectively* computable — e. g. as opposed to the BSS approach [Blum, Shub, and Smale, 1989] — to within any desired degree of accuracy.

1.3.2 Cauchy Sequence Representation

Starting from natural numbers and going up the hierarchy of sets of numbers, one needs to take a much more cautious pace at or around the place where real numbers are to emerge. This is the point where various schools of mathematics diverge. As mentioned earlier on page 13, we are in the business of constructing models according to a set of axioms and rules on the one hand and our needs on the other hand. In the case of real numbers, the set of axioms and rules depend on which logic one works in, which in itself affects our needs as well. Although we do not need the full strength of classical mathematics — in fact, at times it does not match our framework — still it is safe enough to present the Cauchy sequence definition of real numbers in its classical sense.

Definition 1.3.1 (Sequence). *By a sequence s over a set X we mean a function $s\colon \mathbb{N} \to X$. Often for simplicity, we denote the n-th ($n \geq 0$) element of the sequence by s_n rather than $s(n)$.*

A sequence s over rational numbers is called *Cauchy* if its elements get closer and closer (as the index grows), i. e.

Definition 1.3.2 (Cauchy Sequence over Rational Numbers). $s\colon \mathbb{N} \to \mathbb{Q}$ *is* Cauchy *iff:*

$$\forall p \in \mathbb{N}\ \exists n_0 \in \mathbb{N}\ \forall m, n \geq n_0 \colon |s_m - s_n| < \frac{1}{2^p}$$

Cauchy sequences are defined over *metric spaces*[9] in general. Once the metric space satisfies certain conditions Cauchy sequences coincide with *convergent* sequences (see the definition on the facing page). Let us mention right here that the space of rational numbers with the metric

$$d(p, q) := |p - q|$$

does not satisfy those conditions.

[9]See [Rudin, 1976] for the definition of metric spaces.

Definition 1.3.3 (convergent sequence). *A sequence* $s\colon \mathbb{N} \to \mathbb{R}$ converges *to a point* $r \in \mathbb{R}$ *iff:*

$$\forall p \geq 0\ \exists n_0 \in \mathbb{N}\ \forall n \geq n_0 : \ |s_n - r| < \frac{1}{2^p} \tag{1.4}$$

$s\colon \mathbb{N} \to \mathbb{R}$ *is* convergent *iff it converges to some point* $r \in \mathbb{R}$*, otherwise it is* divergent. *In case s converges to* $r \in \mathbb{R}$*, r is called the* limit *of s, written* $\lim_{n\to\infty} s_n = r$.

Over real numbers, Cauchy sequences are the same as convergent ones. This is not the case with rational numbers. The reason is that a Cauchy sequence over the rational numbers can converge to an irrational real number, and it must be mentioned that the limit of a convergent sequence is unique.

The good news is that for any real number r — no matter rational or irrational — there exists a sequence[10] over rational numbers converging to r. For each such r, any sequence over rational numbers converging to it can be chosen as a representative for r. Now going back to the case when all we had was $\mathbb{Q}$ and we were yet to construct $\mathbb{R}$, we use a standard method called *completion* in order to get real numbers out of rational ones.

First consider the following three examples of sequences over $\mathbb{Q}$:

$$\begin{array}{ccc} s_1 : \mathbb{N} \to \mathbb{Q} & s_2 : \mathbb{N} \to \mathbb{Q} & s_3 : \mathbb{N} \to \mathbb{Q} \\ n \mapsto 0 & n \mapsto \dfrac{1}{n+1} & n \mapsto \dfrac{-1}{n+1} \end{array}$$

All three sequences are Cauchy — in fact convergent — over $\mathbb{Q}$. It is not hard to verify that all three sequences converge to 0, using Definition 1.3.3 above. The morale is that different sequences can converge to the same real number. In fact, for any real number $r \in \mathbb{R}$, there are infinitely many sequences (over $\mathbb{Q}$) converging to r. Our next task is to put all such sequences into the same group. We need to be cautious not to mention anything about their common limit as it can well be an irrational number and we are supposed not to have them yet.

Fortunately there is a fairly intuitive relation between sequences converging to the same point, that can be expressed entirely in terms of the elements of the sequences themselves rather than referring to the limit. Let us observe the relation by ourselves:

Assume $s_1, s_2 : \mathbb{N} \to \mathbb{Q}$ converge to the same limit (say) $r \in \mathbb{R}$. We have:

$$\forall p \in \mathbb{N} : \begin{cases} \exists n_0 \in \mathbb{N}\ \forall n \geq n_0 : \ |s_1(n) - r| < 2^{-p} \\ \exists n_1 \in \mathbb{N}\ \forall n \geq n_1 : \ |s_2(n) - r| < 2^{-p} \end{cases} \tag{1.5}$$

[10] in fact infinitely many.

By the *triangle inequality* we have:

$$\begin{aligned}\forall n \geq \max(n_0, n_1)\colon\ |s_1(n) - s_2(n)| &= |(s_1(n) - r) + (r - s_2(n))| \\ \text{(triangle inequality)} &\leq |s_1(n) - r| + |s_2(n) - r| \\ \text{(Equation (1.5) on the preceding page)} &< \frac{1}{2^p} + \frac{1}{2^p} \\ &= \frac{1}{2^{p-1}}\end{aligned}$$

As p in Equation (1.5) on the page before was arbitrary, we can simply claim:

$$\forall p \in \mathbb{N}\ \exists m \in \mathbb{N}\ \forall n \geq m\colon\ |s_1(n) - s_2(n)| < \frac{1}{2^p} \tag{1.6}$$

In words, Equation (1.6) above tells us that as n grows, the terms of both sequences get closer and closer which matches the intuition as after all, they are approaching r, hence each other too.

All in all, we need to declare such pairs of sequences as being *equivalent*. For that, we use *equivalence relations*:

Definition 1.3.4 (equivalence relation). *A binary relation $R \subseteq X \times X$ over a set X is an* equivalence relation *iff the following three conditions are held:*

1. *reflexivity:* $\forall x \in X\colon\ x\,R\,x$
2. *symmetry:* $\forall x, y \in X\colon\ (x\,R\,y) \Rightarrow (y\,R\,x)$
3. *transitivity:* $\forall x, y, z \in X\colon\ (x\,R\,y) \wedge (y\,R\,z)\ \Rightarrow\ (x\,R\,z)$

Note 1.3.5. *Fixing an element $x \in X$, the set $\{y \in X \mid xRy\}$ is called the* equivalence class of *x, which we write as $[x]_R$. Following the three clauses of the definition above, it is not difficult to show that:*

$$\forall y \in [x]_R\colon\ [x]_R = [y]_R$$

The set of all equivalence classes of X over R — written as X/R — is called the quotient of *X* by *R.*

Now let $\mathrm{CS}_{\mathbb{Q}}$ be the set of all **C**auchy **S**equences over $\mathbb{Q}$. We define a binary relation $\sim$ over $\mathrm{CS}_{\mathbb{Q}}$ by:

$$\forall s_1, s_2 \in \mathrm{CS}_{\mathbb{Q}}\colon$$
$$s_1 \sim s_2 \quad\Longleftrightarrow\quad \forall p \in \mathbb{N}\ \exists n_0 \in \mathbb{N}\ \forall n \geq n_0\colon |s_1(n) - s_2(n)| < \frac{1}{2^p}$$

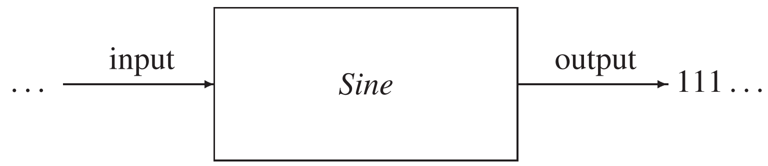
Each equivalence class of $\mathrm{CS}_{\mathbb{Q}}$ under $\sim$ — i. e. any element of $\mathrm{CS}_{\mathbb{Q}}/\sim$ — consists of a set of sequences converging to the same point in $\mathbb{R}$. Although the objects we are dealing with look somewhat bizarre, what we really have in mind is the common point to which their terms converge. In that regard, it is perfectly legitimate to hope that $\mathrm{CS}_{\mathbb{Q}}/\sim$ behaves like $\mathbb{R}$. Fortunately this is the case, though we do not bother to get more involved with the details of that. Nevertheless, we shall not be complacent as $\mathrm{CS}_{\mathbb{Q}}$ has its own deficiencies too. The reason is that we need a model of real numbers *suitable for computation*. As theoretically perfect as $\mathrm{CS}_{\mathbb{Q}}$ may sound, in practice computations over it would lack a fundamental feature — so fundamental that often goes without saying — that for our purposes needs a bit of clarification.

A fair expectation from any reasonable computation process is that it *terminate in finite time*. The presence or absence of this simple requirement can easily change the whole theory of computation. When dealing with natural numbers, a computation process terminates if and only if the final result in terms of natural numbers is produced as a whole. Real numbers do not have that well-behaved feature. Technically we say that *real numbers are not of finite nature*. Of course here by real numbers we mean the ones modelled properly, not the likes of floating point numbers. Cauchy sequences present a nice example. A sequence is an infinite object, never to be exhausted in finite time. In that respect computations over real numbers do not terminate in finite time as well. Instead, we demand the computation to proceed until the result is to within a satisfactory degree of precision. This can be stipulated as a termination condition.

As an example suppose we design a setting in which Cauchy sequences are used to represent real numbers, and furthermore it is possible to get hold of an implementation of some familiar operation, (say) the *sine* function, given to us as a black-box. We need to enter the successive terms of real numbers, i. e. Cauchy sequences representing them, and from the output channel the terms of the result are sent out. Well, the user knows what he is entering, i. e. in practice he knows to which real number the input sequence is converging. On the other hand, from the output channel a very simple sequence (say) of 1's is produced (the figure on the following page). What the black-box does not tell us is how close each output term is to the ideal result. Consequently, the user cannot even be sure if the limit is 1! Assuming the implementation of *sine* to be correct, the output sequence will *eventually* converge to the correct result, but in the meantime — finite time that is — it is not at all clear to the user what the result will be.

The problem is that with Cauchy sequences in their classical form and shape, one never knows how close a term is to the (ideal) limit just knowing its index.

Figure 1.6 Unsuitability of Classical Cauchy sequence



The way to avoid this problem is to explicitly require some upper bound of the distance between the potential limit and any term of the sequence to be known in terms of its index.

Notation 1.3.6 (dtl). *By* dtl *we mean any strictly increasing recursive function:*

$$\mathrm{dtl}\colon \mathbb{N} \to \mathbb{N} \setminus \{0\}$$

where dtl *stands for* ***d****istance* ***t****o* ***l****imit.*

Now we can redefine Cauchy sequences as:

Definition 1.3.7 (Constructive Cauchy Sequences). *A sequence $s\colon \mathbb{N} \to \mathbb{Q}$ is called* constructively Cauchy relative to dtl *iff:*

$$\forall m, n \in \mathbb{N}\colon\ m \geq n \Rightarrow |s_m - s_n| < \frac{1}{\mathrm{dtl}(n)}$$

Perhaps the simplest form dtl can take is:

$$\forall n \in \mathbb{N}\colon \mathrm{dtl}(n) = n + 1$$

which was used by Bishop and Bridges [1985]. Of course we will develop other ways of representing real numbers, but this idea of an already known distance to limit of every term remains more or less visible and essential throughout.

Representations Suitable for Computation

Consider the famous decimal representation of real numbers in which real numbers are represented by infinite streams of digits:

$$d = d_m \ldots d_1 d_0 \,.\, d_{-1} d_{-2} \ldots d_{-n} \ldots$$

where $\forall i \leq m\colon d_i \in \{0, \ldots, 9\}$, and "." is called the *decimal* point. The real number represented by such a stream — which we write as $[\![d]\!]$ — can be calculated by:

$$[\![d]\!] = \sum_{i \leq m} d_i 10^i$$

Even this representation is based on Cauchy sequences, relative to:

$$\begin{aligned} \mathrm{dtl}_{\mathrm{dec}}\colon \mathbb{N} &\to \mathbb{N} \setminus \{0\} \\ n &\mapsto 10^{-n} \end{aligned}$$

For that matter, consider the sequence $s\colon \mathbb{N} \to \mathbb{Q}$ defined by:

$$\forall i \in \mathbb{N}\colon\ s_i = \sum_{j=0}^{m} d_j 10^j + \sum_{j=1}^{i} d_{-j} 10^{-j}$$

This might tempt us to think that decimal representation does satisfy all the previously mentioned criteria in order to become the handy choice for real number representation. However, a more thorough analysis reveals that a setting based on this representation cannot include some very basic functions, the existence of which in any reasonable library is not that much of a high expectation.

The deficiency can be demonstrated by the simple classic example of:

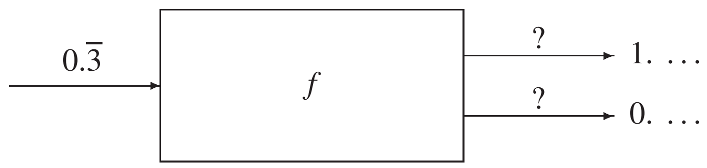
$$\begin{aligned} f\colon \mathbb{R} &\to \mathbb{R} \\ x &\mapsto 3x \end{aligned}$$

Any reasonable library of computable functions over real numbers should at least include "multiplication by 3". But it turns out that it is impossible to implement f in a decimal setting. Assume for the moment that an implementation of f is given to us as a black box, with an input channel receiving incoming digits one-by-one, and of course an output channel emitting digits of the result one-by-one. Let us look at the behaviour of the black box over the input "one-third" represented as "$0.\overline{3}$", according to Notation 1.1.1 on page 8 (Figure 1.7 on the next page). At any (*finite*) moment in time, all f can see is a (*finite*) initial segment of $0\,.\,\overline{3}$, say:

$$0\,.\,\underbrace{3333\ldots333}_{n}$$

From f's point of view even the very next digit can be anything. This prevents f from being able to output even the first digit — immediately to the left of the

Figure 1.7 Decimal representation is not suitable for computation



decimal point, which must either be 0 or 1. The reason is that if the $(n + 1)$-st input digit is greater than 3, i. e. $4, 5, \ldots, 9$, the output will be $1 . \cdots$, otherwise — in case $(n+1)$-st digit is $0, 1, 2$ or 3 —it has to be $0 . \cdots$. In summary, f cannot produce any output digit over the input stream $0 . \overline{3}$ — needless to emphasize *in finite time*.

The root of the problem lies in the representation. Fortunately other classes of representations have been put forward that are suitable for computation. One famous example is the so-called "*signed-digit binary*" representation, in which real numbers are represented by infinite streams of digits of the form:

$$d = d_m \ldots d_1 d_0 . d_{-1} d_{-2} \ldots d_{-n} \ldots$$

where:

$$\forall i \leq m: \ d_i \in \{-1, 0, 1\}$$

The real number represented by d can be calculated by:

$$[\![d]\!] = \sum_{i \leq m} d_i 2^i$$

In this setting, any real number has "infinitely many" different representations. The class of computable functions under this representation includes almost all commonly used functions such as arithmetic operations, sine, logarithm, etc. Signed-digit binary representation is a special instance of a larger class of representations known as *admissible*. We do not get into the details of this here and refer the reader to [Weihrauch, 2000] for a precise definition.

1.4 Summary

In this chapter we tried to introduce real numbers and their position relative to a few other sets of numbers, together with a light touch on the computations involving real numbers. No advanced material was presented throughout; however, for the reader less familiar with the subject, I hope that a general intuition regarding the need for exact real number computation has been attained.

Chapter 2

Technical Background

2.1 λ-calculus

In [Cutland, 1980] an abstract *semantic* formalization of the concept of *computability* has been studied through the (imaginary) **U**nlimited **R**egister **M**achine, URM in short. In the present section, we take the opportunity to get familiar with an abstract *syntactic* formalization of that concept via *λ-calculus*. Of course our primary concern is to fix a few basic definitions. A standard textbook on λ-calculus is [Barendregt, 1984].

Let us start by introducing the language of the λ-calculus. The terms of the syntax are called *λ-terms*. Their construction is fairly simple.

Definition 2.1.1 (λ-term). *The set of* λ-terms*, denoted by* Λ*, is the smallest set satisfying the following properties:*

1. $\forall x \in \mathrm{Var} : \; x \in \Lambda$ *where* Var *is a countable set of* variables.
2. $\forall M, N \in \Lambda : \; \mathrm{appl}(M, N) \in \Lambda$ *where* $\mathrm{appl}\colon \Lambda \times \Lambda \to \Lambda$ *is called* the application function.
3. $\forall M \in \Lambda,\; x \in \mathrm{Var} : \; (\lambda x \,.\, M) \in \Lambda$ *where the* λ *operator is called the* abstraction operator *and terms of the form* $\lambda x.M$ *are called* abstractions.

We let $x, y, z, \ldots$ range over variables and $M, N, P, \ldots$ range over λ-terms. Also, it is easier to use the infix variant of the appl function and write $(M \,.\, N)$, even (MN), instead of $\mathrm{appl}(M, N)$. When a number of abstractions occur consecutively, we use one λ symbol, e. g. instead of $\lambda x \,.\, (\lambda y \,.\, (\lambda z \,.\, M))$ we simply write $\lambda xyz \,.\, M$.

As the definition indicates, a λ-term can be parsed down to its atomic elements. Also, as these terms are constructed recursively, structural induction can be used to prove properties of the terms, and structural recursion can be used to define functions over terms. As an example we define the set of *free variables* of a λ-term:

Definition 2.1.2 (free variables). FV(M)*, the set of* free variables *of a λ-term M is defined recursively by:*

$$
\begin{array}{lcll}
\mathrm{FV}(x) & = & \{x\}, & (x \text{ a variable}) \\
\mathrm{FV}(M\,.\,N) & = & \mathrm{FV}(M) \cup \mathrm{FV}(N) & \\
\mathrm{FV}(\lambda x\,.\,M) & = & \mathrm{FV}(M) \setminus \{x\} &
\end{array}
$$

where $\setminus$ *denotes* set difference *(Notation 1.2.4 on page 16).*

Essentially terms are made up of variables, and during the construction of a term, variables remain free unless abstractions *bind* them, as can be seen in the above definition.

Of course the principal reason behind introducing the λ-calculus here is because all languages we discuss later on throughout the thesis are extensions of a certain variant of λ-calculus, called *simply-typed λ-calculus*, the terms of which are classified into different categories called types.

Definition 2.1.3 (Types). *The set* $\mathbb{T}_\lambda$ *of* types of simply-typed λ-calculus *is generated by the grammar:*

$$\sigma ::= x \mid \sigma \to \sigma$$

where x ranges over type variables.

Definition 2.1.4 (type assignment). *By a* type assignment *we mean any function* $\rho\colon \mathrm{Var} \to \mathbb{T}_\lambda$*. We let ρ range over type assignments.*

Variables need to be assigned their types manually, that is the reason we introduce the definition above. Once this is done, inferring types of other terms follows a set of rules. So, we define a binary relation $\vdash$ between type assignments and terms with types to assert $\rho \vdash M\colon \sigma$, which is read as "according to the type assignment ρ, M is of type σ". Fixing one such ρ, $\vdash$ is defined as:

1. variable : $\rho \vdash x\colon \sigma$ iff $\sigma = \rho(x)$

2. application : $\rho \vdash (M\,.\,N)\colon \tau$ iff $\rho \vdash M\colon \sigma \to \tau, \rho \vdash N\colon \sigma$ for some $\sigma \in \mathbb{T}_\lambda$

3. abstraction : $\rho \vdash \lambda x\,.\,M \colon \sigma \to \tau$ if $\rho \vdash x \colon \sigma, \rho \vdash M \colon \tau$

It is a tradition to explicitly denote the type of the variable x in an abstraction and write it as $\lambda x \colon \sigma\,.\,M$.

By now, still the degree of abstraction exceeds our expectations according to ordinary computation. For example unless dealing exclusively with syntax, we tend to consider $\lambda x\,.\,x$ and $\lambda y\,.\,y$ as the same functions, namely *identity function*. So, roughly speaking we do not care about the name of the bound variables. On the other hand, once we define the notion of substitution, there are cases in which we do want to rename bound variables, in order to avoid trouble. Let us first define *substitution*:

Definition 2.1.5 (substitution). *$M[x/N]$ is the result of substituting the term N for free occurrences of x in M, and is defined inductively as follows:*

$$
\begin{array}{lcl}
x[x/N] & \equiv & N \\
y[x/N] & \equiv & y, \quad \textit{if } x \not\equiv y \\
(\lambda y\,.\,M)[x/N] & \equiv & \lambda y\,.\,(M[x/N]), \quad \textit{if } x \not\equiv y \\
(M_1\,.\,M_2)[x/N] & \equiv & (M_1[x/N])\,.\,(M_2[x/N])
\end{array}
$$

where $\equiv$ denotes syntactic *equivalence.*

Now, in order to declare that terms like $\lambda x\,.\,x$ and $\lambda y\,.\,y$ are to be treated the same, we define a notion of *congruence* called *α-congruence*:

Definition 2.1.6 (α-congruence [Barendregt, 1984]).

1. *A change of bound variables in M is the replacement of a part $\lambda x\,.\,N$ of M by $\lambda y\,.\,(N[x/y])$ where y does not occur (at all) in M.*
2. *M is α-congruent with P, notation $M \equiv_\alpha P$, if P results from M by a series of changes of bound variables.*

For example:

$$\lambda x\,.\,x \equiv_\alpha \lambda y\,.\,y$$
$$\lambda xy\,.\,x \equiv_\alpha \lambda zy\,.\,z$$

At the heart of computation in λ-calculus lies the β-reduction:

Definition 2.1.7 (β-reduction). *β-reduction — denoted by $\to_\beta^\star$ — is the reflexive and transitive closure of the binary relation $\to_\beta$ on Λ defined as in Table 2.1 on the following page.*

Table 2.1 One step β-reduction

$$\frac{}{(\lambda x\,.\,M)N \rightarrow_\beta M[x/N]}$$

$$\frac{M \rightarrow_\beta N}{PM \rightarrow_\beta PN} \qquad \frac{M \rightarrow_\beta N}{MP \rightarrow_\beta NP}$$

$$\frac{M \rightarrow_\beta N}{\lambda x\,.\,M \rightarrow_\beta \lambda x\,.\,N}$$

Definition 2.1.8 (β-equivalence). *λ-terms M and N are said to be β-*equivalent*, written $M \equiv_\beta N$ iff there exist λ-terms $P_0, \ldots, P_n$ ($n \geq 0$) such that*

1. $P_0 = M$
2. $P_n = N$
3. $\forall i \leq n-1 : (P_i \rightarrow_\beta^\star P_{i+1}) \vee (P_{i+1} \rightarrow_\beta^\star P_i) \vee (P_i \equiv_\alpha P_{i+1})$

As mentioned earlier on the page before sometimes we need to rename bound variables in order to avoid trouble. If we adopt the above definitions without any constraint, this trouble shows up fairly easily.

This example is again taken from [Barendregt, 1984]. Consider the λ-term $\mathrm{K} \equiv \lambda xy\,.\,x$. Intuitively, K resembles a function that takes two arguments and returns the first. In other words, for any λ-term M we have :

$$\begin{array}{rcl} \mathrm{K}\,.\,M & = & (\lambda xy\,.\,x)\,.\,M \\ \text{(one step } \beta\text{-reduction)} & \rightarrow_\beta & (\lambda y\,.\,x)\,[x/M] \\ \text{(substitution)} & = & \lambda y\,.\,(x\,[x/M]) \\ \text{(substitution)} & = & \lambda y\,.\,M \end{array}$$

So, $\mathrm{K}\,.\,M$ is the constant function that always returns M, i. e. $\lambda y\,.\,M$. But following the syntax, we must have $\mathrm{K}y = \lambda y\,.\,y$ which is the identity function, not the constant y!

Generally, such problems arise when we try to substitute a term N for x in M, where some free variable of N gets bound. Thus, informally what we do before any substitution is to rename the bound variables so that no new-coming free variable (like y above) gets bound. Including this convention in our previous definitions, we get the satisfactory results and the definitions become complete. This much suffices for our purposes, nonetheless the work of [Pitts, 2001] gives a clearer insight into the subject of bindings.

2.2 Domains for Computation

All programming languages we discuss throughout this thesis are extensions of simply typed λ-calculus by a number of custom designed constants. We make extensive use of mathematical models to interpret the terms of each language. This for us facilitates reasoning about properties of programs that might otherwise — i. e. through purely syntactic arguments — prove to be extremely difficult.

The *fix-point constants* are ubiquitous constants in all the languages to come, with which the programmer is able to define non-trivial terms. *Complete partial orders* (see definition 2.2.7 on the following page) provide the most suitable framework for interpreting the behaviour of fix-point operators. In fact, this — together with a number of other reasons[1] — gave rise to *Domain Theory* in the first place. Today, domains are so commonly used in modelling programming languages that almost any text dealing with *denotational semantics* of programming languages includes a discussion of domains as part of its background material. [Abramsky and Jung, 1994] and [Amadio and Curien, 1998] give more exclusive treatments of domain theory as an independent mathematical subject. Here we present a brief account of the subject according to our own needs.

Definition 2.2.1 (partial order). *Let D be a set and $\sqsubseteq \subseteq D \times D$ a binary relation over D. $(D, \sqsubseteq)$ is called a* partial order *if it satisfies the following:*

- *reflexivity* $\forall x \in D\colon\ x \sqsubseteq x$
- *anti-symmetry* $\forall x, y \in D\colon\ (\ (x \sqsubseteq y) \wedge (y \sqsubseteq x)\ ) \ \Rightarrow\ x = y$
- *transitivity* $\forall x, y, z \in D\colon\ x \sqsubseteq y \sqsubseteq z \Rightarrow x \sqsubseteq z$

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

Notation 2.2.2. *Throughout this thesis, $a \sqsubset b$ means a is* strictly *less than b, i.e:*

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

[1] see [Abramsky and Jung, 1994, part 1.1].

Definition 2.2.3 (bounded (consistent) subsets). *Let $(D, \sqsubseteq)$ be a partial order. Then $B \subseteq D$ is* bounded *(or* consistent*) if it has an upper bound, i. e.*

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

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

Definition 2.2.4 (bounded complete). *A partial order $(D, \sqsubseteq)$ is said to be* bounded complete *if any of its bounded subsets has a* least *upper bound in D.*

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

Definition 2.2.5 (directed subset). *For a partial order $(D, \sqsubseteq)$, a subset $X \subseteq D$ is said to be* directed *if it is non-empty and every pair of its elements have an upper bound in X itself, i.e:*

- $X \neq \emptyset$
- $\forall x, y \in X\, \exists z \in X \colon\, x \sqsubseteq z \;\wedge\; y \sqsubseteq z$

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

A directed set can be viewed as modelling a collection of computational entities flowing towards a common point — supremum — the existence of which is crucial mostly for theoretical purposes, as in practice it proves to be an infinite object except in trivial cases. Hence we restrict the universe of partial orders to:

Definition 2.2.6 (directed-complete partial order (dcpo)). *A poset D is called a* directed-complete partial order*, or **dcpo** for short, if each of its directed subsets $A \subseteq_{dir} D$ has a supremum $\sqcup A$ in D.*

We need to be able to denote the inevitable *non-termination* phenomenon which intuitively falls short of any information. Hence, a *bottom element* $\perp$ added below all other elements of a dcpo results in:

Definition 2.2.7 (complete partial order (cpo)). *A dcpo $(D, \sqsubseteq)$ is called a cpo if it has a least element $\perp$ under $\sqsubseteq$.*

A typical computation process most likely involves (gradual) *approximation* of (ideal) objects via their (*finite*) approximations. Once again, domains turn up as natural choices for interpreting these concepts:

Definition 2.2.8 (finite element). *Let* $(D, \sqsubseteq)$ *be a cpo and* $a, b \in D$*. Then* a *is said to be* way-below b *(or* a approximates b*) — written as* $a \ll b$ *— if the following is true:*

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

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

Definition 2.2.9 (basis, continuous cpo, domain). *Let* $(D, \sqsubseteq)$ *be a cpo.* $B \subseteq D$ *is a* basis *for* D *if:*

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

A cpo $(D, \sqsubseteq)$ *is called* continuous *if* D *is a basis for itself. Throughout this thesis, by* domain *we mean continuous cpo.*[2]

Definition 2.2.10 (algebraic cpo). *A cpo* $(D, \sqsubseteq)$ *is called* algebraic *if the collection of its finite elements forms a basis. In that case, we denote the basis by* $K(D)$*.*

Definition 2.2.11 (ω-continuous, ω-algebraic cpo). *A cpo is called* ω-continuous *if it has a countable basis. If* $(D, \sqsubseteq)$ *is algebraic and* $K(D)$ *is countable, then* $(D, \sqsubseteq)$ *is said to be* ω-algebraic.

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

Example 2.2.12.

1. Let $\mathbb{N}_\perp = \{\perp\} \cup \{0, 1, 2, ...\}$*, partially ordered as follows:*

$$x \sqsubseteq y \iff (x = y \text{ or } x = \perp)$$

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

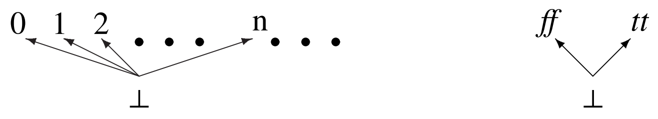
2. Let $\mathbb{B}_\perp = \{\perp, tt, ff\}$ *partially ordered as follows:*

$$x \sqsubseteq y \iff (x = y \text{ or } x = \perp)$$

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

See Figure 2.1 on the next page.

Figure 2.1 $\mathbb{B}_\perp$ and $\mathbb{N}_\perp$



$\mathbb{N}_\perp$ $\mathbb{B}_\perp$

As it is the case with all algebraic theories, we want to have a good definition of *morphism* — i. e. *"structure preserving function"*. We want to use (d)cpo's as models of our data types. Let us think in terms of *information*, in which case, between two elements $a \sqsubseteq b$, b is supposed to contain more information than a. At least two properties can be expected for a morphism between (d)cpo's to satisfy:

1. the more the information in the input of a morphism is given, the more information can be retrieved from the output.

2. if the elements of a (directed) set approximate an (ideal) element in the input, the corresponding output elements also approximate and converge to the image of the ideal element.

Putting these into a formal definition:

Definition 2.2.13 (morphism (continuous functions) between dcpo's).
Let $(D, \sqsubseteq_D)$ and $(E, \sqsubseteq_E)$ be two dcpo's. Then $f\colon (D, \sqsubseteq_D) \to (E, \sqsubseteq_E)$ is said to be a morphism *(or* continuous*) if:*

1. $\forall x, y \in D\colon x \sqsubseteq_D y \Rightarrow f(x) \sqsubseteq_E f(y)$, i. e. f is monotone.

2. $\forall \Delta \subseteq_{dir} D\colon f(\sqcup\Delta) = \sqcup f(\Delta)$, i. e. f preserves the suprema of directed sets.

Note 2.2.14. *As f is supposed to be monotone, it is easy to check that the image of the directed set Δ, namely $f(\Delta)$, is directed.*

If we are using a cpo to model our data type, at times it is natural to expect the image of non-termination under a morphism to be non-termination. This we can formalize as:

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

Definition 2.2.15 (strict morphism). *A morphism* $f\colon (D, \sqsubseteq_D) \to (E, \sqsubseteq_E)$ *between cpo's* $(D, \sqsubseteq_D)$ *and* $(E, \sqsubseteq_E)$ *is* strict *if:*

$$f(\bot_D) = \bot_E$$

Notation 2.2.16. *For cpo's D and E, the collection of all the continuous functions from D to E under the induced pointwise ordering forms a cpo which is written as* $[D \to E]$.

Using the definition of morphism on the facing page it is straightforward to see that the class of dcpo's together with morphisms between them form a *Cartesian closed category*[3]. This is true for cpo's as well. The binary product is constructed pointwise. The order on the objects of any function space is also defined pointwise. The corresponding categories are denoted by **DCPO** and **CPO**, respectively.

2.3 PCF

In one of his seminal works, Gordon Plotkin introduced **PCF** — **P**rogramming language for **C**omputable **F**unctions — which throughout time has established itself as the default basis for designing many more languages. In particular, the ones we will study later on in this thesis are all extensions of PCF. Here we give a description of the language, taken almost entirely from Plotkin's original paper [Plotkin, 1977], slightly modified to match our framework.

PCF is itself an extension of λ-calculus (section 2.1) with suitable ground types and constants to perform arithmetic computations. The set $\mathbb{T}_{\mathrm{PCF}}$ of *types* of PCF is generated by the following grammar:

$$\sigma ::= bool \mid nat \mid \sigma \to \sigma$$

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

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

$$\begin{array}{lcl} true & : & bool \\ false & : & bool \\ if_{bool} & : & bool \to bool \to bool \to bool \\ if_{nat} & : & bool \to nat \to nat \to nat \\ Y_\sigma & : & ((\sigma \to \sigma) \to \sigma) \text{ (one for each } \sigma \in \mathbb{T}_{\mathrm{PCF}}) \end{array}$$

[3]The reader is referred to [Mac Lane, 1971] or [McLarty, 1992] regarding the topic of categories.

To perform arithmetic computations, we also add the following constants to C_0, and call the new set of constants C_{A}:

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

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

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

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

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

to be the set of PCF variables, and:

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

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

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

where $\setminus$ is the relative complement symbol, (see Notation 1.2.4 on page 16).

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

Terms of the form (MN) are called *applications*[4] and sometimes the brackets are dropped, when they are understood as associating to the left. Terms of the form $(\lambda x_i^\sigma\,.\,M)$ are called *abstractions*.

[4]Plotkin called them *combinations*.

Table 2.2 The immediate reduction relation →

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

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

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

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

$$(v) \quad Y_\sigma M \ \rightarrow\ M(Y_\sigma M)$$

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

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

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

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

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

2.3.1 Operational Semantics of PCF

We first define an *immediate reduction relation* → between terms:

Definition 2.3.1 (Immediate Reduction Relation →). *See Table 2.2.*

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

Definition 2.3.2 (operational semantics for PCF: Eval).

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

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

2.3.2 Denotational Semantics of PCF

We will use *cpo's* [5] in our treatment of the denotational semantics. By a *standard collection of domains for PCF* we mean a family

$$\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\mathrm{PCF}}\}$$

of cpo's, i. e. one for each PCF-type $\sigma \in \mathbb{T}_{\mathrm{PCF}}$, such that:

- $\mathbb{D}_{bool} = \mathbb{B}_\perp$
- $\mathbb{D}_{nat} = \mathbb{N}_\perp$
- $\mathbb{D}_{\sigma\rightarrow\tau} = [\mathbb{D}_\sigma \rightarrow \mathbb{D}_\tau]$, i. e. the domain-theoretic function space

The aim is to take $\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\mathrm{PCF}}\}$ as the model and to interpret the terms of PCF inside it. Let us proceed step by step in order to make things clear. First we show how the constants are going to be interpreted via a function which is type-respecting, i.e:

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

Definition 2.3.3. *$\mathcal{A}\colon C_{\mathrm{A}} \rightarrow \cup\{\mathbb{D}_\sigma\}$ is defined as in Table 2.3 on the next page.*

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

$$\begin{aligned} &\rho\colon \mathrm{Var} \rightarrow \cup\{\mathbb{D}_\sigma\} \\ &\rho(x_i^\sigma) \in \mathbb{D}_\sigma \end{aligned}$$

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

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

Now we have all the necessary material to define the *denotational semantics* $[\![\,.\,]\!]\colon \mathcal{T} \rightarrow (\mathrm{Env} \rightarrow \cup\{\mathbb{D}_\sigma\})$ by:

[5] Definition 2.2.7 on page 34

Table 2.3 The function $\mathcal{A}$

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

Definition 2.3.4 (denotational semantics of PCF).

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

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

Having both operational and denotational semantics side-by-side gives us a handy tool for studying properties of objects in one area by analysing the corresponding objects in the other. One such case — indeed an important one — is proving properties of programs via their corresponding object in the model. For instance, if M and N are programs written in PCF, one can prove their "equivalence" via their interpretation in the model. This in turn necessitates the two semantics to match up to a certain degree. It is an important subject with a rich literature behind it. Here we discuss this issue as far as needed for our own purposes, in the context of PCF. For a thorough treatment together with the proofs and details, again see [Plotkin, 1977] from which we will take much of our material, unless stated otherwise.

Perhaps the following theorem is a good place to start from. Here, $\hat{\bot}$ denotes the environment which maps every variable to the bottom element of the corresponding cpo, i. e.

$$\forall x_i^\sigma \in \mathrm{Var}\colon\ \hat{\bot}(x_i^\sigma) = \bot_{\mathbb{D}_\sigma}$$

Theorem 2.3.5. *For any PCF-program $M : \sigma$ and constant $c : \sigma$:*

$$\mathrm{Eval}(M) = c \iff [\![M]\!](\hat{\bot}) = \mathcal{A}(c)$$

This theorem demonstrates how the *behaviour* of a program regarding its *termination* and the constant it evaluates to is reflected in the denotational semantics. Now let us investigate the *equivalence* of programs. For that matter we need the following definition [6]:

Definition 2.3.6 (context with numbered holes). *The set $\mathfrak{C}$ of contexts of PCF with numbered holes — ranged over by C — is generated by the grammar:*

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

where $j \in \mathbb{N}$, $x_i^\sigma \in \mathrm{Var}$ and $c \in C_{\mathrm{A}}$. If all occurrences of the holes bear the same subscript in a term, we denote them by [ ] *for short.*

In other words, contexts are just terms with holes in them. We usually write a context C as $C[.,\dots,.]$ in order not to get confused with ordinary terms. These holes can be filled with terms of appropriate type to turn a context into a term. We denote a context $C[.,\dots,.]$ filled with the terms $M_1,\dots,M_k$ by $C[M_1,\dots,M_k]$. The operation of *filling the holes of a context* is defined as in Table 2.4 on the facing page. Notice that we abbreviate a vector of the form $[N_1,\dots,N_n]$ simply as $[\overrightarrow{N}]$.

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

[6] See [Amadio and Curien, 1998, page 24, Definition 2.1.6]

Table 2.4 Filling the holes of a context

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

Definition 2.3.7 (operational equivalence). *$M \cong N$ if and only if for any context $C[.,\ldots,.]$ such that $C[M,\ldots,M]$ and $C[N,\ldots,N]$ are programs either both of* Eval($C[M,\ldots,M]$) *and* Eval($C[N,\ldots,N]$) *are undefined or else both are defined and equal.*

Note 2.3.8. *It is easy to check that $\cong$ is an equivalence relation.*

One of the reasons we define a denotational semantics for a language is to be able to resort to it as an easier alternative to the (usually) tedious operational semantics when it comes to proving the equivalence of programs, provided the equivalence is reflected properly in the model. Unfortunately this is not the case with PCF and its model as we have defined it. For instance, take the two terms M_0 and M_1 defined as:

$$
\begin{array}{l}
M_i = \lambda x\,.\,\mathit{if}_{nat}(x\ \mathit{true}\ \Omega_{bool}) \\
\qquad\qquad \{\mathit{if}_{nat}(x\ \Omega_{bool}\ \mathit{true}) \\
\qquad\qquad\quad [\mathit{if}_{nat}(x\ \mathit{false}\ \mathit{false}) \\
\qquad\qquad\qquad \Omega_{nat} \\
\qquad\qquad\qquad \bar{i}\ ] \\
\qquad\qquad\quad \Omega_{nat}\} \\
\qquad\qquad \Omega_{nat}
\end{array}
$$

where $i \in \{0,1\}$. Here x is of type $bool \to (bool \to bool)$, Ω_{bool} and Ω_{nat} are non-terminating terms of types *bool* and *nat*, respectively. They could be:

$$
\begin{aligned}
\Omega_{bool} &= Y_{bool}(\lambda x^{bool}\,.\,x) \\
\Omega_{nat} &= Y_{nat}(\lambda x^{nat}\,.\,x)
\end{aligned}
$$

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

$$M_0 \cong M_1$$

On the other hand:

$$[\![M_0]\!](\hat{\bot}) \neq [\![M_1]\!](\hat{\bot}) \tag{2.1}$$

The best way to verify the inequality (2.1) above is to find an argument in the model over which $[\![M_i]\!](\hat{\bot})$'s disagree.

Definition 2.3.9. *The function* $\widehat{\mathsf{por}}\colon \mathbb{B}_\bot \to (\mathbb{B}_\bot \to \mathbb{B}_\bot)$ *is defined by:*

$$\widehat{\mathsf{por}}\ x\ y := \begin{cases} tt & \textit{if} \qquad (x = tt \ \lor \ y = tt) \\ f\!f & \textit{if} \qquad (x = y = f\!f) \\ \bot & \textit{otherwise} \end{cases}$$

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

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

hence the inequality (2.1).

Observing this mismatch, we say that the model is not *fully abstract* for the language. Full abstraction is an important criterion regarding the relation between a language and its model, and in the case of PCF in order to achieve full abstraction there could be two alternative ways ahead: either to purge the model from troublesome objects like $\widehat{\mathsf{por}}$ or otherwise enrich the language. Both possibilities have been pursued but here we follow Plotkin's direction of enriching the language [Plotkin, 1977] in order to get to the parallelism issues.

Let us first add constants:

$$pif_o\colon bool \to o \to o \to o \qquad (o \in \{bool, nat\})$$

to the set C_{A} to get the extended set of constants C^+_{A}, and call the extended language based on that PCF$^+$. Then in order to get an operational semantics for PCF$^+$

Table 2.5 The reduction rules for *pif*

$$
(ix)\quad \begin{cases} pif_o \quad P \quad M \ M \rightarrow_+ M \\ pif_o \ true \ M \ N \rightarrow_+ M \\ pif_o \ false \ M \ N \rightarrow_+ N \end{cases}
$$

$$
(x)\quad \begin{cases} \dfrac{P \rightarrow_+ P'}{pif_o \ P \rightarrow_+ pif_o \ P'} \\ \dfrac{M \rightarrow_+ M'}{pif_o \ P \ M \rightarrow_+ pif_o \ P \ M'} \\ \dfrac{N \rightarrow_+ N'}{pif_o \ P \ M \ N \rightarrow_+ pif_o \ P \ M \ N'} \end{cases}
$$

we extend the relation $\rightarrow$ of Definition 2.3.1 on page 39 by the rules of Table 2.5 for pif_o, ($o \in \{bool, nat\}$), and denote the new immediate reduction relation by $\rightarrow_+$.

Let us mention some facts about the constants pif_o, ($o \in \{bool, nat\}$). As you might have guessed, the prefix p stands for *parallel*, hence we read *pif* as "parallel-if". A closer look at the rules (ix) and (x) of Table 2.5 above reveals the parallelism as pif_o looks at its three arguments at the same time. Rule (x) consists of three rows, with the first one being the only one having a counterpart for *if*, see rule ($viii$), Table 2.2 on page 39.

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

$$
\begin{aligned}
\forall M_1, M_2, M_3 \colon (M_1 \overset{\star}{\rightarrow}_+ M_2) \wedge (M_1 \overset{\star}{\rightarrow}_+ M_3) \\
\Rightarrow \exists M, M' \colon (M \cong_\alpha M') \wedge \left\{ (M_2 \overset{\star}{\rightarrow}_+ M) \wedge (M_3 \overset{\star}{\rightarrow}_+ M') \right\}
\end{aligned}
$$

where $\cong_\alpha$ is the α-congruence between terms, i. e. equivalence up to renaming of bound variables (Definition 2.1.6 on page 31), thus Eval$^+$ is well-defined.

We extend $\mathcal{A}$ of Definition 2.3.3 on page 40 to $\mathcal{A}^+ \colon C_{\mathrm{A}}^+ \rightarrow \cup\{\mathbb{D}_\sigma\}$ as in Table 2.6 on the following page, based on which the definition of the denotational semantics $[\![\,.\,]\!]_+$ for PCF$^+$ should be straightforward.

Table 2.6 The function $\mathcal{A}^+$

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

$$\mathcal{A}^+(pif_o)(p)(x)(y) = \begin{cases} x & (p = tt) \\ y & (p = ff) \\ x & (p = \perp \text{ and } x = y) \\ \perp_o & (p = \perp \text{ and } x \neq y) \end{cases} \qquad (p \in \mathbb{B}_\perp \text{ and } x, y \in \mathbb{D}_o)$$

Remark 2.3.10. *Here we tried to follow Plotkin's original definitions and therefore added both pif_{bool} and pif_{nat} where any of the two would suffice. In fact we could have just as well added a constant $por\colon bool \to (bool \to bool)$ with the following immediate reduction rules:*

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

$$\frac{M \to M'}{por\ M \ \to\ por\ M'}$$

$$\frac{N \to N'}{por\ M\ N \ \to\ por\ M\ N'}$$

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

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

where $\widehat{\mathsf{por}}$ is the function defined in Definition 2.3.9 on page 44. For a proof of the interdefinability of pif_{bool}, pif_{nat} and por in PCF, see [Stoughton, 1991].

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

Theorem 2.3.11. *For any* PCF^+*-terms M_σ and N_σ:*

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

In fact we have more:

Theorem 2.3.12. *Every finite element*[7] *of any $\mathbb{D}_\sigma$ is definable by a* PCF$^+$*-term.*

By Theorem 2.3.11 on the preceding page we have got full abstraction. To see how far we have come, perhaps it is better to go back to the terms M_0 and M_1 on page 43. As PCF$^+$ terms, we have $M_0 \ncong M_1$, as we only need to apply

$$\lambda x \, . \, x \; por$$

to both terms. On the other hand, there exists no such term in PCF. In fact, a closer look reveals that there is no way of distinguishing these two terms unless some parallel mechanism already exists in the language. That is why in PCF we get $M_0 \cong M_1$. The precise proof of this fact though, is a bit involved (see [Plotkin, 1977]).

It seems intuitively reasonable to take the l.u.b.'s of recursively enumerable sets of finite elements of the model as the collection of *computable* elements. Any element defined by a PCF$^+$-term is computable[8] and as the above theorem says, any finite element of the model is captured by the language PCF$^+$. But it turns out that there are computable objects of the model not accounted for in the language. One such object is $\widehat{\exists} \in \mathbb{D}_{(nat \to bool) \to bool}$ defined by:

Definition 2.3.13 (continuous existential quantifier : $\widehat{\exists}$).
The function $\widehat{\exists} : (\mathbb{N}_\perp \to \mathbb{B}_\perp) \to \mathbb{B}_\perp$ is defined as:

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

We then proceed by adding a constant $\exists$: $(nat \to bool) \to bool$ to C_{A}^{+} and denote the new set of constants as C_{A}^{++}. We call the language based on this set PCF^{++}. The reduction rule of Table 2.7 on the following page is added to $\to_+$ to obtain $\overset{\star}{\to}_{++}$. Note that $\overset{\star}{\to}_{++}$ is the transitive-reflexive closure of $\to_{++}$.

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

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

[7] Definition 2.2.8 on page 35
[8] See [Plotkin, 1977].

Table 2.7 The reduction rule for $\exists$

$$(xi)\begin{cases} \dfrac{F\Omega_{nat} \overset{\star}{\rightarrow}_{++} \mathit{false}}{\exists F \rightarrow_{++} \mathit{false}} \\[2ex] \dfrac{F\overline{m} \overset{\star}{\rightarrow}_{++} \mathit{true}}{\exists F \rightarrow_{++} \mathit{true}} & (m \geq 0) \end{cases}$$

is well-defined. Finally we define $\mathcal{A}^{++} \colon C_{\mathrm{A}}^{++} \rightarrow \cup\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\mathrm{PCF}}\}$ by:

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

Again extending $[\![\,.\,]\!]_+$ to a denotational semantics $[\![\,.\,]\!]_{++}$ for PCF^{++} using $\mathcal{A}^{++}$ is straightforward. The language is now rich enough to define all computable elements of the model:

Theorem 2.3.14. *An element of $\mathbb{D}_\sigma$ is computable if and only if it is definable by a* PCF^{++}*-term.*

2.4 Sequentiality

2.4.1 Theory versus Practice

Designing a language and its operational semantics, presenting a model, interpreting the language inside it and studying the relation between the operational and denotational semantics — one might like to categorize them as *theoretical issues* — are just parts of a bigger challenge, though quite important ones. In practice other issues arise such as efficiency regarding time and/or space. In Section 2.3.3 we tried to summarize the theoretical issues and demonstrate the success in that regard, yet we never cared about the efficiency of computations. Whenever we realized something missing, we did not hesitate to remedy by any means.

First consider PCF and the immediate reduction rules over its terms (Table 2.2 on page 39). For any term, there is *at most* one rule that applies. Hence the operational semantics is *deterministic*. Imagine a machine implementing PCF, in the middle of a computation, reducing a term $C[M_1, \ldots, M_i, \ldots, M_k]$. If M_i is the subterm being worked on at the moment, there is no way the process will jump to another subterm M_j ($j \neq i$) unless the computation on M_i is finished off and evaluated to a constant. We try to formulate a property that captures this intuition and call the languages satisfying this property *sequential*.

Sequentiality can be studied both syntactically and semantically. Plotkin's *activity lemma* [Plotkin, 1977] is an example of a *syntactic* formulation, Berry's *syntactic sequentiality theorem* [Berry, 1979][9] is another which for the reader can serve as a good motivation for a *semantic* investigation of sequentiality first introduced by Vuillemin [1974], though in reality Vuillemin's work preceded that of Berry's!

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

2.4.2 Vuillemin-Sequentiality

Definition 2.4.1 (flat cpo). *Given any nonempty set X,* $(X_\perp, \sqsubseteq)$ *defined by:*

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

and

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

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

Examples of flat cpo's we use are $\mathbb{B}_\perp$ and $\mathbb{N}_\perp$ (Example 2.2.12 on page 35) and $\mathbb{2}$ (Definition 3.3.2 on page 74).

Definition 2.4.2 (sequential function (Vuillemin)).

- *Let* $D, D_1, \ldots, D_n$ *be flat cpo's, and* $f : D_1 \times \cdots \times D_n \to D$ *continuous. Now consider* $\boldsymbol{x} = (x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n$*, and suppose that* $f(\boldsymbol{x}) = \perp$*. We say that* f *is sequential at* $\boldsymbol{x}$ *if either* $f(\boldsymbol{z}) = \perp$ *for all* $\boldsymbol{z} \sqsupseteq \boldsymbol{x}$*, or there exists* i *such that* $x_i = \perp$ *and:*

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

  Such an i *is called a* sequentiality index for f at $\boldsymbol{x}$.

- f *is* sequential *if it is sequential at all* $\boldsymbol{x}$ *in its domain.*

[9] See [Amadio and Curien, 1998, section 2.4, page 41] for an English version.

Note that Vuillemin-sequentiality is defined *only for first order functions over flat cpo's*. In fact the definition as it is cannot be generalized to higher-order types because a function like $\widehat{\exists}$ (Definition 2.3.13 on page 47) which is intuitively of an infinite parallel nature would become Vuillemin-sequential, while

$$\mathsf{ev}\colon (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$$

would not satisfy the sequentiality criteria. But at first-order the definition works well as it can be shown that for *compact* first order functions f in $\cup\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\mathrm{PCF}}\}$, Vuillemin-sequentiality coincides with PCF-definability (Theorem 2.4.8 on page 52).

Also of interest is the Vuillemin-sequentiality of *all* unary first order functions over flat cpo's. This can be used to show that although all first order PCF-definable functions in $\cup\{\mathbb{D}_\sigma\}$ are Vuillemin-sequential [Amadio and Curien, 1998, Exercise 6.5.5, page 137], the converse is not true as there are uncountably many elements in $\mathbb{D}_{nat\to nat}$, all of which are Vuillemin-sequential, while there are only countably many PCF-definable elements in there.

Equally interesting, at least for our purposes, is Sieber's approach [Sieber, 1992] which proves to have a tight relation with Vuillemin's definition, as we shall see in Theorem 2.4.8 on page 52. But before that, we need to take a look at an important tool called *logical relations*.

2.4.3 Logical Relations

Notation 2.4.3 ($\Lambda(C)$). *Throughout this thesis, by $\Lambda(C)$ we mean the extension of the simply-typed λ-calculus with a set of constants C.*

Definition 2.4.4 (Logical Relations).

1. Let

1. $\mathcal{M}_i$, $(1 \le i \le k)$ be k models of $\Lambda(C)$.

2. $D_i^\sigma = [\![\sigma]\!]_{\mathcal{M}_i}$ (where σ ranges over types).

3. for any ground type o, R_k^o be some arbitrary k-ary relation over $D_1^o \times \cdots \times D_k^o$, i. e.

$$R_k^o \subseteq D_1^o \times \cdots \times D_k^o$$

Then the k-ary logical relation R_k between $\mathcal{M}_1, \ldots, \mathcal{M}_k$ is built up from R_k^o's by the following rule for function types:

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

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

2. *$f \in D^\sigma$ is said to* preserve *— or to be* invariant *under — the logical relation R_k if and only if:*

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

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

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

Logical relations are useful especially when it comes to establishing links between syntax and semantics as in Jung-Tiuryn's theorem on lambda-definability [Jung and Tiuryn, 1993]. Of course, Jung and Tiuryn used a more powerful class of logical relations called *Kripke logical relations* which, unlike our definition, have varying arities.

Anyway, Definition 2.4.4 on the facing page is enough for our purposes. The important part of defining a logical relation is over the ground types, as that is the part over which we have control. Then having defined a suitable logical relation, we make extensive use of the following important lemma:

Lemma 2.4.5 (Fundamental lemma of logical relations). *Let R be a k-ary C-logical relation — C a set of constants — between k models of $\Lambda(C)$, namely $\mathcal{M}_1, \ldots, \mathcal{M}_k$. Then for any closed $\Lambda(C)$-term M of type σ we have:*

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

Note 2.4.6. *To find out more about logical relations as presented here, including a proof of the above lemma, see [Amadio and Curien, 1998, Chapter 4, Section 5].*

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

Definition 2.4.7 (Sieber-sequential relations). *Let $A \subseteq B \subseteq \{1, \ldots, n\}$ and consider the following n-ary relations $\mathcal{S}^n_{A,B}$ over ground types $o \in \{nat, bool\}$:*

$$(d_1, \ldots, d_n) \in \mathcal{S}^n_{A,B} \iff (\exists\, i \in A : d_i = \bot) \vee (\forall i, j \in B : d_i = d_j)$$

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

It can be shown that C-logical relations, where C is the set of all PCF constants C_{A} on page 38 are exactly the Sieber-sequential relations of Definition 2.4.7 on the page before (see [Amadio and Curien, 1998, Exercise 6.5.3, page 136]).

Now let us take some special cases of Sieber-sequential relations, namely:

$$S_{k+1} \coloneqq \mathcal{S}^{k+1}_{\{1,\ldots,k\},\{1,\ldots,k+1\}}$$

where $k \geq 1$ and:

$$(x_1, \ldots, x_{k+1}) \in S_{k+1} \iff (\exists j \leq k \colon x_j = \bot) \vee (x_1 = \cdots = x_{k+1} \neq \bot) \tag{2.2}$$

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

Theorem 2.4.8. *For a compact first-order function f in $\cup\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{PCF}\}$, the following are equivalent:*

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

For a proof of this theorem see [Amadio and Curien, 1998, Theorem 6.5.4, page 137].

2.5 Real-PCF

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

Traditionally we are used to computation on real numbers via *floating point approximations* which is satisfactory for everyday business but can prove to be extremely unreliable in special circumstances. Floating point computation carries the problem of *round-off errors* with it, which we try to ignore in everyday life applications for a variety of reasons. Ménissier-Morain [1996] explored this subject together with two interesting examples demonstrating the *unreliability of floating point approach*. Accordingly the idea of *exact real number computation* has been put forward which is, as opposed to floating point computation, reliable, i. e. the output produced is guaranteed to be correct. Moreover the results can be computed *effectively* (e.g. as opposed to BSS approach [Blum et al., 1989]) to within any desired degree of accuracy.

Exact real number computation itself can be approached in two ways. At first people focused on *representation* while neglecting the issue of *data types* for real numbers, among which [Böhm et al., 1986; Böhm and Cartwright, 1990] are considered seminal. On the other hand, perhaps [Di Gianantonio, 1993] is among the earliest works where there is a clear distinction between a representation-dependent operational semantics and a representation-independent denotational semantics. Di Gianantonio added to PCF a ground type which is interpreted as a domain of real numbers. This domain turns out to be algebraic[10] and therefore *cannot have the real line as its subspace of maximal elements*. This creates the possibility of defining multi-valued functions over real numbers [Di Gianantonio, 1993, page 62][11].

Escardó [1997] introduced Real-PCF following similar ideas. He added to PCF a ground type for real numbers interpreted as the so-called *unit interval domain*[12] which has the interval $[0, 1]$ as its subspace of maximal elements. Also the problem of multi-valuedness with Di Gianantonio's approach is avoided in Real-PCF. Of course there is much more to both Di Gianantonio's and Escardó's works. Here we present an overview of Real-PCF as it is part of the necessary background to the material in Chapter 3. It also serves the purpose of familiarizing the reader with what a language for computation over real numbers can look like.

Definition 2.5.1 (Real-PCF).

- *The set $\mathbb{T}_{RPCF}$ of Real-PCF types is generated by the grammar:*

$$\sigma ::= bool \mid nat \mid I \mid \sigma \to \sigma$$

- *The set of Real-PCF constants $\mathcal{RC}_{\mathrm{A}}$ is the extension of the set of PCF-constants C_{A} on page 38 with the ones shown in Table 2.8 on the following page*

The aim is to take the ground type I as the type of *real numbers in the unit interval, i. e.* $[0, 1]$ and use the constants introduced in the table for computation

[10] Definition 2.2.10 on page 35.

[11] Of course on the same page, Di Gianantonio himself provides an ad hoc method of avoiding multi-valuedness, nonetheless it does not *prevent* programmer from writing multi-valued functions.

[12] Definition 2.5.3 on the following page.

Table 2.8 The set $\mathcal{RC}_{\mathrm{A}}$ of Real-PCF constants

$$\begin{aligned} &cons_a &&: I \to I \\ &tail_a &&: I \to I \\ &head_r &&: I \to bool \\ &pif_I &&: bool \to I \to I \to I \end{aligned}$$

over them. The parameter a — as a subscript of *cons* — ranges over intervals with rational end-points in $[0, 1]$, i. e.

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

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

$$p < q$$

The subscript r ranges over rational numbers in the interior of the unit interval, so:

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

Notation 2.5.2 (RPCF). *We freely use the abbreviation RPCF for Real-PCF.*

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

1. $\mathcal{T}$: the set of *RPCF-terms*.
2. Var : the set of *RPCF-variables*.
3. FV(M) : the set of the *free variables of the RPCF-term M*.

2.5.1 Denotational Semantics: Interval Domain Model

Let us begin with the ground type I whose denotation $\mathbb{D}_I$ we take to be:

Definition 2.5.3 (unit interval domain). *The cpo $(\mathcal{I}, \sqsubseteq)$ defined by:*

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

- $\forall a, b \in \mathcal{I}: a \sqsubseteq b \iff a \supseteq b$

is called the ***unit interval domain****.*

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

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

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

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

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

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

Moreover the countable set

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

forms a basis[13] for $\mathcal{I}$ and makes it an *ω-continuous cpo*[14].

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

Definition 2.5.4 (collection of domains for RPCF). $\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{RPCF}\}$ *is* the collection of domains for RPCF *defined by induction over the structure of σ as follows:*

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

[13] Definition 2.2.9 on page 35.

[14] Definition 2.2.11 on page 35.

As we did for PCF[15] we first try to interpret the constants via a function

$$\mathcal{RA}\colon \mathcal{R}C_{\mathrm{A}} \to \cup\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\mathrm{RPCF}}\}$$

and then extend it in a natural way to a denotational semantics. For any constant $c \in C_{\mathrm{A}}$, we simply define $\mathcal{RA}(c) \coloneqq \mathcal{A}(c)$, where $\mathcal{A}$ is defined as in Table 2.3 on page 41.

Remark 2.5.5. *With each RPCF-type σ which is not a PCF-type, a new fix-point constant Y_σ is added to the language. Nevertheless, the interpretation is formulated the same as in Table 2.3 on page 41 for PCF-types, i. e.*

$$\mathcal{RA}(Y_\sigma)(f) = \sqcup \{f^n(\perp) \mid n \geq 0\} \qquad (f \in \mathbb{D}_{\sigma\to\sigma})$$

For the proper RPCF-constants, let us present their denotations in a more intuitive fashion. From a geometric point of view, it does not matter if we choose two other numbers $r < s$, rather than 0 and 1, and build our interval domain upon it. Let us suppose $0 \leq r \leq s \leq 1$ and denote the interval $[r, s]$ by a, and the interval domain built upon it by $a\mathcal{I}$, bearing in mind that $a\mathcal{I}$ is the singleton domain in the case $r = s$. If $r < s$ then $\mathcal{RA}(cons_a)$ is defined to be the scaling isomorphism — denoted by Cons_a — from $\mathcal{I}$ to $a\mathcal{I}$, otherwise (i. e. $r = s$) it is simply the unique constant function available. We always consider the codomain to be the whole of $\mathcal{I}$, hence say $\mathcal{RA}(cons_a)\colon \mathcal{I} \to \mathcal{I}$. There is apparently another scaling isomorphism from $a\mathcal{I}$ back to $\mathcal{I}$ (if $r \neq s$) that we call $\mathsf{Tail}_a\colon a\mathcal{I} \to \mathcal{I}$. We can extend the domain of the function from $a\mathcal{I}$ to $\mathcal{I}$ so that it remains a morphism in the category **CPO**. In fact we consider the maximal extension (under the order relation on $[\mathcal{I} \to \mathcal{I}]$) and for simplicity denote it by the same name Tail_a, hence $\mathsf{Tail}_a : \mathcal{I} \to \mathcal{I}$, and we assign $\mathcal{RA}(tail_a) = \mathsf{Tail}_a$. Figure 2.2 on the facing page may give a better intuition.

Now perhaps the following formulae are easier to follow:

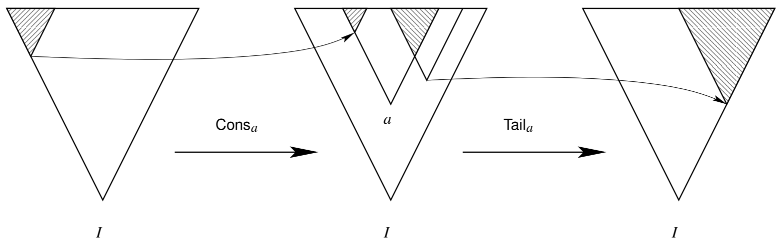
$$\mathcal{RA}(cons_a)(b) = [r + (s - r)x, r + (s - r)y] \tag{2.3}$$
$$\mathcal{RA}(tail_a)(b) = [(x' - r)/(s - r), 1 - (s - y')/(s - r)] \tag{2.4}$$

where:

$$\begin{aligned} a &= [r, s] \\ b &= [x, y] \\ x' &= \min(\max(r, x), s) \\ y' &= \max(\min(s, y), r) \end{aligned}$$

[15] Section 2.3.2.

Figure 2.2 Cons and Tail



The denotation of the constants $head_r$ and pif_I can be easily described explicitly while their corresponding immediate reduction rule (see Definition 2.5.7 on page 59) tells us all about their expected behaviour. Let r be a rational number $0 < r < 1$:

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

$$\forall p \in \mathbb{B}_\perp, \forall x, y \in \mathcal{I}: (\mathcal{RA}(pif_I))\,(p)(x)(y) = \begin{cases} x & \text{if} \quad p = tt \\ y & \text{if} \quad p = ff \\ x \sqcap y & \text{if} \quad p = \perp \end{cases}$$

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

$$[\![\,.\,]\!]: \mathcal{T} \to (\text{Env} \to \cup\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\text{RPCF}}\})$$

for RPCF, following the same style as we did for PCF in Definition 2.3.4 on page 41.

Note 2.5.6. *We have not modified any symbols from PCF to RPCF (except $\mathcal{A}$ to $\mathcal{RA}$) and it will be clear from the context which one of PCF or RPCF we are dealing with, hence there should be no confusion.*

Consider any two intervals $a, b \in \mathcal{I}$ and define:

$$ab = [a_1 + (a_2 - a_1)b_1, a_1 + (a_2 - a_1)b_2] \tag{2.5}$$

where

$$\begin{cases} a = [a_1, a_2] \\ b = [b_1, b_2] \end{cases}$$

It is easy to verify that:

$$\mathcal{RA}(cons_{ab}) = \mathcal{RA}(cons_a) \circ \mathcal{RA}(cons_b)$$

where $\circ$ is just functional composition.

Now consider $a, b \in \mathcal{I}$ as above, this time subject to the conditions:

1. $a_1 \neq a_2$
2. $a \sqsubseteq b$

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

$$c = [(b_1 - a_1)/(a_2 - a_1), (b_2 - a_1)/(a_2 - a_1)] \tag{2.6}$$

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

$$(cons_{a_1},\ cons_{a_1} cons_{a_2}, \ldots, cons_{a_1} \ldots cons_{a_n}, \ldots)$$

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

$$cons_{a_1} cons_{a_2} \ldots cons_{a_n} \ldots$$

2.5.2 Operational Semantics

We extend the immediate reduction relation of PCF[16] to one for RPCF and still denote it by $\rightarrow$. The aim is to reduce any Real-PCF program M of type I to some $cons_a M'$ — where a has rational end-points — and then continue reducing M' to $cons_{a'} M''$, and so on. This way we produce a stream of digits.

Before presenting the definition, take note of the following:

1. There are RPCF-types σ that are not PCF-types. Correspondingly, there are new *fix-point combinators* $Y_\sigma \colon (\sigma \to \sigma) \to \sigma$ that are not PCF-constants. But still the reduction rule is the same as clause (v) of Table 2.2 on page 39, i. e.

$$Y_\sigma M = M(Y_\sigma M)$$

[16]Table 2.2 on page 39.

in particular:

$$\begin{cases} Y_I\ cons_{[0,1/2]} = 0 \\ Y_I\ cons_{[1/2,1]} = 1 \end{cases}$$

2. For intervals $a, b \in \mathcal{I}$ we define:

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

where $a = [a_1, a_2]$ and $b = [b_1, b_2]$.

3. Similarly for a real number r and an interval $a = [a_1, a_2]$ we define:

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

Definition 2.5.7 (immediate reduction relation for RPCF). *The immediate reduction relation $\rightarrow$ is the extension of the corresponding relation from PCF (Table 2.2 on page 39) to Real-PCF by the rules of Table 2.9 on the following page.*

Notation 2.5.8. *We denote the reflexive and transitive closure of $\rightarrow$ by $\overset{\star}{\rightarrow}$.*

Definition 2.5.9 (operational semantics for RPCF). *The map* Eval *of Definition 2.3.2 on page 39 can be extended to a partial map over RPCF-programs (which we still denote by* Eval*) by the following case for programs M of type I:*

$$\mathrm{Eval}(M) \coloneqq \{a \in \mathcal{I} \mid M \overset{\star}{\rightarrow} cons_a M', \textit{ for some } M'\}$$

Note 2.5.10. *For a more precise and comprehensive treatment of Real-PCF, see [Escardó, 1997].*

Table 2.9 The immediate reduction rules of RPCF

1. $cons_a(cons_b M) \rightarrow cons_{ab} M$
2. $tail_a(cons_b M) \rightarrow Y_I\ cons_{[0,1/2]}$ .... (if $b \leq a$)
3. $tail_a(cons_b M) \rightarrow Y_I\ cons_{[1/2,1]}$ .... (if $a \leq b$)
4. $tail_a(cons_b M) \rightarrow cons_{b \backslash a} M$ .... (if $a \sqsubseteq b$ and $a \neq b$)
5. $tail_a(cons_b M) \rightarrow cons_{(a \sqcup b) \backslash a}(tail_{(a \sqcup b) \backslash b} M)$

   (if $a \uparrow b,\ a \not\sqsubseteq b,\ b \not\sqsubseteq a,\ a \nleq b,\ b \nleq a$)
6. $head_r(cons_a M) \rightarrow true$ .... (if $a < r$)
7. $head_r(cons_a M) \rightarrow false$ .... (if $a > r$)
8. $$\begin{cases} pif\ true\ M\ N \rightarrow M \\ pif\ false\ M\ N \rightarrow N \end{cases}$$
9. $pif\ L\ (cons_a M)\ (cons_b N) \rightarrow$

   $cons_{a \sqcap b}(pif\ L\ (cons_{a \backslash (a \sqcap b)} M)\ (cons_{b \backslash (a \sqcap b)} N))$

   (if $a \sqcap b \neq \perp$)
10. $$\frac{N \rightarrow N'}{MN \rightarrow MN'}$$ .... ($M \in \{cons_a,\ tail_a,\ head_r, pif_I\}$)
11. $$\frac{M \rightarrow M'}{pif_I\ L\ M \rightarrow pif_I\ L\ M'} \qquad \frac{N \rightarrow N'}{pif_I\ L\ M\ N \rightarrow pif_I\ L\ M\ N'}$$

Part II

New Material

Chapter 3

Real-PCF without parallel-if

Unlike PCF, RPCF has a constant for parallel computation, i. e. pif_I. By the time Escardó [1997] put forward RPCF, it was already speculated that representation-independent real number computation needs parallel operators (see e. g. [Böhm et al., 1986; Di Gianantonio, 1993]). Therefore pif_I was included in RPCF right from the beginning. Escardó [1997] showed that all computable elements of rank at most 1 in the interval domain model are definable in RPCF. Of course, like PCF, there are higher-order objects such as $\widehat{\exists}$ of Definition 2.3.13 on page 47 not definable in RPCF. But by adding a constant $\exists$ for existential quantification, the language becomes *universal* for the model, i. e. all computable objects of any order in the interval domain model are definable (see [Escardó, 1996]).

Though adding parallel operators to the language solves the definability problems, they come at a heavy cost. Therefore one would rather have a more efficient substitute for pif_I, which prompts us to analyze the language and its model more carefully to have a better view of (potential) choices.

One might think of minimal changes in the setting, keeping the model intact, while getting rid of *all* kinds of parallelism in the *language*. This idea was ruled out by Escardó, Hofmann, and Streicher [1998, 2004] where they proved that already if functions as basic as average $\oplus\colon [0,1] \to [0,1] \to [0,1]$ defined by

$$\forall x, y \in [0,1]\colon\ x \oplus y := \frac{x+y}{2} \tag{3.1}$$

are definable in the language[1], then the existence of some parallel mechanism is inevitable. Of course this result crucially relies on the fact that we presumed

[1] To be precise, one demands the definability of some *continuous extension* of average to the whole interval domain $\mathcal{I}$.

keeping the model intact. For instance, in the interval domain model — the model in which this result was studied — all functions on real numbers are *extensional* over both partial and total real numbers. Thus, in particular, for any RPCF-terms $a, b \colon I$ and $f \colon I \to I$:

$$[\![a]\!] = [\![b]\!] \Rightarrow [\![fa]\!] = [\![fb]\!]$$

even when $[\![a]\!]$ is only a *partial* real number.

In practice, however, extensionality might even be considered as aesthetics. Therefore, in view of the previous argument, one can think of coming up with a new setting in which extensionality is traded off for sequentiality, in addition — of course — to replacing pif_I with a more efficient constant[2]. Here a subtle problem arises:

> In principle, before commencing on any search for more efficient substitutes for pif_I, one needs to be assured of the fragment of RPCF without pif_I being sequential.

In Sections 3.2, 3.3 and 3.4 we investigate this problem from three different angles.

3.1 weak-RPCF

Definition 3.1.1 ($w\mathcal{RC}_{\mathrm{A}}$). *Let $w\mathcal{RC}_{\mathrm{A}}$ denote the set of RPCF-constants with pif_I removed, i. e.*

$$w\mathcal{RC}_{\mathrm{A}} := \mathcal{RC}_{\mathrm{A}} \setminus \{pif_I\}$$

where $\mathcal{RC}_{\mathrm{A}}$ is as in Definition 2.5.1 on page 53.

Definition 3.1.2 (weak-RPCF (wRPCF)). *By* wRPCF *we mean the fragment of RPCF built upon the set of constants $w\mathcal{RC}_{\mathrm{A}}$ of the above definition. In other words,* wRPCF *is RPCF without pif_I.*

Remark 3.1.3. *Note that the set of* wRPCF*-types is the same as that for RPCF.*

[2] Marcial-Romero and Escardó [2004] put forward a framework in which single-valuedness — over *both* partial and total real numbers —is sacrificed in favour of sequentiality. On the other hand, in Chapter 4 we will present a language in which extensionality is relaxed in favour of sequentiality, but *only over partial real numbers*.

3.2 Vuillemin-Sequentiality

To study sequentiality, first we need to fix a criterion and then test wRPCF against it. The one we consider in this section is a generalization of Vuillemin-sequentiality of Definition 2.4.2 on page 49 to functions over the interval domain. This way we will show that the first-order wRPCF-definable functions are sequential.[3]

Definition 3.2.1 (generalized Vuillemin-sequentiality).

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

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

  is said to be ***Vuillemin-sequential*** *(or simply* ***sequential****) at*

  $$\boldsymbol{x} = (x_1, \ldots, x_n) \in D_1 \times \cdots \times D_n$$

  if either of the following holds:

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

  $$(\boldsymbol{z} \sqsupset \boldsymbol{x}) \wedge (f(\boldsymbol{z}) \sqsupset f(\boldsymbol{x})) \Rightarrow z_i \sqsupset x_i$$

  The i *in clause 2 above is called a* ***sequentiality index*** *for* f *at* $\boldsymbol{x}$.

- f *is* ***sequential*** *if it is sequential at all* $\boldsymbol{x}$ *in its domain.*

Remark 3.2.2. *Note that by the definition above, all unary first-order functions of the interval domain model are trivially sequential.*

Although as discussed before[4] Vuillemin-sequentiality was originally introduced for flat cpo's only, our generalization can be made legitimate on the following grounds:

[3] In [Farjudian, 2003a] we considered another criteria, i. e. *conservativity over PCF*. This we will present in Section 3.4.

[4] See page 50.

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

Matching the expectations : As we shall see (from Remark 3.2.4 and Lemma 3.2.7 on page 69) any function with intuitive parallel behaviour is not Vuillemin-sequential. In particular, functions like $\widehat{\mathsf{pif}}_I$ or $\widehat{\mathsf{por}}$ are not Vuillemin-sequential.

We demonstrate that all first-order wRPCF-definable functions are sequential. Our proof is a generalization of that of Theorem 2.4.8 on page 52 as presented in [Amadio and Curien, 1998, Theorem 6.5.4, page 137], which can give a much better idea as to why we define the logical relations S_{k+1} as in the definition below — a direct generalization of the logical relations defined in Equation (2.2) on page 52 to non-flat domains. Also, it is worth mentioning that both proofs are crucially based on the so-called *fundamental lemma of logical relations* (Lemma 2.4.5 on page 51).

Let us assume that for any RPCF-type σ, $\mathbb{D}_\sigma$ is the interpretation of type σ in the interval domain model[5].

Definition 3.2.3 (S_{k+1}). *For any $k \geq 1$, we define the relation S^o_{k+1} of arity $k+1$ over the elements of $\mathbb{D}_o$, where o ranges over the ground types bool, nat or I, by:*

$$(x_1, \dots, x_k, x_{k+1}) \in S^o_{k+1} \iff \exists j \leq k\ \forall i \leq k+1 : x_j \sqsubseteq x_i$$

over which the logical relation S_{k+1} is built up[6].

Remark 3.2.4. *It is worth mentioning right here that the so-called parallel operators do not preserve all S_{k+1}'s. As an example, take $\widehat{\mathsf{por}}$ (Definition 2.3.9 on page 44) and $k = 2$. In Table 3.1 on the facing page the first two left columns are elements of S_3^{bool} whereas the rightmost column, which is the result of applying $\widehat{\mathsf{por}}$ over the elements of the first two, is not in S_3^{bool}.*

[5] See Definition 2.5.4 on page 55.
[6] as in Definition 2.4.4 on page 50.

Table 3.1 $\widehat{\mathsf{por}}$ does not preserve S_3

$\top$	$\bot$	$\xrightarrow{\widehat{\mathsf{por}}}$	$\top$
$\bot$	$\top$	$\xrightarrow{\widehat{\mathsf{por}}}$	$\top$
$\bot$	$\bot$	$\xrightarrow{\widehat{\mathsf{por}}}$	$\bot$

On the other hand, it is pretty easy to show that each S_{k+1} is in fact a $w\mathcal{RC}_{\mathrm{A}}$-logical relation. If c is a unary constant in $w\mathcal{RC}_{\mathrm{A}}$, then the S_{k+1}'s are preserved as a result of $\mathcal{RA}(c)$ being monotone[7]. If $c \in \{if_{bool}, if_{nat}\}$, then it can be verified by case analysis over the boolean argument. The only non-trivial case may be the *fix-point operators* Y_σ:

Remember from Remark 2.5.5 on page 56 that for each wRPCF-type σ there is a wRPCF-constant $Y_\sigma : (\sigma \to \sigma) \to \sigma$, with the denotation $[\![Y_\sigma]\!] \in \mathbb{D}_{(\sigma\to\sigma)\to\sigma}$ defined by:

$$\forall f \in \mathbb{D}_{\sigma\to\sigma} : [\![Y_\sigma]\!](f) = \sqcup\{f^{(n)}(\bot_{\mathbb{D}_\sigma}) \mid n \geq 0\} \tag{3.2}$$

So, to prove that Y_σ preserves all S_{k+1}'s, we first show that the set of S_{k+1} invariant elements of any $\mathbb{D}_\sigma^{(k+1)}$ forms a so-called *inclusive predicate*:

Definition 3.2.5 (Inclusive Predicate). *A predicate $\mathcal{P}$ over the dcpo D is inclusive if it is closed under l.u.b.'s of ascending chains. In particular, in case D happens to be a cpo — following the convention of taking the l.u.b. of the empty chain to be $\bot_D$ — $\mathcal{P} \subseteq D$ is inclusive if and only if the following two conditions hold:*

- $\mathcal{P}(\bot_D)$
- *For any ascending chain $(x_i)_{i\in\mathbb{N}} \subseteq D$:*

$$(\forall i \in \mathbb{N} : \mathcal{P}(x_i)) \;\Rightarrow\; \mathcal{P}(\bigsqcup_{i\in\mathbb{N}}(x_i))$$

Thus, we need to verify two properties at each type σ:

1. $(\bot_{\mathbb{D}_1^\sigma}, \ldots, \bot_{\mathbb{D}_{k+1}^\sigma}) \in S_{k+1}^\sigma$, where $\mathbb{D}_i^\sigma$ is the denotation of the type σ in the model $\mathcal{M}_i$, for each $i \in \{1, \ldots, k+1\}$.
2. If $\{\boldsymbol{x}^i = (x_1^i, \ldots, x_{k+1}^i) \in S_{k+1}^\sigma \mid i \in \mathbb{N}\}$ is an ascending chain in S_{k+1}^σ, then $\sqcup\{\boldsymbol{x}^i\} \in S_{k+1}^\sigma$.

[7] which follows from their being continuous.

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

- If σ is a ground type then (1) on the page before holds by the definition of S^{σ}_{k+1} at ground types. To prove (2), suppose the sequence $\{\boldsymbol{x}^i\}$ is given. For each $i \geq 0$, there exists an index $j_i \leq k$ such that $\forall m \leq k+1\colon x^i_{j_i} \sqsubseteq x^i_m$, because $\boldsymbol{x}^i \in S^{\sigma}_{k+1}$ and σ is ground. So, in particular, there must be an index $\ell \leq k$ for which there are infinitely many i's — e.g. elements of an infinite set $A \subseteq \mathbb{N}$ — with $\forall m \leq k+1\colon x^i_\ell \sqsubseteq x^i_m$. Hence we have:

  $$\forall m \leq k+1\colon \sqcup\{x^i_\ell \mid i \in \mathbb{N}\} = \sqcup\{x^i_\ell \mid i \in A\} \sqsubseteq \sqcup\{x^i_m \mid i \in A\} = \sqcup\{x^i_m \mid i \in \mathbb{N}\}$$

  which means $\sqcup\{\boldsymbol{x}^i \mid i \in \mathbb{N}\} \in S^{\sigma}_{k+1}$.

- If $\sigma = \sigma_1 \to \sigma_2$ then both (1) and (2) hold by induction hypothesis on σ_2.

Proposition 3.2.6. *For any $k \geq 1$, S_{k+1} is a C-logical relation, where C is the set of* wRPCF*-constants* $w\mathcal{RC}_{\mathrm{A}}$[8].

Proof. It remains to show the proof for the constants $Y_\sigma\colon (\sigma \to \sigma) \to \sigma$. To prove that Y_σ preserves S^{σ}_{k+1}, by Equation (3.2) on the page before it suffices to show three properties at each type σ:

1. $(\perp_{\mathbb{D}^{\sigma}_1}, \ldots, \perp_{\mathbb{D}^{\sigma}_{k+1}}) \in S^{\sigma}_{k+1}$, where $\mathbb{D}^{\sigma}_i$ is the denotation of the type σ in the model $\mathcal{M}_i$, for each $i \in \{1, \ldots, k+1\}$.

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

   $$(F_1 x_1, \ldots, F_{k+1} x_{k+1}) \in S^{\sigma}_{k+1}$$

3. If $\{\boldsymbol{x}^i = (x^i_1, \ldots, x^i_{k+1}) \in S^{\sigma}_{k+1} \mid i \in \mathbb{N}\}$ is an ascending chain in S^{σ}_{k+1}, then $\sqcup\{\boldsymbol{x}^i\} \in S^{\sigma}_{k+1}$.

We have already shown that (1) and (3) hold at each type σ, and (2) holds by the definition of logical relations. □

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

[8]Definition 3.1.1 on page 64.

Lemma 3.2.7. *Let*

$$f : \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \to \mathbb{D}_0$$

be a first-order function in the interval domain model of RPCF, i. e.

$$\forall j \in \{0, \dots, n\} : \mathbb{D}_j \in \{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\}$$

Then the following are equivalent:

1. *f is Vuillemin-sequential.*
2. *f is invariant under all S_{k+1}'s, $(k \geq 1)$.*

Proof. **(2 ⇒ 1)** : Consider f as in the statement of the theorem and assume that it preserves all S_{k+1}'s, $(k \geq 1)$. We prove that f is Vuillemin-sequential by contradiction. So suppose f is *not Vuillemin-sequential* at a point $\boldsymbol{x} = (x_1, \dots, x_n)$ in its domain. Define:

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

As $A \neq \emptyset$[9], without loss of generality let us suppose $A = \{1, \dots, k\}$, $k \leq n$. For any $j \in A$, there must exist an $\boldsymbol{x}^j = (x^j_1, \dots, x^j_n)$ such that:

(i) $x^j_j = x_j$
(ii) $x^j_i = x_i$, for all $i > k$, if any[10].
(iii) $\boldsymbol{x}^j \sqsupset \boldsymbol{x}$
(iv) $f(\boldsymbol{x}^j) \sqsupset f(\boldsymbol{x})$

Now consider the $(k+1) \times n$ matrix X of Table 3.2 on the following page defined by:

$$\mathsf{X}_{i,j} = \begin{cases} x^i_j & \text{if} \quad i \leq k \\ x_j & \text{if} \quad i = k+1 \end{cases}$$

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

- If $j \leq k$ then $\forall m \leq k+1 : \mathsf{X}_{j,j} = x^j_j = x_j \sqsubseteq x^m_j = \mathsf{X}_{m,j}$
- If $j > k$, then $\mathsf{X}_{1,j} = \mathsf{X}_{2,j} = \cdots = \mathsf{X}_{k+1,j}$

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

Table 3.2 The matrix X

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

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

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

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

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

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

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

$$f(x_{j(1)}^1, \ldots, x_{j(i_0)}^{i_0}, \ldots, x_{j(n)}^n) = f(x_{j(i_0)}^1, \ldots, x_{j(i_0)}^{i_0}, \ldots, x_{j(i_0)}^n) \tag{3.3}$$

because

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

and the two vectors agree on their i_0-th components. On the other hand for any $j \neq j(i_0)$:

$$f(x_j^1, \ldots, x_j^n) \sqsupseteq f(x_{j(1)}^1, \ldots, x_{j(n)}^n) \tag{3.4}$$

as

$$(x_j^1, \ldots, x_j^n) \sqsupseteq (x_{j(1)}^1, \ldots, x_{j(n)}^n)$$

Thus, by Equations (3.3) and (3.4) on the facing page:

$$\forall j \leq k+1 \colon f(x^1_{j(i_0)}, \ldots, x^n_{j(i_0)}) \sqsubseteq f(x^1_j, \ldots, x^n_j)$$

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

Combining Proposition 3.2.6 and Lemma 3.2.7 on pages 68–69, we obtain the main result of this section:

Theorem 3.2.8. *Let* $f \colon \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \to \mathbb{D}_0$ *be a first-order* wRPCF-definable *function in the interval domain model of RPCF, i. e.*

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

Then f is Vuillemin-sequential.

Therefore, by virtue of the theorem above, functions like *parallel-or* are ruled out from being definable in wRPCF (see Remark 3.2.4 on page 66).

Although we will show that not all unary first-order functions of the interval domain model are definable in wRPCF[11], they are Vuillemin-sequential[12] and adding them to the language does not affect the sequentiality:

Corollary 3.2.9. *Let*

1. $\mathbb{D}_1$ *and* $\mathbb{D}_2$ *range over the set* $\{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\}$.

2. $\Gamma = w\mathcal{RC}_{\mathrm{A}} + C$*, where for any* $c \in C$, $[\![c]\!] \colon \mathbb{D}_1 \to \mathbb{D}_2$ *is a continuous unary first-order function.*

3. wRPCF_Γ *denote the language built upon the constants in* Γ.

Then any first-order wRPCF_Γ*-definable function is Vuillemin-sequential.*

Proof. For any $c \in C, [\![c]\!]$ is monotone — as it is continuous — hence preserves S_{k+1} for any k. Now Lemma 3.2.7 on page 69 is applicable. □

[11] see Lemma 3.3.10 on page 78 and Theorem 3.3.16 on page 86.
[12] by Remark 3.2.2 on page 65.

Note 3.2.10. *In general, logical relations are handy tools for studying the behaviour of (first-order) functions definable in extensions of the λ-calculus. As a simple example, one can show that excluding if_{bool} and if_{nat} from* wRPCF *leaves us with a language in which all functions are essentially **unary**:*

Let $\Delta := w\mathcal{RC}_{\mathbb{A}} \setminus \{if_{bool}, if_{nat}\}$, and wRPCF_Δ *be the fragment of* wRPCF *built upon the set of constants Δ. For any $k \geq 1$, let T_{k+1} be the $(k+1)$-ary logical relation defined at the ground types $o \in \{bool, nat, I\}$ by:*

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

then similar to the proof of the main theorem, it can be shown that wRPCF_Δ*-definable functions preserve T_{k+1} for any $k \geq 1$. Now using this, one can get a model theoretic proof of the following simple fact:*

> *Let $f\colon \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \to \mathbb{D}_0$ be a first-order* wRPCF_Δ*-definable function, i. e.*
>
> $$\forall j \in \{0, \ldots, n\} \colon \mathbb{D}_j \in \{\mathbb{B}_\perp, \mathbb{N}_\perp, \mathcal{I}\}$$
>
> *Then for some $i \in \{1, \ldots, n\}$, there exists a* wRPCF_Δ*-definable $f_i \colon \mathbb{D}_i \to \mathbb{D}_0$ such that :*
>
> $$f = f_i \circ \pi_i$$
>
> *where $\pi_i \colon \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \to \mathbb{D}_i$ is the projection over the i-th component.*

In words, first-order wRPCF_Δ*-definable functions are essentially functions of one argument. The counter-example of Table 3.3 below shows how if_{nat} does not preserve T_3, where the first three columns on the left are elements of T_3, while the rightmost column — the result of applying if_{nat} to the elements of the first three — is not.*

Table 3.3 ***if_{nat}* does not preserve *T_3***

tt	0	1	$\xrightarrow{if_{nat}}$	0
ff	1	0	$\xrightarrow{if_{nat}}$	0
tt	1	0	$\xrightarrow{if_{nat}}$	1

3.3 Piece-wise Affinity

In this section we derive another non-definability result in a fragment of RPCF which contains wRPCF[13] as a sub-fragment. For that we need to clarify some terms and definitions:

Definition 3.3.1 (piece-wise affine).

- *Let* $-\infty \leq p \leq q \leq +\infty$ *and* $f\colon [p,q] \to \mathbb{R}$*. We say that* f *is* ***affine*** *if and only if:*
$$\exists m, n \in \mathbb{R}\, \forall x \in [p,q] \colon f(x) = mx + n \tag{3.5}$$
- *A continuous function* f *is said to be* ***piece-wise affine*** *if and only if for some:*
$$\{p_0, \ldots, p_i, p_{i+1}, \ldots, p_n\} \subseteq [p,q]$$
such that:
$$p = p_0 \leq \cdots \leq p_i \leq p_{i+1} \leq \cdots \leq p_n = q$$
$f \upharpoonright [p_i, p_{i+1}]$ *— the restriction of* f *to* $[p_i, p_{i+1}]$ *— is affine for all* $0 \leq i \leq n-1$.

Now take $a = [r,s] \in \mathcal{I}$ and consider $[\![cons_a]\!]$ acting on the maximal elements of $\mathcal{I}$, i. e. $[0,1]$. By Equation (2.3) on page 56 we have:

$$\forall x \in [0,1] \colon [\![cons_a]\!](x) = (s-r)x + r$$

therefore by taking:

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

in Equation (3.5) above, we observe that $[\![cons_a]\!]$ acts as an affine (hence trivially piece-wise affine) function over the maximal elements of $\mathcal{I}$.

Assuming $r \neq s$ let us take a look at how $[\![tail_a]\!]$ acts over $[0,1]$. According to Equation (2.4) on page 56, considering:

$$(p_0 \coloneqq 0) \;\leq\; (p_1 \coloneqq r) \;\leq\; (p_2 \coloneqq s) \;\leq\; (p_3 \coloneqq 1)$$

we see that by taking:

1. $m \coloneqq 0 \qquad n \coloneqq 0$

[13] Definition 3.1.2 on page 64

Figure 3.1 The Sierpinski space $\mathbb{2}$

$$\begin{tikzcd} \top \arrow[d, dash] \\ \bot \end{tikzcd}$$

2. $m \coloneqq 1/(s-r) \qquad n \coloneqq 0$

3. $m \coloneqq 0 \qquad n \coloneqq 1$

in Equation (3.5) on the page before, $[\![tail_a]\!]$ is indeed affine over:

1. $[p_o, p_1] = [0, r]$

2. $[p_1, p_2] = [r, s]$

3. $[p_2, p_3] = [s, 1]$

respectively. Hence $[\![tail_a]\!]$ is piece-wise affine over $[0, 1]$.

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

Definition 3.3.2 (Sierpinski space). *We call the flat cpo* $(\mathbb{2}, \sqsubseteq)$ *where:*

$$\begin{array}{l} \mathbb{2} = \{\bot, \top\} \\ \forall x, y \in \mathbb{2} \colon\ x \sqsubseteq y \iff (x = y \ \vee\ x = \bot) \end{array}$$

the ***Sierpinski space****. See Figure 3.1.*

Definition 3.3.3 (weak parallel-or). $\widehat{\mathsf{wpor}} \colon \mathbb{2} \times \mathbb{2} \to \mathbb{2}$ *is defined by:*

$$\forall (a, b) \in \mathbb{2} \times \mathbb{2} \colon\ \widehat{\mathsf{wpor}}(a, b) = \begin{cases} \top & \textit{if } (a = \top) \vee (b = \top) \\ \bot & \textit{otherwise} \end{cases}$$

Definition 3.3.4 (weakly-parallel RPCF : wPR).

1. *The set* $\mathbb{T}_{wPR}$ *of wPR-types are generated by the grammar:*

$$\sigma ::= \S \mid bool \mid nat \mid I \mid \sigma \to \sigma$$

2. $\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{wPR}\}$ *is the collection of domains for wPR if:*

$$\begin{array}{lcl} \mathbb{D}_\S & = & \mathbb{2} \\ \mathbb{D}_{bool} & = & \mathbb{B}_\bot \\ \mathbb{D}_{nat} & = & \mathbb{N}_\bot \\ \mathbb{D}_I & = & \mathcal{I} \\ \mathbb{D}_{\sigma\to\tau} & = & [\mathbb{D}_\sigma \to \mathbb{D}_\tau] \end{array}$$

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

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

where $w\mathcal{RC}_{\mathrm{A}}$ *is the set of* wRPCF*-constants as in Definition 3.1.1 on page 64.*

4. *By* ***wPR*** *we mean the extension of* wRPCF *built upon* $w\mathcal{PRC}_{\mathrm{A}}$*, the immediate reduction rules of which are those of* wRPCF *extended with those of Table 3.4 below. and whose denotational semantics is that of* wRPCF *extended*

Table 3.4 The reduction rules of weak-parallel-or

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

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

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

with the following interpretation of the new constants:[14]

$$\begin{array}{rcl} [\![wpor]\!] & = & \widehat{\mathsf{wpor}} \\ [\![\star]\!] & = & \top \end{array}$$

where $\widehat{\mathsf{wpor}}$ *is the function defined in Definition 3.3.3 on the facing page.*

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

Note 3.3.5. *In Definition 3.3.4 on the page before we just mentioned the basics of wPR, so the exact definition is left to the reader, as we believe with the material presented here, a method similar to that of defining RPCF (Section 2.5) would lead to a complete definition for wPR in a straightforward manner.*

Remark 3.3.6. *wPR as it is cannot be regarded as a fragment of RPCF due to the presence of the type § and the constants that come with it.*

Definition 3.3.7 ($\underline{x}, \overline{x}, m(x), \mu(x)$). *Whenever x denotes an interval, by $\underline{x}$, $m(x)$, $\overline{x}$ and $\mu(x)$ we mean the left end-point, the middle point, the right end-point and the length of that interval, respectively. In other words, we presume:*

$$\begin{aligned} \underline{x} &= \inf \{y \mid y \in x\} \\ \overline{x} &= \sup \{y \mid y \in x\} \\ m(x) &= (\underline{x} + \overline{x})/2 \\ \mu(x) &= \overline{x} - \underline{x} \end{aligned}$$

For each $k \geq 1$, we define a k-ary logical relation R_k which — to some extent — carries the meaning of piece-wise affinity on first-order functions:

- When $o \in \{\S, bool, nat\}$, R_k^o is defined as:

$$\forall (x_1, \ldots, x_k) \in (\mathbb{D}_o)^k \colon (x_1, \ldots, x_k) \in R_k^o \iff \exists i \leq k \, \forall j \leq k \colon x_i \sqsubseteq x_j$$

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

$$\boxed{\boldsymbol{x} \in R_k^{\mathcal{I}} \iff P_1(\boldsymbol{x}) \vee (P_{2a}(\boldsymbol{x}) \wedge P_{2b}(\boldsymbol{x}))}$$

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

  – $P_1(\boldsymbol{x}) \iff \exists i \in \{1, \ldots, k\} \, \forall j \in \{1, \ldots, k\} \colon x_i \sqsubseteq x_j$

  – $$\begin{aligned} P_{2a}(\boldsymbol{x}) \iff & \big(\forall i \in \{1, \ldots, k-1\} \colon \overline{x_i} = \underline{x_{i+1}}\big) \\ \wedge & \big(\forall i < j \in \{1, \ldots, k\} : \\ & \qquad \mu(x_i)\,\mu(x_j) \neq 0 \Rightarrow \forall \ell \in \{i, \ldots, j\} : \mu(x_\ell) \neq 0\big) \end{aligned}$$

  – $$\begin{aligned} P_{2b}(\boldsymbol{x}) \iff & \exists d > 0 \, \forall i \in \{2, \ldots, k-1\} \colon \\ & \qquad \mu(x_{i-1})\,\mu(x_{i+1}) \neq 0 \Rightarrow \mu(x_i) = d \end{aligned}$$

  Intuitively, one may say:

- $P_1(\boldsymbol{x}) \iff$ one coordinate is smaller than all the other ones.
- $P_{2a}(\boldsymbol{x}) \iff$ consecutive intervals join up and non-maximal intervals may not surround maximal ones.
- $P_{2b}(\boldsymbol{x}) \iff$ in the non-maximal section, all intervals are of the same length, except probably for the first and the last one.

Definition 3.3.8 ($\boldsymbol{R_k}$). *R_k is the logical relation generated by the ground type cases on the facing page.*

The aim is to show that R_k is C-logical, where C is the set $w\mathcal{PRC}_\mathrm{A}$ of Definition 3.3.4 on page 75. As before, the tricky part is to show that the set of R_k invariant elements of $\mathcal{I}^k$ forms an inclusive predicate.

Lemma 3.3.9. *If $\{\boldsymbol{x}^i = (x^i_1, \ldots, x^i_k) \in R^I_k \mid i \in \mathbb{N}\}$ is an ascending chain then*

$$\boldsymbol{x} = \sqcup\{\boldsymbol{x}^i \mid i \in \mathbb{N}\} \in R^I_k$$

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

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

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

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

Case (b) $\exists n_0 \in \mathbb{N}\, \forall i \geq n_0 \colon P_{2a}(\boldsymbol{x}^i) \wedge P_{2b}(\boldsymbol{x}^i)$. In particular $P_{2a}(\boldsymbol{x}^{n_0}) \wedge P_{2b}(\boldsymbol{x}^{n_0})$. Hence for some $d_{n_0} > 0$ one has:

$$\forall j \in \{2, \ldots, k-1\} \colon \mu(x^{n_0}_{j-1})\,\mu(x^{n_0}_{j+1}) \neq 0 \Rightarrow \mu(x^{n_0}_j) = d_{n_0}$$

Now take $\boldsymbol{x} = (x_1, \ldots, x_k) := \sqcup\{\boldsymbol{x}^i | i \in \mathbb{N}\}$. As:

(a) $\{\boldsymbol{x}^i \mid i \in \mathbb{N}\}$ is ascending.

(b) $\forall i \geq n_0 \colon P_{2a}(\boldsymbol{x}^i)$

we have:

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

which entails that:

$$\forall j \in \{2, \ldots, k-1\} \colon \mu(x_{j-1})\,\mu(x_{j+1}) \neq 0 \Rightarrow \mu(x_j) = d_{n_0}$$

therefore $P_{2b}(\boldsymbol{x})$. To finish, we notice that:

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

hence $P_{2a}(\boldsymbol{x})$. □

Lemma 3.3.10. *R_k is C-logical, where C is the set of wPR constants $w\mathcal{PRC}_{\mathrm{A}}$.*

Proof. Let us check the more interesting constants:

1. $c \in C$ is the constant $tail_a \colon I \to I$ for some non-maximal $a \in \mathcal{I}$, and $(x_1, \ldots, x_k) \in R_k^I$. There are two cases:

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

   $$P_1(tail_a(x_1), \ldots, tail_a(x_k))$$

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

   $$\begin{cases} \forall j < i_1 \colon tail_a(x_j) = [0, 0] \\ \forall j > i_2 \colon tail_a(x_j) = [1, 1] \end{cases}$$

   If $d > 0$ makes $P_{2b}(x_1, \ldots, x_k)$ true, then

   $$\frac{d}{\overline{a} - \underline{a}}$$

   will make $P_{2b}(tail_a(x_1), \ldots, tail_a(x_k))$ true (see Equation (2.4) on page 56). $P_{2a}(tail_a(x_1), \ldots, tail_a(x_k))$ holds trivially. Note that in case $i_1 = i_2$ then the choice of d is irrelevant as $P_{2b}(tail_a(x_1), \ldots, tail_a(x_k))$ will trivially hold.

2. $c \in C$ is the constant $head_r \colon I \to bool$ for some $r \in \mathbb{Q} \cap (0, 1)$ and $\boldsymbol{x} = (x_1, \ldots, x_k) \in R_k^I$. Again there are two cases to consider:

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

$$P_1(head_r(x_1), \ldots, head_r(x_k))$$

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

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

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

ii. $r < \underline{x_1}$: we have

$$(head_r(x_1), \ldots, head_r(x_k)) = (\mathit{ff}, \ldots, \mathit{ff}) \in R_k^{bool}$$

iii. $\exists i \leq k \colon r \in x_i$: in this case $head_r(x_i) = \perp$ so

$$\forall j \leq k \colon head_r(x_i) \sqsubseteq head_r(x_j)$$

which implies

$$(head_r(x_1), \ldots, head_r(x_k)) \in R_k^{bool}$$

3. $c \in C$ is a fix point constant Y_σ : Using Lemma 3.3.9 on page 77, the proof in this case is by induction over σ as was done before in Proposition 3.2.6 on page 68 for S_{k+1}'s.

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

□

Definition 3.3.11. *We define $\mathcal{I}_{max}$ to be the set of maximal elements of $\mathcal{I}$, which is isomorphic to* $[0, 1]$ *via the following pair of functions:*

$$\begin{cases} e_{\mathcal{I}} : [0,1] \to \mathcal{I}_{max} \\ \qquad r \mapsto [r, r] \end{cases} \qquad\qquad \begin{cases} p_{\mathcal{I}} : \mathcal{I}_{max} \to [0,1] \\ \qquad [r, r] \mapsto r \end{cases}$$

Lemma 3.3.12. *Let $f : \mathcal{I} \to \mathcal{I}$ be continuous and total, i. e.*

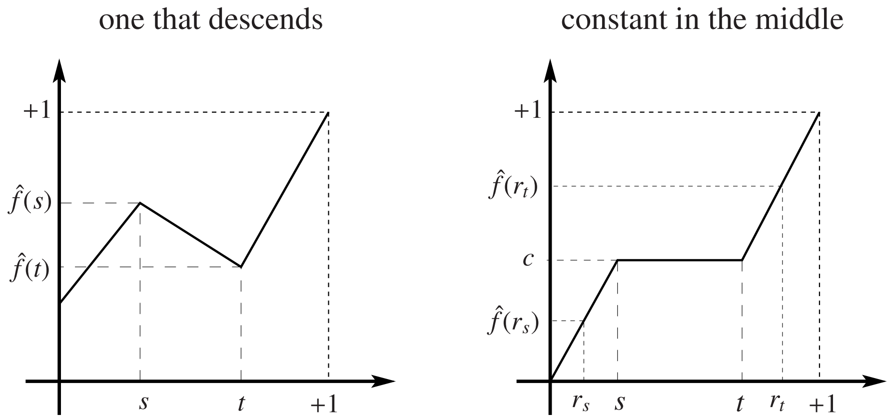
$$\forall x \in \mathcal{I}_{max} : f(x) \in \mathcal{I}_{max}$$

and define:

$$\begin{aligned} \hat{f} : [0,1] &\to [0,1] \\ x &\mapsto (p_{\mathcal{I}} \circ f \circ e_{\mathcal{I}})(x) \end{aligned}$$

Then the following hold (see Figure 3.2 on the following page):

Figure 3.2 Examples of non-wPR-definable functions



1. *If for two points $s, t \in [0, 1]$:*

$$(s < t) \wedge (\hat{f}(s) > \hat{f}(t))$$

then f does not preserve some R_k.

2. *Assume that for some interval $[s, t] \subseteq [0, 1]$ one has:*

   (a) $0 < s < t < 1$
   (b) $\exists c \in \mathbb{R}\, \forall x \in (s, t) : \hat{f}(x) = c$
   (c) $\exists r_s, r_t \in [0, 1] : (r_s < s) \wedge (t < r_t) \wedge \left[(c - \hat{f}(r_s))\, (c - \hat{f}(r_t)) \neq 0 \right]$

   Then f does not preserve some R_k.

Proof.

1. $\hat{f}$ is continuous and $[0, 1]$ is a compact subset of $\mathbb{R}$. Therefore $\hat{f}$ is uniformly continuous over $[0, 1]$. First let

$$\epsilon_1 = \frac{\hat{f}(s) - \hat{f}(t)}{2}$$

and consider $\delta_1 > 0$ for which we have:

$$\forall x, y \in [0, 1] : |x - y| < \delta_1 \Rightarrow |\hat{f}(x) - \hat{f}(y)| < \epsilon_1$$

On the other hand by the intermediate-value theorem we know that there exists a δ_2 such that:

$$\forall x \in [s, s+\delta_2]\, \forall y \in [t-\delta_2, t] : \hat{f}(x) > \hat{f}(y) \tag{3.6}$$

Now choose $N \in \mathbb{N}$ large enough to make

$$\frac{1}{N} < \min\{\delta_1, \delta_2/2\} \tag{3.7}$$

true and let $\boldsymbol{a} = (a_1, \ldots, a_N) \in R_N^I$ be defined by

$$\forall i \in \{1, \ldots, N\} : a_i = [(i-1)/N, i/N]$$

Under f, this element of R_N^I is sent to some $\boldsymbol{b} = (b_1, \ldots, b_N)$. By Equation (3.7) above we have $1/N < \delta_1$. Therefore:

$$\neg\,(\exists i \in \{1, \ldots, N\}\, \forall j \in \{1, \ldots, N\} : b_i \sqsubseteq b_j)$$

and $P_1(\boldsymbol{b})$ *cannot* hold. On the other hand $1/N < \delta_2/2$, so there exist i_0, $j_0 \in \{1, \ldots, N\}$ such that:

$$(i_0 < j_0) \wedge (a_{i_0} \subseteq [s, s+\delta_2]) \wedge (a_{j_0} \subseteq [t-\delta_2, t])$$

Thus, by Equation (3.6) above we have

$$\overline{f(b_{i_0})} > \underline{f(b_{j_0})}$$

As a result, $P_{2a}(\boldsymbol{b})$ does not hold either.

2. We assume $\hat{f}(r_s) < c < \hat{f}(r_t)$, otherwise by part 1 of the lemma we get the result. Let us take

$$\epsilon_1 = \frac{\hat{f}(r_t) - \hat{f}(r_s)}{2}$$

As $\hat{f}$ is uniformly continuous over $[0, 1]$, there exists a $\delta_1 > 0$ such that

$$\forall x, y \in [0, 1] : |x-y| < \delta_1 \Rightarrow |\hat{f}(x) - \hat{f}(y)| < \epsilon_1$$

Now take a sufficiently large $N \in \mathbb{N}$ for which we have:

$$\frac{1}{N} < \min\{\delta_1, (t-s)/2\}$$

and consider $\boldsymbol{a} = (a_1, \ldots, a_N) \in R_N^I$ defined by

$$\forall i \in \{1, \ldots, N\} : a_i = [(i-1)/N, i/N]$$

Under f, the vector $\boldsymbol{a}$ is sent to some $\boldsymbol{b} = (b_1, \ldots, b_N)$. We chose N such that $1/N < \delta_1$, therefore:

$$\neg\,(\exists i \in \{1, \ldots, N\}\, \forall j \in \{1, \ldots, N\} : b_i \sqsubseteq b_j)$$

and $P_1(\boldsymbol{b})$ *cannot* hold. On the other hand, as $1/N < (t-s)/2$ there exist i_0, k such that

$$\forall j \in \{i_0, \ldots, i_0 + k\} : a_j \subseteq (s, t) \tag{3.8}$$

We also have:

$$\exists j_0, j_1 \in \{1, \ldots, N\} : (r_s \in a_{j_0}) \wedge (r_t \in a_{j_1}) \tag{3.9}$$

Therefore from Equations (3.8) and (3.9) we get:

1. $\forall j \in \{i_0, \ldots, i_0 + k\} : b_j = [c, c]$, i. e. b_j is maximal.
2. $\exists \ell_s \in \{1, \ldots, i_0\} : b_{\ell_s}$ is not maximal.
3. $\exists \ell_t \in \{i_0 + k, \ldots, N\} : b_{\ell_t}$ is not maximal.

In other words, we have got maximal elements surrounded by non-maximal ones. This violates P_{2a}, hence $P_{2a}(\boldsymbol{b})$ cannot hold. □

There are certain issues to be addressed regarding Lemma 3.3.12 on page 79. The logical relations R_k do not by any means characterize piecewise affinity, as functions such as $\mathsf{neg}\colon \mathcal{I} \to \mathcal{I}$:

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

which are obviously affine do not preserve all R_k's — in this case, because neg is not increasing. Thus on one hand we have a strong result which shows that even affinity does not entail wPR-definability. On the other hand, non-affine functions are highly unlikely to preserve all R_k's. Take $f\colon \mathcal{I} \to \mathcal{I}$ defined by:

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

and consider:

$$\boldsymbol{a} = (a_1, \ldots, a_4) \in R_4^I$$

where

$$\forall i \in \{1, \ldots, 4\}\colon a_i = [(i-1)/4, i/4]$$

Under f, this element of R_4^I is sent to:

$$\boldsymbol{b} = (b_1, \ldots, b_4)$$

where

$$\forall i \in \{1, \ldots, 4\}\colon b_i = [(i-1)^2/16, i^2/16]$$

Hence in particular we get:

$$\begin{cases} \mu(b_2) = 3/16 \\ \mu(b_3) = 5/16 \end{cases}$$

Therefore $\boldsymbol{b} = (b_1, \ldots, b_4) \notin R_4^I$.

Although we have looked at a special case, it suggests a uniform approach to showing non-affine functions **not preserving** some R_k. Yet instead of going down that route, we try to prove a much stronger statement. We show that unary total wPR-definable functions over real numbers are essentially of the same shape as that of Cons and Tail, i. e. "constant-linear-constant". In order to get there we need to prove a few lemmas on the way:

Lemma 3.3.13.

1. *Let $r, s \in \mathbb{R}$ with $r < s$ and assume*

   $$f : [r, s] \to \mathbb{R}$$

   is continuous and satisfies:

   (a) $f(r) = f(s) = 0$

   (b) $\forall x, y \in [r, s] : f(x) = f(y) = 0 \Rightarrow f((x + y)/2) = 0$

   Then $\forall x \in [r, s] : f(x) = 0$.

2. *In part 1, the interval $[r, s]$ can be replaced by any of $(r, s]$, $[r, s)$ or (r, s).*

Proof.

1. Without loss of generality we assume $r = 0$ and $s = 1$. By clause (b) in the statement of the lemma we derive that for all dyadic $d \in [0, 1]$ one has $f(d) = 0$. Then using the continuity of f and the fact that the set of dyadic numbers between 0 and 1 is dense in $[0, 1]$ we get the result.

2. For $(r, s]$ one only needs to consider the restrictions of f to each element of the sequence $(A_n)_{n\in\mathbb{N}}$ of intervals defined as follows:

$$\forall n \in \mathbb{N} : A_n \coloneqq [r + (s - r)/2^{n+1}, s]$$ □

Corollary 3.3.14.

1. Let $r, s \in \mathbb{R}$ with $r < s$ and assume $f : [r, s] \to \mathbb{R}$ is continuous and satisfies

$$\forall x, y \in [r, s] : f((x + y)/2) = \frac{f(x) + f(y)}{2}$$

Then f is affine.

2. In part 1, the interval $[r, s]$ can be replaced by any of $(r, s]$, $[r, s)$ or (r, s).

Proof.

1. If we define $g : [r, s] \to \mathbb{R}$ by

$$g(x) = f(x) - \frac{f(s) - f(r)}{s - r}(x - r) - f(r)$$

then g satisfies the conditions of Lemma 3.3.13 on the preceding page, therefore

$$\forall x \in [r, s] : g(x) = 0$$

Hence we get:

$$\forall x \in [r, s] : f(x) = \frac{f(s) - f(r)}{s - r}(x - r) + f(r)$$

2. Left to the reader. □

Let us seek the main property stated in Corollary 3.3.14 above inside functions which preserve R_k's:

Lemma 3.3.15. *Let $f : \mathcal{I} \to \mathcal{I}$ be continuous and total, i. e.*

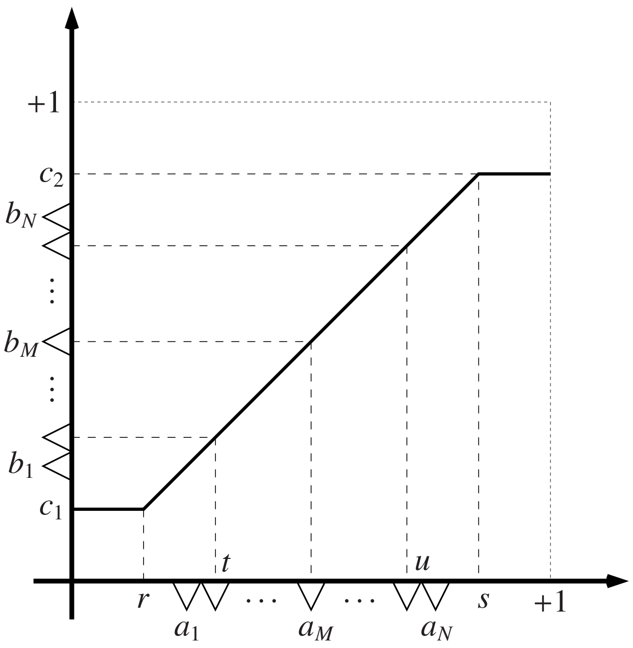
$$\forall x \in \mathcal{I}_{max} : f(x) \in \mathcal{I}_{max}$$

and define:

$$\begin{aligned} \hat{f} : [0, 1] &\to [0, 1] \\ x &\mapsto (p_{\mathcal{I}} \circ f \circ e_{\mathcal{I}})(x) \end{aligned}$$

Now assume that each R_k is preserved by f. Then either $\hat{f}$ is constant or otherwise there exist $r, s, c_1, c_2 \in [0, 1]$ such that $r < s, c_1 < c_2$ and we have:

Figure 3.3 The shape of R_k preserving functions



1. $\forall x \in [0, r] : \hat{f}(x) = c_1$
   $\forall x \in [s, 1] : \hat{f}(x) = c_2$
   $\forall x, y \in (r, s) : x < y \Rightarrow c_1 < \hat{f}(x) < \hat{f}(y) < c_2$

2. *In addition to part 1, one gets:*

$$\forall t, u \in (r, s) : \hat{f}((t + u)/2) = \frac{\hat{f}(t) + \hat{f}(u)}{2}$$

Proof.

1. Follows from Lemma 3.3.12 on page 79. We leave the details to the reader.

2. Without loss of generality assume that $t < u$, (see Figure 3.3). Let

$$\epsilon = \min\{t - r, s - u\}$$

and let k_0 be some sufficiently large natural number for which

$$(u - t)/2^{k_0} < \epsilon/2$$

and assign:

$$\begin{aligned} d &= (u - t)/2^{k_0} \\ N_0 &= 2^{k_0} + 3 \\ M &= 2^{k_0 - 1} + 2 \end{aligned}$$

Let $\boldsymbol{a} = (a_1, \ldots, a_{N_0}) \in R^I_{N_0}$ be defined by:

$$\forall i \in \{1, \ldots, N_0\} : a_i = [t + (i-2)d - d/2, t + (i-2)d + d/2]$$

It is easy to see that:

(a) $\forall i \in \{1, ..., N_0\} : a_i \subseteq (r, s)$

(b) $(t = m(a_2)) \wedge \big(u = m(a_{N_0 - 1})\big) \wedge \big(\,((t+u)/2) = m(a_M)\,\big)$

Under f, the vector $\boldsymbol{a}$ is sent to some $\boldsymbol{b} = (b_1, \ldots, b_{N_0})$. Again, the following is straightforward and the details are omitted here:

(i) $P_1(\boldsymbol{b})$ does not hold, hence we must have $P_{2a}(\boldsymbol{b}) \wedge P_{2b}(\boldsymbol{b})$.

(ii) $\forall i \in \{1, \ldots, N_0\} : b_i$ is non-maximal.

Now notice that $\hat{f}((t+u)/2) \in b_M$, therefore:

$$\frac{\hat{f}(t) + \hat{f}(u)}{2} - 2\frac{c_2 - c_1}{N_0} \leq \hat{f}((t+u)/2) \leq \frac{\hat{f}(t) + \hat{f}(u)}{2} + 2\frac{c_2 - c_1}{N_0} \qquad (3.10)$$

For any $k \geq k_0$, we could repeat the process with $N = 2^k + 3$ and Equation (3.10) above would hold for any such N. Therefore we get

$$\hat{f}((t+u)/2) = \frac{\hat{f}(t) + \hat{f}(u)}{2} \qquad \square$$

By combining Corollary 3.3.14 and Lemma 3.3.15 we understand that any total unary first-order wPR-definable function over real numbers is of the shape "constant-linear-constant", i. e.

Theorem 3.3.16. *Let $f : \mathcal{I} \to \mathcal{I}$ be continuous and total, i. e.*

$$\forall x \in \mathcal{I}_{max} : f(x) \in \mathcal{I}_{max}$$

and define:

$$\begin{aligned} \hat{f} : [0,1] &\to [0,1] \\ x &\mapsto (p_{\mathcal{I}} \circ f \circ e_{\mathcal{I}})(x) \end{aligned}$$

Then f preserves all R_k's if and only if there exist $r, s, c_1, c_2 \in [0, 1]$ such that

$$(c_1 \leq c_2) \wedge (r < s)$$

and:

$$\forall x \in [0,1] : \hat{f}(x) = \begin{cases} c_1 & \text{if } x \in [0, r] \\ \dfrac{c_2 - c_1}{s - r}(x - r) + c_1 & \text{if } x \in [r, s] \\ c_2 & \text{if } x \in [s, 1] \end{cases} \tag{3.11}$$

In particular:

> *if f is wPR-definable, then $\hat{f}$ is of the shape described in Equation* (3.11) *above.*

We may well be close to characterizing total unary wPR-definable functions over real numbers. That depends on studying the nature of parameters r, s, c_1 and c_2 in Equation (3.11) above. We conjecture that a necessary and sufficient condition for wPR-definability can be obtained by demanding these four parameters to be computable, see Conjecture 6.1.1 on page 173.

But more important than that is to get inspiration from R_k's and work out a logical relation which truly characterizes piece-wise affinity. We conjecture that wPR augmented by any set of primitives denoting piecewise affine functions still maintains the piece-wise affinity. This is the most important outcome of the present section which will hopefully motivate some future work, (see Conjecture 6.1.3 on page 174).

Remark 3.3.17. *None of the following functions:*

$$\begin{array}{lcl} \widehat{\mathsf{por}} & : & \mathbb{B}_\perp \times \mathbb{B}_\perp \to \mathbb{B}_\perp \\ \widehat{\mathsf{pif}}_{bool} & : & \mathbb{B}_\perp \times \mathbb{B}_\perp \times \mathbb{B}_\perp \to \mathbb{B}_\perp \\ \widehat{\mathsf{pif}}_{nat} & : & \mathbb{B}_\perp \times \mathbb{N}_\perp \times \mathbb{N}_\perp \to \mathbb{N}_\perp \\ \widehat{\mathsf{pif}}_{I} & : & \mathbb{B}_\perp \times \mathcal{I} \times \mathcal{I} \to \mathcal{I} \end{array}$$

preserves all the logical relations R_k. Take $\widehat{\mathsf{pif}}_I$ for example. In Table 3.5 below, the left three columns are elements of R_3, whereas the rightmost column — the result of applying $\widehat{\mathsf{pif}}_I$ over the elements of the first three — is not. Hence none of

Table 3.5 $\widehat{\mathsf{pif}}$ does not preserve R_3

tt	$[0,1/3]$	$[0,1/4]$	$\stackrel{\widehat{\mathsf{pif}}_I}{\to}$	$[0,1/3]$
$\perp$	$[1/3,2/3]$	$[1/4,3/4]$	$\stackrel{\widehat{\mathsf{pif}}_I}{\to}$	$[1/4,3/4]$
tt	$[2/3,1]$	$[3/4,1]$	$\stackrel{\widehat{\mathsf{pif}}_I}{\to}$	$[2/3,1]$

them is definable in wPR. As a minor result, we have another confirmation of the fact that:

$$\boxed{\textit{wpor is strictly weaker than por.}}$$

Remark 3.3.18. *Take* wRPCF$^+$*, the extension of* wRPCF *with the constant*

$$avg\colon I \to I \to I$$

interpreted as the maximal continuous extension of the average operator (Equation (3.1) on page 63). Escardó, Hofmann, and Streicher [1998, 2004] show that $\widehat{\mathsf{wpor}}$ *is* wRPCF$^+$*-definable. On the other hand, it is easy to show that* wRPCF$^+$*-definable functions preserve all R_k's, which proves that none of the four* parallel *operations mentioned in Remark 3.3.17 on the page before is* wRPCF$^+$*-definable. Thus:*

Although average is inherently parallel, once added to weak-RPCF does not offer the full (parallel) power of $\widehat{\mathsf{pif}}_I$.

3.4 Conservativity of weak-RPCF Over PCF

It is already known from the previous works done by Escardó et al. [1998, 2004] and the present author's results in [Farjudian, 2003b][15] that without any *parallel* mechanism, even basic first-order functions on real numbers - such as average,

[15] see Sections 3.2 and 3.3.

square function, or in fact any non-sequential or non-affine function - are not definable in Real-PCF. However, one may wonder if it could still be possible to define some new objects in the extensional model of PCF just by adding a data type for real numbers and the sequential constants of Real-PCF. Here in this section we present a **negative** answer to this question. In other words, we show that *the sequential fragment of the Real-PCF — i. e. weak-Real-PCF — is conservative over PCF*.

3.4.1 The Sequence Model

In this section we make use of an auxiliary model for RPCF which has a better connection to the cpo model for PCF. First consider the set

$$\mathcal{I}_{\mathbb{Q}} = \{a \in \mathcal{I} \mid a \text{ has rational end-points}\}$$

By an effective enumeration, $\mathcal{I}_{\mathbb{Q}}$ can be simply written as:

$$\mathcal{I}_{\mathbb{Q}} = \{a_i \mid i \in \mathbb{N}\}$$

It is worth mentioning that by Equations (2.5) and (2.6) on pages 57–58 $\mathcal{I}_{\mathbb{Q}}$ is closed under both product and division of intervals, i. e.

1. $\forall i, j \in \mathbb{N} \colon a_i\, a_j \in \mathcal{I}_{\mathbb{Q}}$
2. $\forall i, j \in \mathbb{N} \colon (a_i \neq a_j) \wedge (a_i \sqsubseteq a_j) \Rightarrow a_j \setminus a_i \in \mathcal{I}_{\mathbb{Q}}$

Definition 3.4.1 (The Sequence Domain: $\mathcal{S}$).
Let $\mathcal{S}$ be the set of all (finite or infinite) sequences over the alphabet

$$\Sigma = \{\mathsf{Cons}_{a_i} \mid i \in \mathbb{N}\}$$

and let

$$\sqsubseteq_{\mathcal{S}} \subseteq \mathcal{S} \times \mathcal{S}$$

*be the prefix order. We call the ω-*algebraic domain $(\mathcal{S}, \sqsubseteq_{\mathcal{S}})$ *— which we simply write as $\mathcal{S}$ — the **Sequence Domain**.*

Notation 3.4.2. *We adopt the Haskell notation and denote a typical sequence as:*

$$[a_1, \ldots, a_n, \ldots]$$

In particular:

1. *The empty sequence is denoted by* [ ].

2. *The sequence of length* 1 *whose only element is d is denoted by* $[d]$.

The 'cons' operator over sequences is denoted by ':'. Thus we have:

$$h : [a_1, \ldots, a_n, \ldots] \coloneqq [h, a_1, \ldots, a_n, \ldots]$$

We consider a new denotational semantics for Real-PCF based on $\mathcal{S}$, which we denote by $[\![.]\!]_{\mathcal{S}}$. In this setting, the denotation of the data type I is $\mathcal{S}$, i. e.

$$[\![I]\!]_{\mathcal{S}} = \mathcal{S}$$

and for the constants we define the denotational semantics using pattern matching as in Table 3.6 on page 103. For the rest of this section, our goal is to demonstrate that:

> Let σ be any PCF-type. Then any wRPCF-definable object $x \in \mathbb{D}_\sigma$ in the interval domain model of RPCF is PCF-definable.

3.4.2 Translating wRPCF into PCF

We start by translating $\mathcal{S}$ into $[\mathbb{N}_\perp \to \mathbb{N}_\perp]$ which is the denotation of the pure PCF type ($nat \to nat$).

- embedding $\mathcal{S}$ into $[\mathbb{N}_\perp \to \mathbb{N}_\perp]$ via $to_P \colon \mathcal{S} \to (\mathbb{N}_\perp \to \mathbb{N}_\perp)$

$$\begin{aligned} to_P[\,] &= \lambda m \colon nat\,.\,\perp \\ to_P(\mathsf{Cons}_{a_i} \colon x) &= \lambda m \colon nat\,.\,if\ Zero(m)\ then\ \bar{i}\ else\ (to_P(x))(pred\ m) \end{aligned}$$

- projecting $(\mathbb{N}_\perp \to \mathbb{N}_\perp)$ onto $\mathcal{S}$ via $to_W \colon (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathcal{S}$

$$to_W(f) = \begin{cases} [\,] & \text{if } f(\bar{0}) \text{ is undefined} \\ \mathsf{Cons}_{a_{f(\bar{0})}} \colon to_W(\lambda m \colon nat\,.\,f(succ\ m)) & \text{otherwise} \end{cases}$$

Proposition 3.4.3. *to_W and to_P form a projection/embedding pair that make $\mathcal{S}$ a computable retract of* $(\mathbb{N}_\perp \to \mathbb{N}_\perp)$.

Proof. Computability is obvious from the definition of to_W and to_P. It is also straightforward to show

$$to_W \circ to_P = id_{\mathcal{S}}$$

using structural induction. As a result we have

$$(to_P \circ to_W)^2 = to_P \circ to_W$$

Now take any $f : \mathbb{N}_\perp \to \mathbb{N}_\perp$. It is not difficult to show that for each $n \geq 0$ we have:

$$(to_P \circ to_W)(f)(n) = \begin{cases} \perp & \text{if } \exists i \leq n : f(i) \text{ is undefined} \\ f(n) & \text{otherwise} \end{cases}$$

Hence $(to_P \circ to_W) \sqsubseteq id_{\mathbb{N}_\perp \to \mathbb{N}_\perp}$. □

We want to extend the previous proposition to all wRPCF types involving I. Let us consider $\xi(x)$ to be the set of all type expression defined by the following grammar:

$$\xi(x) ::= x \mid bool \mid nat \mid \xi(x) \to \xi(x) \tag{3.12}$$

It should be obvious that any Real-PCF type is of the form $\xi(I)$ for some $\xi(x)$.

Now we can phrase our proposition in a more proper way:

Notation 3.4.4. *Throughout this* ***section*** *we simply write* $[\![\,.\,]\!]$ *where we really mean* $[\![\,.\,]\!]_{\mathcal{S}}$*, i.e denotational semantics w. r. t. the sequence model, whenever no confusion is likely to happen.*

Proposition 3.4.5. *For any type expression* $\xi(x)$ *— as in Equation (3.12) above —* $[\![\xi(I)]\!]$ *is a computable retract of* $[\![\xi(nat \to nat)]\!]$ *via a projection/embedding pair*

$$[\![\xi(I)]\!] \underset{p_\xi}{\overset{e_\xi}{\rightleftarrows}} [\![\xi(nat \to nat)]\!]$$

Moreover, $e_\xi \circ p_\xi$ *is PCF-definable.*

Proof. Induction over the structure of $\xi(x)$:

- $\xi(x) = bool$ or $\xi(x) = nat$: obvious.

- $\xi(x) = x$: in this case we have

$$\begin{aligned} &[\![\xi(I)]\!] = \mathcal{S} \\ &[\![\xi(nat \to nat)]\!] = [\mathbb{N}_\perp \to \mathbb{N}_\perp] \end{aligned}$$

hence (to_W, to_P) is the natural choice for the projection/embedding pair. To verify definability of $to_P \circ to_W$ we assume — without loss of generality — that product over natural numbers is strict in its second argument, and by that:

$$to_P \circ to_W = \lambda f\colon (nat \to nat)\,.\,\lambda n\colon nat\,.\,(1 + 0 \times \mathsf{prod}(f, n))f(n)$$

where:

$$\begin{aligned}&\mathsf{prod}\colon (nat \to nat, nat) \to nat\\ &\mathsf{prod}\ (f, 0) = f(0)\\ &\mathsf{prod}\ (f, n) = f(n) \times \mathsf{prod}\ (f, n-1)\end{aligned}$$

Thus, $to_P \circ to_W$ is indeed PCF-definable.

- $\xi(x) = \xi_1(x) \to \xi_2(x)$: by induction hypothesis there are pairs (p_{ξ_1}, e_{ξ_1}) and (p_{ξ_2}, e_{ξ_2}) where:

$$[\![\xi_1(I)]\!] \underset{p_{\xi_1}}{\overset{e_{\xi_1}}{\rightleftarrows}} [\![\xi_1(nat \to nat)]\!]$$

$$[\![\xi_2(I)]\!] \underset{p_{\xi_2}}{\overset{e_{\xi_2}}{\rightleftarrows}} [\![\xi_2(nat \to nat)]\!]$$

and (p_{ξ_i}, e_{ξ_i}) is a projection/embedding pair that makes $[\![\xi_i(I)]\!]$ a computable retract of $[\![\xi_i(nat \to nat)]\!]$, for $i \in \{1, 2\}$.

Now it is again straightforward to check that

$$(p_{\xi_1} \to e_{\xi_2})\colon [\![\xi_1(I) \to \xi_2(I)]\!] \to [\![\xi_1(nat \to nat) \to \xi_2(nat \to nat)]\!]$$

is a computable embedding and

$$(e_{\xi_1} \to p_{\xi_2})\colon [\![\xi_1(nat \to nat) \to \xi_2(nat \to nat)]\!] \to [\![\xi_1(I) \to \xi_2(I)]\!]$$

is a computable projection. Thus we assign:

$$\begin{cases} e_{\xi_1\to\xi_2} = \lambda f\colon \xi_1(I) \to \xi_2(I)\,.\,e_{\xi_2} \circ f \circ p_{\xi_1}\\ p_{\xi_1\to\xi_2} = \lambda f\colon \xi_1(nat \to nat) \to \xi_2(nat \to nat)\,.\,p_{\xi_2} \circ f \circ e_{\xi_1}\end{cases}$$

and observe that:

$$\begin{aligned} e_\xi \circ p_\xi &= e_{\xi_1\to\xi_2} \circ p_{\xi_1\to\xi_2}\\ &= \lambda f\colon \xi_1(nat \to nat) \to \xi_2(nat \to nat)\,.\,(e_{\xi_2} \circ p_{\xi_2}) \circ f \circ (e_{\xi_1} \circ p_{\xi_1})\end{aligned}$$

which is PCF-definable because $e_{\xi_i} \circ p_{\xi_i}$ (for $i = 1, 2$) are PCF-definable by induction hypothesis. □

Next we move on to translating terms. For that, we define a function ϕ_t from wRPCF-terms to PCF-terms. First let us present some basic cases:

(i). For any $i \in \mathbb{N}$:

$$\phi_t(cons_{a_i})\colon (nat \to nat) \to (nat \to nat)$$
$$\phi_t(cons_{a_i}) = \lambda f\colon (nat \to nat)\,.\,\lambda m\colon nat\,.\,if\ Zero(m)\ then\ \bar{i}\ else\ f(pred\ m)$$

(ii). For $\phi_t(tail_{a_i})$'s, again we go back to the one-step reduction rules for *tail* as in Table 2.9 on page 60. Note that if $a = [\underline{a}, \overline{a}]$ and $b = [\underline{b}, \overline{b}]$ are elements of $\mathcal{I}$ with rational end-points, then all the predicates

$$\begin{aligned} p_1(a,b) &= b \le a \\ p_2(a,b) &= a \le b \\ p_3(a,b) &= (a \sqsubseteq b) \wedge (a \neq b) \\ p_4(a,b) &= (a \uparrow b) \wedge (a \not\sqsubseteq b) \wedge (b \not\sqsubseteq a) \wedge (a \not\le b) \wedge (b \not\le a) \end{aligned}$$

can be expressed in terms of $\underline{a}, \overline{a}, \underline{b}, \overline{b}$ and ordinary order relations on rational numbers, hence they are all PCF-definable. Moreover, the expressions:

$$\begin{gathered} b \setminus a \\ (a \sqcup b) \setminus a \\ (a \sqcup b) \setminus b \end{gathered}$$

can also be expressed in terms of $\underline{a}, \overline{a}, \underline{b}, \overline{b}$, ordinary arithmetic and ordinary order relations on rational numbers. Therefore, they are PCF-definable as well. As a result, for any $i \in \mathbb{N}$:

$$\phi_t(tail_{a_i})\colon (nat \to nat) \to (nat \to nat)$$

can be defined by:

$$\begin{aligned} \phi_t(tail_{a_i}) = \lambda f &: (nat \to nat)\,. \\ &\quad if\ p_1(a_i, a_{f(\overline{0})})\ then\ \lambda m\colon nat\,.\,\overline{i_0} \\ &else\ if\ p_2(a_i, a_{f(\overline{0})})\ then\ \lambda m\colon nat\,.\,\overline{i_1} \\ &else\ if\ p_3(a_i, a_{f(\overline{0})})\ then\ \phi_t(cons_{a_{i_2}})(\lambda m\colon nat\,.\,f(succ\ m)) \\ &else\ if\ p_4(a_i, a_{f(\overline{0})})\ then\ \phi_t(cons_{a_{i_3}})(\phi_t(tail_{a_{i_4}})(\lambda m\colon nat\,.\,f(succ\ m))) \\ &else\ \phi_t(tail_{a_i})(g(f)) \end{aligned}$$

where:

1. $a_{i_0} = [0, 1/2]$
2. $a_{i_1} = [1/2, 1]$
3. $a_{i_2} = a_{f(\overline{0})} \setminus a_i$
4. $a_{i_3} = (a_i \sqcup a_{f(\overline{0})}) \setminus a_i$
5. $a_{i_4} = (a_i \sqcup a_{f(\overline{0})}) \setminus a_{f(\overline{0})}$
6. $\begin{cases} g\colon (nat \to nat) \to (nat \to nat) \\ g(f) = \lambda m\colon nat\,.\ if\ Zero\ m\ then\ \overline{j}\ else\ f(succ\ m) \\ \qquad \text{where } a_j = a_{f(\overline{0})} \,.\, a_{f(\overline{1})} \end{cases}$

(iii). For $\phi_t(head_r)$'s too we note that for $r \in \mathbb{Q} \cap (0,1)$ and $a = [\underline{a}, \overline{a}] \in \mathcal{I}$ with rational end-points:

$$\begin{aligned} p_1(a,r) &= a < r = \overline{a} < r \\ p_2(a,r) &= a > r = \underline{a} > r \end{aligned}$$

are obviously PCF-definable. Therefore:

$$\phi_t(head_r)\colon (nat \to nat) \to bool$$

can be defined as:

$$\begin{aligned} \phi_t(head_r) = \lambda f\colon (nat \to nat)\,.\ & if\ p_1(a_{f(\overline{0})}, r)\ then\ true \\ & else\ if\ p_2(a_{f(\overline{0})}, r)\ then\ false \\ & else\ \phi_t(head_r)(g(f)) \end{aligned}$$

where once again g is defined as in 6 above.

(iv). The function

$$\phi_t(if_I)\colon bool \to (nat \to nat) \to (nat \to nat) \to (nat \to nat)$$

is easily defined by:

$$\phi_t(if_I)\ p\ f\ g \ = \ \lambda n\colon nat\,.\, if_{nat}\ p\ then\ f(n)\ else\ g(n)$$

In the next lemma, we define ϕ_t over all wRPCF-terms by induction. In order to simplify the phrasing of the lemma's statement, we fix the following couple of definitions:

Definition 3.4.6 (Translating Sets of Variables). *Consider any set $A \subseteq \mathrm{Var}$ of variables, each of which is assigned a* wRPCF*-type $\xi(I)$. By*

$$\phi_v(A)$$

we mean the same set of variables, this time equipped with suitable PCF-types according to the following rule:

$$\forall x \in \mathrm{Var}: \quad x\colon \xi(nat \to nat) \in \phi_v(A) \iff x\colon \xi(I) \in A$$

Definition 3.4.7 (Translating Environments). *Consider any environment*

$$\rho\colon \mathrm{Var} \to \cup\{\mathbb{D}_\sigma \mid \sigma \text{ a PCF-type}\}$$

assigning to each variable $x\colon \sigma$ an element of the cpo $\mathbb{D}_\sigma$ which is in the extensional model of PCF. Corresponding to each such ρ we define the environment:

$$\psi_{Env}(\rho)\colon \mathrm{Var} \to \cup\{\mathbb{D}_\sigma \mid \sigma \text{ a wRPCF-type}\}$$

such that

$$\forall x\colon \xi(I)\colon\ \psi_{Env}(\rho)(x\colon \xi(I)) = p_\xi \circ \rho(x\colon \xi(nat \to nat))$$

where p_ξ is the projection half of the projection/embedding pair

$$[\![\xi(I)]\!] \underset{p_\xi}{\overset{e_\xi}{\rightleftarrows}} [\![\xi(nat \to nat)]\!]$$

Lemma 3.4.8. *To any* wRPCF*-term $M\colon \xi(I)$ there corresponds a PCF-term*

$$\phi_t(M)\colon \xi(nat \to nat)$$

such that:

1. $\mathrm{FV}(\phi_t(M)) = \phi_v(\mathrm{FV}(M))$
2. *For any PCF-environment*

   $$\rho\colon \mathrm{Var} \to \cup\{\mathbb{D}_\sigma \mid \sigma \text{ a PCF-type}\}$$

   the following equation holds:

   $$[\![M\colon \xi(I)]\!](\psi_{Env}(\rho)) = p_\xi\ ([\![\phi_t(M)\colon \xi(nat \to nat)]\!]\ (\rho)) \tag{3.13}$$

   where p_ξ is the projection half of the projection/embedding pair

   $$[\![\xi(I)]\!] \underset{p_\xi}{\overset{e_\xi}{\rightleftarrows}} [\![\xi(nat \to nat)]\!]$$

Proof. Induction over the structure of M:

- Case 1 : $M\colon \sigma$ is a PCF constant:

$$\phi_t(M) = M\colon \sigma$$

- Case 2 : M is any of the wRPCF constants $cons_{a_i}$, $tail_{a_i}$, $head_r$ or if_I: we have already defined $\phi_t(M)$ for these cases in definitions on pages 93–94.

- Case 3 : $M = x\colon \xi(I)$ is a variable: we take

$$\phi_t(x) = x\colon \xi(nat \to nat)$$

and Equation (3.13) on the page before follows from Proposition 3.4.5 on page 91.

- Case 4 : $M = Y_{\xi(I)}$ is a fix-point constant: in this case $Y_{\xi(nat\to nat)}$ is the natural candidate and we define:

$$\phi_t(Y_{\xi(I)}) = Y_{\xi(nat\to nat)}$$

To see that the choice is a correct one we should verify Equation (3.13) for this special case. Take any $f \in [\![\xi(I) \to \xi(I)]\!]$ and you have:

$$\begin{array}{rcl}
[\![Y_{\xi(I)}]\!](\psi_{Env}(\rho))\,(f) & = & \sqcup\{f^n(\bot_{[\![\xi(I)]\!]}) \mid n \in \mathbb{N}\} \\
(p_\xi \text{ is strict}) & = & \sqcup\{f^n(p_\xi(\bot_{[\![\xi(nat\to nat)]\!]})) \mid n \in \mathbb{N}\} \\
(p_\xi \circ e_\xi = id_{[\![\xi(I)]\!]}) & = & \sqcup\{p_\xi(e_\xi f^n p_\xi)(\bot_{[\![\xi(nat\to nat)]\!]}) \mid n \in \mathbb{N}\} \\
(p_\xi \circ e_\xi = id_{[\![\xi(I)]\!]}) & = & \sqcup\{p_\xi(e_\xi f p_\xi)^n(\bot_{[\![\xi(nat\to nat)]\!]}) \mid n \in \mathbb{N}\} \\
(p_\xi \text{ is continuous}) & = & p_\xi(\sqcup\{(e_\xi f p_\xi)^n(\bot_{[\![\xi(nat\to nat)]\!]}) \mid n \in \mathbb{N}\}) \\
(\text{definition of } e_{\xi\to\xi}) & = & p_\xi(\sqcup\{(e_{\xi\to\xi}(f))^n(\bot_{[\![\xi(nat\to nat)]\!]}) \mid n \in \mathbb{N}\}) \\
(\text{definition of } [\![Y]\!]) & = & p_\xi([\![Y_{\xi(nat\to nat)}]\!](\rho)\,(e_{\xi\to\xi}(f))) \\
(\text{definition of } p_{(\xi\to\xi)\to\xi}) & = & p_{(\xi\to\xi)\to\xi}([\![Y_{\xi(nat\to nat)}]\!](\rho)\,(f)
\end{array}$$

Therefore, as f was arbitrary:

$$[\![Y_{\xi(I)}]\!](\psi_{Env}(\rho)) = p_{(\xi\to\xi)\to\xi}([\![Y_{\xi(nat\to nat)}]\!](\rho))$$

• Case 5 : $M = M_1M_2$ with $M_1 : \xi_1(I) \to \xi(I)$ and $M_2 : \xi_1(I)$: in this case we take

$$\phi_t(M) = \phi_t(M_1) \circ (e_{\xi_1} \circ p_{\xi_1}) \circ \phi_t(M_2)$$

on the basis of $(e_{\xi_1} \circ p_{\xi_1})$ having been proved to be PCF-definable by Proposition 3.4.5 on page 91. Now:

$$\begin{array}{rcl} [\![M : \xi(I)]\!](\psi_{Env}(\rho)) & = & ([\![M_1]\!](\psi_{Env}(\rho)))\ ([\![M_2]\!](\psi_{Env}(\rho))) \\ \text{(induction hypothesis)} & = & (p_{\xi_1\to\xi}[\![\phi_t(M_1)]\!](\rho))\ (p_{\xi_1}[\![\phi_t(M_2)]\!](\rho)) \\ \text{(definition of } p_{\xi_1\to\xi}) & = & (p_\xi\ ([\![\phi_t(M_1)]\!](\rho))\ e_{\xi_1})\ (p_{\xi_1}[\![\phi_t(M_2)]\!](\rho)) \\ & = & p_\xi([\![\phi_t(M_1) \circ (e_{\xi_1} \circ p_{\xi_1}) \circ (\phi_t(M_2))]\!](\rho)) \end{array}$$

• Case 6 : $M = (\lambda x : \xi_1(I) . M_1) : \xi_1(I) \to \xi_2(I)$: in this case we take

$$\phi_t(M) = \lambda x : \xi_1(nat \to nat) . \phi_t(M_1)$$

and in return we get:

$$\begin{array}{rcl} [\![\lambda x : \xi_1(I) . M_1]\!](\psi_{Env}(\rho)) & = & \lambda y : [\![\xi_1(I)]\!]. \\ & & \quad ([\![M_1 : \xi_2(I)]\!](\psi_{Env}(\rho))_{x\mapsto y}) \\ \text{(induction hypothesis)} & = & \lambda y : [\![\xi_1(I)]\!]. \\ & & \quad (p_{\xi_2}([\![\phi_t(M_1)]\!](\rho_{x\mapsto e_{\xi_1}(y)}))) \\ \text{(easy verification}^{16}) & = & p_{\xi_2}(\lambda y : [\![\xi_1(nat \to nat)]\!]. \\ & & \quad ([\![\phi_t(M_1)]\!](\rho_{x\mapsto y})))\ e_{\xi_1} \\ \text{(definition of } p_{\xi_1\to\xi_2}) & = & p_{\xi_1\to\xi_2}(\lambda y : [\![\xi_1(nat \to nat)]\!] . \\ & & \quad [\![\phi_t(M_1)]\!](\rho_{x\mapsto y})) \\ \text{(definition of } [\![\lambda y . \phi_t(M_1)]\!]) & = & p_{\xi_1\to\xi_2}([\![\lambda x : \xi_1(nat \to nat) . \\ & & \quad \phi_t(M_1)]\!](\rho)) \end{array}$$

□

3.4.3 Conservativity w. r. t. the Interval Domain Model

What we have achieved so far is not acceptable as the ultimate conservativity result, simply because we have considered the sequence model[17] of wRPCF. The original model of (weak) Real-PCF is the *interval domain model*[18] to which we

[16] It is only enough to apply both sides on any element $y \in [\![\xi_1(I)]\!]$.
[17] Subsection 3.4.1
[18] Subsection 2.5.1

need to link our result. This time we define a binary logical relation and then resort to the *fundamental lemma of logical relations*[19].

We first consider a function $to\mathcal{I} : \mathcal{S} \to \mathcal{I}$ such that for any $x \in \mathcal{S}$, $to\mathcal{I}(x) \in \mathcal{I}$ is the interval x is supposed to represent. Let $\mathcal{S}_f \subseteq \mathcal{S}$ be the subset of $\mathcal{S}$ consisting of all the finite sequences. We define an auxiliary function:

Definition 3.4.9 ($to\mathcal{I}_f$).

$$\begin{aligned}
&to\mathcal{I}_f : \mathcal{S}_f \to \mathcal{I}\\
&to\mathcal{I}_f\ [\,] = [0, 1]\\
&to\mathcal{I}_f\ (\mathsf{Cons}_{a_i} : x) = [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}_f(x))
\end{aligned}$$

Now for any $x \in \mathcal{S}$, let $x_{<n}$ denote the initial segment of x of length not more than n. In particular:

$$(x \in \mathcal{S}) \wedge (length(x) \leq n) \;\Rightarrow\; x_{<n} = x$$

This way we can define $to\mathcal{I} : \mathcal{S} \to \mathcal{I}$ by:

Definition 3.4.10 ($to\mathcal{I}$).

$$to\mathcal{I}(x) = \sqcup\{to\mathcal{I}_f(x_{<n}) \mid n \in \mathbb{N}\}$$

Now we are able to define the binary logical relation R between the sequence model and the interval domain model:

Definition 3.4.11 (the binary logical relation R). *We first define $R^o \subseteq [\![o]\!]_{\mathcal{S}} \times [\![o]\!]_{\mathcal{I}}$ — where $o \in \{nat, bool, I\}$ is a ground type — as follows:*

1. *If $o \in \{nat, bool\}$ then*
$$x\, R^o\, y \iff x = y$$
Note that $[\![nat]\!]_{\mathcal{S}} = [\![nat]\!]_{\mathcal{I}} = \mathbb{N}_\perp$ and $[\![bool]\!]_{\mathcal{S}} = [\![bool]\!]_{\mathcal{I}} = \mathbb{B}_\perp$.

2. *For any $x \in \mathcal{S}$ and $y \in \mathcal{I}$:*
$$x\, R^I\, y \iff to\mathcal{I}(x) = y$$

R is the logical relation built up over these ground cases.

[19] Lemma 2.4.5 on page 51.

The aim is to show that R is a wRPCF-logical relation. As usual, the tricky part is to demonstrate that the fix-point operators preserve the logical relation.

Proposition 3.4.12. *R^I forms an inclusive predicate*[20] *over* $[\![I]\!]_{\mathcal{S}} \times [\![I]\!]_{\mathcal{I}} = \mathcal{S} \times \mathcal{I}$.

Proof.

1. $\bot_{\mathcal{S}}\, R^I\, \bot_{\mathcal{I}}$ because:

$$to\mathcal{I}(\bot_{\mathcal{S}}) = to\mathcal{I}([\,]) = [0,1] = \bot_{\mathcal{I}}$$

2. Suppose $\{(x_1^i, x_2^i) \mid i \in \mathbb{N}\} \subseteq R^I$ is an ascending chain - which in particular implies $\forall i \in \mathbb{N}\colon to\mathcal{I}(x_1^i) = x_2^i$. Then by continuity of $to\mathcal{I} : \mathcal{S} \to \mathcal{I}$ we have:

$$to\mathcal{I}(\sqcup\{x_1^i \mid i \in \mathbb{N}\}) = \sqcup\{to\mathcal{I}(x_1^i) \mid i \in \mathbb{N}\} = \sqcup\{x_2^i \mid i \in \mathbb{N}\}$$

therefore

$$(\sqcup\{x_1^i \mid i \in \mathbb{N}\})\, R^I\, (\sqcup\{x_2^i \mid i \in \mathbb{N}\})$$

□

Lemma 3.4.13. *For any type σ:*

$$[\![Y_\sigma]\!]_{\mathcal{S}}\, R^{(\sigma\to\sigma)\to\sigma}\, [\![Y_\sigma]\!]_{\mathcal{I}}$$

Proof. According to the definition of $[\![Y_\sigma]\!]$, it suffices to verify the following for every type σ:

1. $\bot_{[\![\sigma]\!]_{\mathcal{S}}}\, R^\sigma\, \bot_{[\![\sigma]\!]_{\mathcal{I}}}$
2. $(x_1\, R^\sigma\, x_2) \wedge (F_1 R^{\sigma\to\sigma} F_2) \Rightarrow F_1(x_1)\, R^\sigma\, F_2(x_2)$
3. If $\{(x_1^i, x_2^i) \mid i \in \mathbb{N}\} \subseteq R^\sigma$ is an ascending chain, then

$$(\sqcup\{x_1^i \mid i \in \mathbb{N}\})\, R^\sigma\, (\sqcup\{x_2^i \mid i \in \mathbb{N}\})$$

First we note that (2) follows immediately from the definition of the logical relations. (1) and (3) can also be proved by easy induction over σ, i. e.

- $\sigma \in \{nat, bool\}$: As in this case $R^\sigma = id_{[\![\sigma]\!]}$, (1) and (3) become trivial.
- $\sigma = I$: Proposition 3.4.12 above.

[20] Definition 3.2.5 on page 67.

- $\sigma = \sigma_1 \to \sigma_2$: Routine calculations reduce the argument to σ_2 for which (1) and (3) hold by induction hypothesis. □

Theorem 3.4.14. *R is a $w\mathcal{RC}_{\mathrm{A}}$-logical relation.*

Proof. We need to show that for any wRPCF constant $c\colon \sigma$

$$[\![c]\!]_{\mathcal{S}}\, R^{\sigma}\, [\![c]\!]_{\mathcal{I}} \tag{3.14}$$

- Case 1 : c is a PCF constant : then Equation (3.14) above follows trivially.
- Case 2 : $c = cons_{a_i}$ for some $i \in \mathbb{N}$: in this case we need to show that:

$$\forall x \in \mathcal{S}\colon [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}(x)) = to\mathcal{I}([\![cons_{a_i}]\!]_{\mathcal{S}}(x)) \tag{3.15}$$

We first prove (3.15) above for elements of $\mathcal{S}_f$ — i. e. the finite sequences — by structural induction. Notice that on finite sequences we have:

$$\forall x \in \mathcal{S}_f\colon to\mathcal{I}(x) = to\mathcal{I}_f(x) \tag{3.16}$$

(a) Case $x = [\,]$

$$\begin{array}{rcl}
[\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}(x)) & = & [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}([\,])) \\
\text{(Equation (3.16) above)} & = & [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}_f([\,])) \\
\text{(definition of } to\mathcal{I}_f\text{)} & = & to\mathcal{I}_f(\mathsf{Cons}_{a_i} : [\,]) \\
\text{(Equation (3.16) above)} & = & to\mathcal{I}(\mathsf{Cons}_{a_i} : [\,]) \\
\text{(definition of } [\![cons_{a_i}]\!]_{\mathcal{S}}\text{)} & = & to\mathcal{I}([\![cons_{a_i}]\!]_{\mathcal{S}}([\,])) \\
 & = & to\mathcal{I}([\![cons_{a_i}]\!]_{\mathcal{S}}(x))
\end{array}$$

(b) Case $x = \mathsf{Cons}_{a_j} : y$

$$\begin{array}{rcl}
[\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}(X)) & = & [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}(\mathsf{Cons}_{a_j} : y)) \\
\text{(Equation (3.16) above)} & = & [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}_f(\mathsf{Cons}_{a_j} : y)) \\
\text{(definition of } to\mathcal{I}_f\text{)} & = & to\mathcal{I}_f(\mathsf{Cons}_{a_i} : \mathsf{Cons}_{a_j} : y) \\
\text{(definition of } [\![cons_{a_i}]\!]_{\mathcal{S}}\text{)} & = & to\mathcal{I}_f([\![cons_{a_i}]\!]_{\mathcal{S}}(\mathsf{Cons}_{a_j} : y)) \\
 & = & to\mathcal{I}([\![cons_{a_i}]\!]_{\mathcal{S}}(x))
\end{array}$$

Now for any arbitrary $x \in \mathcal{S}$, by Definition 3.4.10 on page 98 we can write:

$$\begin{array}{rcl} [\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}(x)) & = & [\![cons_{a_i}]\!]_{\mathcal{I}}(\sqcup\{to\mathcal{I}_f(x_{<n}) \mid n \in \mathbb{N}\}) \\ ([\![cons_{a_i}]\!]_{\mathcal{I}} \text{ is continuous}) & = & \sqcup\{[\![cons_{a_i}]\!]_{\mathcal{I}}(to\mathcal{I}_f(x_{<n}) \mid n \in \mathbb{N}\}) \\ (\text{definition of } to\mathcal{I}_f) & = & \sqcup\{to\mathcal{I}_f(\mathsf{Cons}_{a_i} : (x_{<n})) \mid n \in \mathbb{N}\}) \\ (\text{definition of } [\![cons_{a_i}]\!]_{\mathcal{S}}) & = & \sqcup\{to\mathcal{I}_f([\![cons_{a_i}]\!]_{\mathcal{S}}(x_{<n})) \mid n \in \mathbb{N}\}) \\ (\text{definition of } to\mathcal{I}) & = & to\mathcal{I}([\![cons_{a_i}]\!]_{\mathcal{S}}(x)) \end{array}$$

- Case 3 : $c = tail_{a_i}$ or $head_r$: similar to Case 2 on the preceding page.
- Case 4 : $c = if_I$: trivial.
- Case 5 : $c = Y_\sigma$ for some σ : Lemma 3.4.13 on page 99. □

We have now finished the tedious part. Here are a couple of easier observations:

Proposition 3.4.15. *If σ is a PCF-type, then:*

$$\forall f, g \in [\![\sigma]\!] : f\, R^{\sigma}\, g \iff f = g$$

Proof. Induction on σ:

- Case 1 : $\sigma \in \{nat, bool\}$: Trivial by definition of R.
- Case 2 : $\sigma = \sigma_1 \to \sigma_2$: ($\Rightarrow$) Suppose $f\, R^{\sigma_1\to\sigma_2}\, g$, hence for all $x \in [\![\sigma_1]\!]$:

$$\begin{array}{rcl} (\text{induction hypothesis on } \sigma_1) & \Rightarrow & x\, R^{\sigma_1}\, x \\ (\text{as } f\, R^{\sigma_1\to\sigma_2}\, g) & \Rightarrow & f(x)\, R^{\sigma_2}\, g(x) \\ (\text{induction hypothesis on } \sigma_2) & \Rightarrow & f(x) = g(x) \end{array}$$

i. e.

$$\forall x \in [\![\sigma_1]\!] : f(x) = g(x)$$

which implies $f = g$ as the model is extensional.

($\Leftarrow$) Let $f = g$, hence in particular as the model is extensional:

$$\forall x, y \in [\![\sigma_1]\!] : x = y \Rightarrow f(x) = g(y) \in [\![\sigma_2]\!] \tag{3.17}$$

Thus, for any $x, y \in [\![\sigma_1]\!]$ such that $x\, R^{\sigma_1}\, y$:

$$\begin{array}{rcl} \text{(induction hypothesis on } \sigma_1) & \Rightarrow & x = y \\ \text{(Equation (3.17) on the page before)} & \Rightarrow & f(x) = g(y) \\ \text{(induction hypothesis on } \sigma_2) & \Rightarrow & f(x)\, R^{\sigma_2}\, g(y) \end{array}$$

i. e.

$$\forall x, y \in [\![\sigma_1]\!] :\ x\, R^{\sigma_1}\, y \Rightarrow f(x)\, R^{\sigma_2}\, g(y)$$

therefore $f\, R^{\sigma_1 \to \sigma_2}\, g$ □

Lemma 3.4.16. *Let $M : \sigma$ be a closed* wRPCF*-term of a **PCF-type** σ. Then:*

$$[\![M]\!]_{\mathcal{S}} = [\![M]\!]_{\mathcal{I}}$$

Proof. By Theorem 3.4.14 we have:

$$[\![M]\!]_{\mathcal{S}}\, R^{\sigma}\, [\![M]\!]_{\mathcal{I}}$$

Now Proposition 3.4.15 is applicable. □

Finally, combining Lemma 3.4.8 on page 95 and Lemma 3.4.16 we get the desired conservativity result:

Theorem 3.4.17 (conservativity). *Let $M : \sigma$ be a closed* wRPCF*-term of the **PCF-type** σ. Then there exists a PCF-term $N : \sigma$ such that:*

$$[\![M]\!]_{\mathcal{I}} = [\![N]\!]_{\mathcal{I}}$$

Proof. We take N to be $\phi_t(M)$ as in Lemma 3.4.8. As M is closed and σ is a PCF-type, Equation (3.13) on page 95 reduces to:

$$[\![M]\!]_{\mathcal{S}} = [\![N]\!]_{\mathcal{S}}$$

On the other hand:

1. By Lemma 3.4.16: $[\![M]\!]_{\mathcal{S}} = [\![M]\!]_{\mathcal{I}}$
2. N is a PCF-term, so: $[\![N]\!] = [\![N]\!]_{\mathcal{I}} = [\![N]\!]_{\mathcal{S}}$

therefore:

$$[\![M]\!]_{\mathcal{I}} = [\![N]\!]_{\mathcal{I}}$$ □

Table 3.6 Denotational semantics of RPCF w. r. t. the sequence model

- $\forall i, j \in \mathbb{N}$:

$$\llbracket cons_{a_i} \rrbracket_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}$$
$$\llbracket cons_{a_i} \rrbracket_{\mathcal{S}}\,[\,] = [\mathsf{Cons}_{a_i}]$$
$$\llbracket cons_{a_i} \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_j} : x) = \mathsf{Cons}_{a_i} : \mathsf{Cons}_{a_j} : x$$

- For $tail_{a_i}$'s, we follow the one-step reduction rules for *tail* as in Table 2.9 on page 60. Thus, $\forall i, j, k \in \mathbb{N}$:

$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}$$
$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}}\,[\,] = [\,]$$
$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_j} : x) = \mathsf{Cons}_{[0,\frac{1}{2}]} : \mathsf{Cons}_{[0,\frac{1}{2}]} : \ldots \quad (\text{if } a_j \leq a_i)$$
$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_j} : x) = \mathsf{Cons}_{[\frac{1}{2},1]} : \mathsf{Cons}_{[\frac{1}{2},1]} : \ldots \quad (\text{if } a_i \leq a_j)$$
$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_j} : x) = \mathsf{Cons}_{a_j \backslash a_i} : x \quad (\text{if } a_i \sqsubseteq a_j, a_i \neq a_j)$$
$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_j} : x) = \mathsf{Cons}_{(a_i \sqcup a_j)\backslash a_i} : (\llbracket tail_{(a_i \sqcup a_j)\backslash a_j} \rrbracket_{\mathcal{S}}(x))$$
$$(\text{if } a_i \uparrow a_j,\ a_i \not\sqsubseteq a_j,\ a_j \not\sqsubseteq a_i,\ a_i \not\leq a_j,\ a_j \not\leq a_i)$$
$$\llbracket tail_{a_i} \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_j} : \mathsf{Cons}_{a_k} : x) = \llbracket tail_{a_i} \rrbracket_{\mathcal{S}}(\mathsf{Cons}_{a_j a_k} : x)$$

- $\forall r \in \mathbb{Q} \cap (0,1), i, j \in \mathbb{N}$:

$$\llbracket \mathsf{head}_r \rrbracket_{\mathcal{S}} : \mathcal{S} \to \mathbb{B}_\perp$$
$$\llbracket \mathsf{head}_r \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_i} : x) = \begin{cases} true & \text{if } a_i < r \\ false & \text{if } a_i > r \end{cases}$$
$$\llbracket \mathsf{head}_r \rrbracket_{\mathcal{S}}\,(\mathsf{Cons}_{a_i} : \mathsf{Cons}_{a_j} : x) = \llbracket \mathsf{head}_r \rrbracket_{\mathcal{S}}(\mathsf{Cons}_{a_i a_j} : x)$$

- $\forall p \in \mathbb{B}_\perp, i, j \in \mathbb{N}$:

$$\llbracket pif_I \rrbracket_{\mathcal{S}}(p)\,(\mathsf{Cons}_{a_i} : x)(\mathsf{Cons}_{a_j} : y) = \begin{cases} (\mathsf{Cons}_{a_i} : x) & \text{if } p = true \\ (\mathsf{Cons}_{a_j} : y) & \text{if } p = false \\ A & \text{if } p = \perp \end{cases}$$

where

$$A = \mathsf{Cons}_{a_i \sqcap a_j} : (\llbracket pif_I \rrbracket_{\mathcal{S}}(p)(\mathsf{Cons}_{a_i \backslash (a_i \sqcap a_j)} : x)\,(\mathsf{Cons}_{a_j \backslash (a_i \sqcap a_j)} : y))$$

Chapter 4

The Language SHRAD

4.1 Comparison

Although exact real number computation has not yet found its way to our pocket calculators, there is a fairly rich literature available on the subject containing various approaches. Here we are about to make an attempt to find a middle-ground between *efficiency* and *data-abstraction*, thereby presenting a new framework in which major problems with previous works are avoided. This in turn necessitates an overview of the literature, with the intention of highlighting how our framework differs from all the previous ones. Thus we hand-pick a number of previous frameworks and briefly go through their descriptions. Again we emphasize that the literature is too vast to let our modest overview get anywhere near being comprehensive.

Böhm et al. [1986] present the implementations of the four basic arithmetic operators — addition, subtraction, multiplication and division — based on the representation of real numbers via redundant sequences of *b*-adic numbers. Ménissier-Morain [1996] takes this matter further and extends the implementations to more complicated functions. Böhm and Cartwright's work is regarded as seminal though in retrospect the lack of any kind of semantics makes it extremely hard to verify the correctness of any of the algorithms presented in [Ménissier-Morain, 1996].

Di Gianantonio [1993] was the first to introduce a framework for exact real number computation which included a representation-independent denotational semantics versus a representation-dependent operational one [Di Gianantonio, 1993, 1999]. Yet the framework suffers from the following deficiencies:

1. Existence of parallel operators.
2. Multi-valuedness is allowed in the language.
3. The domain which models the data type for real numbers — as it is — cannot have the real line as its space of maximal elements.

Real-PCF was put forward by Escardó [1997] who used interval domain (Definition 2.5.3 on page 54) to model the data type of real numbers. We have already discussed this approach in more detail in Section 2.5, yet it is useful to mention how Real-PCF solved the problem with Di Gianantonio's approach. The interval domain is a continuous domain which contains the real line as its subspace of maximal elements. *Extensionality* is kept over both partial and total real numbers. In fact this has made Real-PCF unique regarding data abstraction. Universality at first-order is already given by the language while adding a constant for the *existential quantifier* (Definition 2.3.13 on page 47) results in universality at all types [Escardó, 1996]. Yet the presence of parallel operators makes it extremely impractical to use Real-PCF even for computations involving basic operations over real numbers.

LFT Approach was introduced by Edalat and Potts [2000] and incorporated the (familiar) idea of representing real numbers as shrinking sequences of rational intervals. Later Edalat and Heckmann [2002] amalgamated signed-digit binary representation of real numbers and the LFT implementation of operations over them into a unified framework. Over non-negative real numbers this rather elegant approach maintains the single-valuedness property [Potts et al., 1997], though once extended over to the whole real line the presence of *redundant-if* allows multi-valuedness [Edalat et al., 1998]. But once again the major problem lies in the inefficiency of the framework, this time regarding *space*. This matter has been studied from certain angles by Heckmann [1998, 2000b] who has also carried out further investigation into the LFT approach [Heckmann, 2000a, 2001, 2002]. Looking at the issue from another angle, one can take the definition of the *tangent* function as presented in [Edalat et al., 1998, Section 3.3]. It is obvious that the input stream has to be kept in its entirety throughout the computation process as even the first element is not consumed at any stage. Adding to the previous analyses — bearing in mind the limits we face in practice regarding space — Konečný [2000, 2002] proved that any function computable by a finite transducer using LFT-representation is essentially affine.

Real-RAM: Müller [2000] implemented basic arithmetic operations together with the limit operation over real numbers inside C++. This leads to a relatively fast implementation which is inhabited side-by-side with *most* other imperative features of the language C++, but *not all*. This framework is neither single-valued nor meant to be so. In fact it closely follows the ideas of Brattka [1996] where corresponding to μ-recursive *functions* over natural numbers, a class of recursive *relations* over real numbers have been introduced that can be generated from a class of basic recursive relations — e. g. addition, multiplication, limit, etc. However, no formal relation between the two frameworks has been presented yet.

Marcial-Romero and Escardó [2004] present a framework the syntax of which is essentially the same as Real-PCF, with ordinary sequential *if* and *rtest*[1] replacing *pif* and *head*, respectively.[2] This way the inefficiency of the parallel-if present in Real-PCF is avoided at the cost of losing single-valuedness over *both* partial and total real numbers. Hence the original interval domain model is of no help anymore and a special category of *power domains* is employed as model.

Our Approach We present the implementation of an **abstract data type** for real numbers in a **sequential** setting, which is at least **expressive enough** to give us all computable[3] first-order functions on real numbers. Moreover, we **syntactically prevent multi-valuedness** over *total* real numbers. These properties cannot be found *all at the same time* in any of the previous works available in the literature.

Of course, Escardó et al. [1998, 2004] demonstrated that sequentiality and extensionality are impossible to co-exist in a framework based on the interval domain or any of its variants. Thus we choose to relax extensionality over *partial* real numbers in order to maintain sequentiality. But preventing multi-valuedness over total real numbers syntactically is an achievement which we put more emphasis on. These terms will be explained in detail in due course.

[1] redundant-test.
[2] cf. *redundant-if* construct studied by Edalat et al. [1998].
[3] e. g. in the TTE sense [Weihrauch, 2000].

4.2 The Syntax of Shrad

Definition 4.2.1 (Types of Shrad). *The set $\mathbb{T}_{\mathcal{SH}}$ is defined by the following grammar:*

$$\sigma ::= nat \mid bool \mid r \mid \sigma \to \sigma$$

Definition 4.2.2 (Shrad). *The language* Shrad[4] *is the extension of PCF with a ground type r together with the set $\mathcal{P}_{\mathcal{SH}}$ of the following constants:*

1. $L\colon r \to r, \qquad M\colon r \to r, \qquad R\colon r \to r$
2. $L^{-1}\colon r \to r, \qquad M^{-1}\colon r \to r, \qquad R^{-1}\colon r \to r$
3. $qtoR\colon nat \to nat \to nat \to r$
4. $avg\colon r \to r \to r$
5. $mult\colon r \to r \to r$
6. $abs\colon r \to r$
7. $isNeg\colon r \to bool$
8. $limit\colon (nat \to r) \to r$
9. $Y_\sigma : (\sigma \to \sigma) \to \sigma$ *(for each* Shrad*-type* $\sigma \in \mathbb{T}_{\mathcal{SH}}$*)*

4.3 Denotational Semantics

The PCF fragment of Shrad is interpreted as in subsection 2.3.2. As regards the type r we need to clarify what our objectives are. The constants L, M and R are going to be used as *digits*, and following tradition we aim for reducing programs of type r to sequences of digits. Hence we reserve symbol $\mathcal{D}$ and define:

[4] It might seem helpful right here to mention that:

- **S** stands for ***S**equential*
- **H** stands for ***H**igher order*, due to the presence of the second-order constant *limit*.
- **RAD** stands for ***R**eal numbers as an **A**bstract **D**ata type*

Definition 4.3.1 (The set $\mathcal{D}$ of digits).

$$\mathcal{D} := \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$$

Notation 4.3.2 (Important). *In Definition 4.3.1 above we have already made a not-so-bold typographical distinction between the syntactic entities L, M, R — written in* ***italic*** *shape — and their semantic counterparts* $\mathsf{L}, \mathsf{M}, \mathsf{R}$ *— written in* **upshape sans serif** font. We maintain this distinction throughout the thesis. Thus, the denotation of the constant *avg* will be written as avg, and so on.

Definition 4.3.3 (X^*, X^ω, X^∞). *Let X be an arbitrary set:*

1. *by X^* we mean the set of all* finite *sequences over X.*
2. *by X^ω we mean the set of all* infinite *sequences over X.*
3. *by X^∞ we mean the union of the previous two, i. e.*

$$X^\infty := X^* \cup X^\omega$$

Definition 4.3.4 ($\mathcal{D}^\infty$). *$(\mathcal{D}^\infty, \sqsubseteq_{\mathcal{D}^\infty})$ is the algebraic domain of finite and infinite sequences over the alphabet $\mathcal{D}$, i. e.*

$$\mathcal{D}^\infty := \mathcal{D}^* \cup \mathcal{D}^\omega$$

partially ordered by:

$$\forall x, y \in \mathcal{D}^\infty : \; x \sqsubseteq_{\mathcal{D}^\infty} y \iff x \text{ is an initial segment of } y$$

which we simply write as $\mathcal{D}^\infty$.

The flow of material already reveals that we have the *signed-digit binary* representation in the back of our minds. In fact, to bring it to the surface, let us define an interval domain $\mathcal{J}$ which is in all aspects isomorphic to the unit interval domain $\mathcal{I}$ except that it is built up over $[-1, 1]$:

Definition 4.3.5 (The interval domain $\mathcal{J}$). *The interval domain $\mathcal{J}$ is built up over* $[-1, 1]$ *in exactly the same manner as $\mathcal{I}$ was built up over* $[0, 1]$ *in Definition 2.5.3 on page 54.*

Note 4.3.6. *In particular, we can have functions:*

1. $\mathsf{Cons}_a, \mathsf{Tail}_a \colon \mathcal{J} \to \mathcal{J}$

2. $\mathsf{head}_r \ : \ \mathcal{J} \ \to \ \mathbb{B}_\perp$

similar to those over $\mathcal{I}$, with the difference that a ranges over the set:

- $\{[p, q] \mid -1 \leq p \leq q \leq 1\}$*, as a subscript to* Cons.
- $\{[p, q] \mid -1 \leq p < q \leq 1\}$*, as a subscript to* Tail.

and r over that of $(-1, 1)$.

Note 4.3.7. *Consider the embedding $e \colon [-1, 1] \hookrightarrow \mathcal{J}$ defined by:*

$$\forall x \in [-1, 1] \colon e(x) = [x, x]$$

We freely talk about "the maximal element $x \in \mathcal{J}$" instead of $[x, x]$, unless the difference happens to be too crucial to ignore.

It is straightforward to interpret elements of $\mathcal{D}^\infty$ inside $\mathcal{J}$ using the following recursive definition:

Definition 4.3.8 ($to_{\mathcal{J}} \colon \mathcal{D}^\infty \to \mathcal{J}$).

$$\begin{array}{lcl} to_{\mathcal{J}}\ ([\,]) & = & [-1, 1] \\ to_{\mathcal{J}}\ (\mathsf{L} \colon x) & = & \mathsf{Cons}_{[-1,0]}\ (to_{\mathcal{J}}\ x) \\ to_{\mathcal{J}}\ (\mathsf{M} \colon x) & = & \mathsf{Cons}_{[-1/2,1/2]}\ (to_{\mathcal{J}}\ x) \\ to_{\mathcal{J}}\ (\mathsf{R} \colon x) & = & \mathsf{Cons}_{[0,1]}\ (to_{\mathcal{J}}\ x) \end{array}$$

An element $d \in \mathcal{D}^\infty$ is said to represent *— be a* representation *of — an interval $a \in \mathcal{J}$ iff $to_{\mathcal{J}}(d) = a$. In particular, following the Note 4.3.7 above, we say that the infinite sequence $d_x \in \mathcal{D}^\omega$ is a* representation *of the real number $x \in [-1, 1]$, iff:*

$$to_{\mathcal{J}}(d_x) = [x, x] = e(x)$$

Plotkin's terminology (see [Plotkin, 1977]) is adopted as follows:

Definition 4.3.9. *The collection $\mathbb{D}$ of domains for* Shrad *is the set*

$$\mathbb{D} := \{\mathbb{D}_\sigma \mid \sigma \text{ a } \textsc{Shrad}\text{-}type\ \}$$

built recursively by:

Table 4.1 Where to find the interpretation of SHRAD constants

Constant	Implemented as	Definition	Page
L, M, R	$\mathsf{Cons}_{\mathsf{L,M,R}}$	4.4.4	114
L^{-1}, M^{-1}, R^{-1}	$\mathsf{L}^{-1}, \mathsf{M}^{-1}, \mathsf{R}^{-1}$	4.4.16	119
qtoR	qtoR	4.4.22	122
avg	avg	4.4.28	124
mult	mult	4.4.37	128
abs	abs	4.4.45	131
isNeg	isNeg	4.4.51	132
limit	limit	4.4.67	143

- $\mathbb{D}_{bool} = \mathbb{B}_\perp, \qquad \mathbb{D}_{nat} = \mathbb{N}_\perp, \qquad \mathbb{D}_r = \mathcal{D}^\infty$
- $\mathbb{D}_{\sigma\to\tau} = [\mathbb{D}_\sigma \to \mathbb{D}_\tau]$, *i. e. the domain-theoretic function space.*

Definition 4.3.10 (Interpreting types). *The interpretation of any type σ — written as $[\![\sigma]\!]$ — is $\mathbb{D}_\sigma$, i. e.*

$$[\![\sigma]\!] := \mathbb{D}_\sigma$$

Definition 4.3.11 (Interpreting constants). *We interpret the elements of $\mathcal{P}_{SH}$ via the function:*

$$\mathcal{A}_{SH} : \mathcal{P}_{SH} \to \cup\{\mathbb{D}_\sigma \mid \sigma \text{ a SHRAD-type}\}$$

which is type-respecting, i. e.

$$\forall c \in \mathcal{P}_{SH} : \ c : \sigma \ \Rightarrow \mathcal{A}_{SH}(c) \in \mathbb{D}_\sigma$$

The fix-point constants are dealt with as usual:

$$\forall f \in \mathbb{D}_{\sigma\to\sigma} : \quad \mathcal{A}_{SH}(Y_\sigma)(f) = \sqcup \{f^n(\perp) \mid n \geq 0\}$$

For other constants $c \in \mathcal{P}_{SH}$, Table 4.1 shows where the reader can find the exact definition of $\mathcal{A}_{SH}(c)$ together with a discussion of its correctness.

Definition 4.3.12 (Denotational Semantics). *Relative to each environment ρ, the denotational semantics of* SHRAD*-terms is defined recursively as follows:*

1. $[\![x]\!]_\rho := \rho(x)$ *where x is a variable.*

2. $[\![c]\!]_\rho := \mathcal{A}(c)$ *where c is a constant of PCF and $\mathcal{A}$ is the function defined in Table 2.3 on page 41.*

3. $[\![c]\!]_\rho := \mathcal{A}_{\mathcal{SH}}(c)$ *where $c \in \mathcal{P}_{SH}$ is a proper* SHRAD*-constant and $\mathcal{A}_{\mathcal{SH}}$ is the function defined in Definition 4.3.11 on the page before.*

4. $[\![MN]\!]_\rho := [\![M]\!]_\rho [\![N]\!]_\rho$

5. $\forall a \in \mathbb{D}_\sigma \colon [\![\lambda x \colon \sigma \,.\, M]\!]_\rho(a) := [\![M]\!]_{\rho_{x \mapsto a}}$

4.4 Interpreting constants

We already know how to interpret a program $P\colon r$ as a (partial) real number, all we need to do is to consider $to_{\mathcal{J}}\ [\![P]\!]$. Now let us move up to first-order and consider a term $P\colon r^m \to r$, $(m \geq 1)$. Perhaps the natural way of observing this term's behaviour is to define some

$$to_{\mathcal{J}^m \to \mathcal{J}} \colon [(\mathcal{D}^\infty)^m \to \mathcal{D}^\infty] \to (\mathcal{J}^m \to \mathcal{J})$$

where, given a function $f \in [(\mathcal{D}^\infty)^m \to \mathcal{D}^\infty]$ and a vector $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathcal{J}^m$:

$$to_{\mathcal{J}^m \to \mathcal{J}}\ f\ \boldsymbol{a} = \begin{cases} to_{\mathcal{J}}\ (f\ d_1 \ldots d_m) & \text{if } \forall i \leq m \colon to_{\mathcal{J}}\ d_i = a_i \\ \text{undefined} & \text{if no such } \boldsymbol{d} \text{ exists} \end{cases} \tag{4.1}$$

For a SHRAD-definable f, the question is whether $to_{\mathcal{J}^m \to \mathcal{J}}\ f$ is a well-defined *function* or not? In other words, for such a function, does the specific choice of $\boldsymbol{d}$ in (4.1) above really matter? The answer to this last question is — unfortunately — "yes", and a counter-example is readily available.

Take the constant $avg\colon r \to r \to r$ and consider its denotation $\mathsf{avg}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$ as in Definition 4.4.28 on page 124. Also consider the intervals:

$$I_1 = [-1/2, 0] \qquad I_2 = [-1/2, 1/2]$$

It is easy to verify that:

$$\begin{aligned} to_{\mathcal{J}}\ (\mathsf{L}\colon \mathsf{R}\colon [\,]) = to_{\mathcal{J}}\ (\mathsf{M}\colon \mathsf{L}\colon [\,]) &= I_1 \\ to_{\mathcal{J}}\ (\mathsf{M}\colon [\,]) &= I_2 \end{aligned}$$

Yet we have:

$$\begin{cases} \mathsf{avg}\ (\mathsf{L}\colon \mathsf{R}\colon [\,])\ (\mathsf{M}\colon [\,]) = [\,] \\ \mathsf{avg}\ (\mathsf{M}\colon \mathsf{L}\colon [\,])\ (\mathsf{M}\colon [\,]) = \mathsf{M}\colon [\,] \end{cases}$$

which leads to:

$$\begin{cases} to_{\mathcal{J}}(\mathsf{avg}\ (\mathsf{L}\colon \mathsf{R}\colon [\,])\ (\mathsf{M}\colon [\,])) = [-1, 1] \\ to_{\mathcal{J}}(\mathsf{avg}\ (\mathsf{M}\colon \mathsf{L}\colon [\,])\ (\mathsf{M}\colon [\,])) = [-1/2, 1/2] \end{cases} \tag{4.2}$$

Thus, in general $to_{\mathcal{J}^m \to \mathcal{J}}[\![P]\!]$ is a *relation* rather than a function. In the special case of the above counter-example, we say that $to_{\mathcal{J}^2 \to \mathcal{J}}\ \mathsf{avg}\colon \mathcal{J}^2 \to \mathcal{J}$ — or by an abuse of language $\mathsf{avg}\colon (\mathcal{D}^\infty)^2 \to \mathcal{D}^\infty$ for that matter — is *non-extensional* at $([-1/2, 0], [-1/2, 1/2])$.

But how bad is the situation? Looking at it from another angle, we should remember that it is the behaviour of the function over *total* real numbers that ultimately matters. Therefore we need to know whether for any term $P\colon r^m \to r$ it is possible that $to_{\mathcal{J}^m \to \mathcal{J}}\ [\![P]\!]$ is non-extensional over any maximal $\boldsymbol{a} \in \mathcal{J}^m$.

For the rest of this section we go through the denotation of each individual constant and demonstrate that for each such constant, say $c \in \mathcal{P}_{SH}$, $[\![c]\!]$ preserves the partial equivalence relation of Definition 4.4.1 below, *except for limit*. Then in subsection 4.6.1 we weaken the extensionality criteria by defining a binary logical relation $\mathcal{X}$ which is indeed preserved by any term of the language.[5]

Definition 4.4.1 ($\simeq_{\mathcal{D}^\infty} \subseteq \mathcal{D}^\infty \times \mathcal{D}^\infty$). *$x, y \in \mathcal{D}^\infty$ are equivalent if and only if they represent the same* total *real number, i. e.*

$$\boxed{x \simeq_{\mathcal{D}^\infty} y \iff (length(x) \notin \mathbb{N}) \wedge (to_{\mathcal{J}}\ x = to_{\mathcal{J}}\ y)}$$

At the same time, we embark on proving the correctness of each interpretation, for which we present the following definition:

Notation 4.4.2 ($\mathcal{J}_{max}$). *By $\mathcal{J}_{max}$ we mean the set $[-1, 1]$, the image of which under the embedding e of Note 4.3.7 on page 110 is the set of maximal elements of $\mathcal{J}$.*

Definition 4.4.3 ($to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}}$). *For each $m \geq 1$, let*

$$to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}} \colon [(\mathcal{D}^\infty)^m \to \mathcal{D}^\infty] \rightharpoonup (\mathcal{J}^m_{max} \rightharpoonup \mathcal{J}_{max})$$

be the partial *function where given any $f \in [(\mathcal{D}^\infty)^m \to \mathcal{D}^\infty]$, the function*

$$to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}}(f)$$

[5] See Conjecture 6.2.2 on page 175.

1. *is* undefined *if for some* $\boldsymbol{a} \in \mathcal{J}_{max}^m$*, there are two representations* $\boldsymbol{d_1}, \boldsymbol{d_2} \in (\mathcal{D}^\infty)^m$ *such that:*

$$to_{\mathcal{J}}(f(\boldsymbol{d_1})) \cap to_{\mathcal{J}}(f(\boldsymbol{d_2})) = \emptyset$$

2. *otherwise is the (partial) function* $\hat{f} \colon \mathcal{J}_{max}^m \rightharpoonup \mathcal{J}_{max}$ *for which:*

   (a) $dom(\hat{f})$ *is the set of all* $\boldsymbol{x} \in \mathcal{J}_{max}^m$ *such that for any two* $\boldsymbol{d_1}, \boldsymbol{d_2} \in (\mathcal{D}^\infty)^m$ *representing* $\boldsymbol{x}$*:*

$$\big(to_{\mathcal{J}}(f(\boldsymbol{d_1})) \in \mathcal{J}_{max}\big) \wedge \big(to_{\mathcal{J}}(f(\boldsymbol{d_1})) = to_{\mathcal{J}}(f(\boldsymbol{d_2}))\big)$$

   (b) $\forall \boldsymbol{a} \in dom(\hat{f}) : \hat{f}(\boldsymbol{a}) = to_{\mathcal{J}}(f(\boldsymbol{d}))$ *where:*

$$\boldsymbol{d} = (d_1, \ldots, d_m) \in (\mathcal{D}^\infty)^m$$

   is some representation of $\boldsymbol{a}$*.*

For instance, in Corollary 4.4.33 on page 127 we show that:

$$\forall x, y \in [-1, 1] \colon (to_{\mathcal{J}_{max}^2 \to \mathcal{J}_{max}}(\mathsf{avg}))\, x\, y = \frac{x + y}{2}$$

After discussing each individual constant, we will have a further task to do and that is demonstrating that the extensionality property is preserved throughout the construction of more complicated terms out of the basic ones. Of course one has to bear in mind that with the introduction of *isNeg* and *limit*, we will be bound to get partial functions over real numbers. This is yet another motivation for a reformulation of extensionality by the logical relation $\mathcal{X}$ as we do in section 4.6. Thereafter, we are assured that in fact for *any* SHRAD-definable

$$f \colon (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$$

one gets a proper (partial) function

$$\hat{f} \colon [-1, 1]^m \rightharpoonup [-1, 1]$$

4.4.1 $\boxed{\mathsf{Cons_L}, \mathsf{Cons_M}, \mathsf{Cons_R} \colon \mathcal{D}^\infty \to \mathcal{D}^\infty}$

We begin by defining:

Definition 4.4.4 ($\mathsf{Cons_L}, \mathsf{Cons_M}, \mathsf{Cons_R}$).

$$\begin{cases} \mathsf{Cons_L} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{Cons_L}(x) = \mathsf{L} : x \end{cases} \quad \begin{cases} \mathsf{Cons_M} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{Cons_M}(x) = \mathsf{M} : x \end{cases} \quad \begin{cases} \mathsf{Cons_R} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{Cons_R}(x) = \mathsf{R} : x \end{cases}$$

Next we set:

$$\mathcal{A}_{\mathcal{SH}}(L) \coloneqq \mathsf{Cons_L} \qquad \mathcal{A}_{\mathcal{SH}}(M) \coloneqq \mathsf{Cons_M} \qquad \mathcal{A}_{\mathcal{SH}}(R) \coloneqq \mathsf{Cons_R}$$

For each $d \in \mathcal{D}^\infty$, let $[a_d, b_d] = to_{\mathcal{J}}\, d$. We have:

$$\begin{array}{rcl} to_{\mathcal{J}}(\mathsf{Cons_L}\ d) & = & to_{\mathcal{J}}(\mathsf{L} : d) \\ (\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Cons}_{[-1,0]}(to_{\mathcal{J}}\ d) \\ & = & \mathsf{Cons}_{[-1,0]}\ [a_d, b_d] \\ (\text{definition of } \mathsf{Cons}_{[-1,0]} : \mathcal{J} \to \mathcal{J}) & = & [(a_d - 1)/2\ , (b_d - 1)/2] \end{array}$$

Adding similar arguments, we get:

Proposition 4.4.5. *For each $d \in \mathcal{D}^\infty$, let $[a_d, b_d] = to_{\mathcal{J}}\, d$. We have:*

1. $to_{\mathcal{J}}(\mathsf{Cons_L}\ d) = [(a_d - 1)/2\ , (b_d - 1)/2]$
2. $to_{\mathcal{J}}(\mathsf{Cons_M}\ d) = [a_d/2\ , b_d/2]$
3. $to_{\mathcal{J}}(\mathsf{Cons_R}\ d) = [(a_d + 1)/2\ , (b_d + 1)/2]$

Lemma 4.4.6. *For any $d_x, e_x \in \mathcal{D}^\infty$ such that $d_x \simeq_{\mathcal{D}^\infty} e_x$, we have:*

1. $\mathsf{Cons_L}\ d_x \simeq_{\mathcal{D}^\infty} \mathsf{Cons_L}\ e_x$
2. $\mathsf{Cons_M}\ d_x \simeq_{\mathcal{D}^\infty} \mathsf{Cons_M}\ e_x$
3. $\mathsf{Cons_R}\ d_x \simeq_{\mathcal{D}^\infty} \mathsf{Cons_R}\ e_x$

Proof. Let $d_x, e_x \in \mathcal{D}^\omega$ be representations of the same total real number, say, $x \in [-1, 1]$, i. e.

$$x = to_{\mathcal{J}}\ d_x = to_{\mathcal{J}}\ e_x$$

Now by Proposition 4.4.5 we have:

$$\begin{array}{rcl} to_{\mathcal{J}}(\mathsf{Cons_L}\ d_x) & = & (x - 1)/2 \\ & = & to_{\mathcal{J}}(\mathsf{Cons_L}\ e_x) \end{array}$$

Therefore:

$$\mathsf{Cons_L}\ d_x \simeq_{\mathcal{D}^\infty} \mathsf{Cons_L}\ e_x$$

Similar arguments apply to both $\mathsf{Cons_M}$ and $\mathsf{Cons_R}$. □

Combining Proposition 4.4.5 and Lemma 4.4.6, one gets:

Corollary 4.4.7 (Extensionality). $\mathsf{Cons_L}$, $\mathsf{Cons_M}$ *and* $\mathsf{Cons_R}$ *are extensional over total real numbers and hence are interpreted as the following functions over* $[-1, 1]$*:*

$$\begin{cases} to_{\mathcal{J}_{max}\to\mathcal{J}_{max}}\ \mathsf{Cons_L} : [-1,1] \to [-1,1] \\ (to_{\mathcal{J}_{max}\to\mathcal{J}_{max}}\ \mathsf{Cons_L})(x) = (x-1)/2 \end{cases}$$

$$\begin{cases} to_{\mathcal{J}_{max}\to\mathcal{J}_{max}}\ \mathsf{Cons_M} : [-1,1] \to [-1,1] \\ (to_{\mathcal{J}_{max}\to\mathcal{J}_{max}}\ \mathsf{Cons_M})(x) = x/2 \end{cases}$$

$$\begin{cases} to_{\mathcal{J}_{max}\to\mathcal{J}_{max}}\ \mathsf{Cons_R} : [-1,1] \to [-1,1] \\ (to_{\mathcal{J}_{max}\to\mathcal{J}_{max}}\ \mathsf{Cons_R})(x) = (x+1)/2 \end{cases}$$

Notation 4.4.8. *Although we gave a proper name* $\mathsf{Cons_L}$ *to* $\mathcal{A}_{\mathcal{SH}}(L)$*, we will freely use the (Haskell style) notation* $\mathsf{L}\colon x$ *instead of the cumbersome* $\mathsf{Cons_L}\ x$ *wherever convenient. The same applies to* $\mathsf{M}\colon$ *and* $\mathsf{R}\colon$.

4.4.2 $\boxed{\mathsf{L}^{-1}, \mathsf{M}^{-1}, \mathsf{R}^{-1} : \mathcal{D}^\infty \to \mathcal{D}^\infty}$

For any real number $x \in (-1, 1)$, infinitely many representations are available inside $\mathcal{D}^\infty$. For instance, the following three sequences all represent the real number *zero*:

$$\begin{aligned} d_0 &= \mathsf{L}\colon \mathsf{R}\colon \mathsf{R}\colon \mathsf{R}\ldots \\ d_0' &= \mathsf{M}\colon \mathsf{M}\colon \mathsf{M}\colon \mathsf{M}\ldots \\ d_0'' &= \mathsf{R}\colon \mathsf{L}\colon \mathsf{L}\colon \mathsf{L}\ldots \end{aligned}$$

Now suppose that at some stage we get a sequence $d_s = \mathsf{L}\colon x$ representing a real number $s \in [-1, 1]$, and for some reason we need to rewrite the sequence in such a way that it starts with, say, M, and of course, the resulting sequence $e_s = \mathsf{M}\colon x'$ also represents s. Well, first of all this is not possible at all unless $s \in [-1/2, 1/2]$, in which case of course, there is a straightforward way of getting e_s from d_s. But even in case s lies outside $[-1/2, 1/2]$, we can still produce some $e_{s'} = \mathsf{M}\colon x'$ which represents one of the two values $\{-1/2, 1/2\}$, i. e. the one closer to s. So let us start implementing M^{-1}.

The implementation is a bit involved, hence we break it into different stages. We need a few definitions to begin with.

Definition 4.4.9 ($d_{-1}, d_1 \in \mathcal{D}^\infty$). *We let d_{-1}, $d_1 \in \mathcal{D}^\infty$ denote the representations of the real numbers -1 and 1, respectively, i. e.*

$$\begin{aligned} d_{-1} &= \mathsf{L}\colon \mathsf{L}\colon \mathsf{L}\colon \ldots \\ d_1 &= \mathsf{R}\colon \mathsf{R}\colon \mathsf{R}\colon \ldots \end{aligned}$$

Next we define (versions of) (-1) and $(+1)$ functions over $\mathcal{D}^\infty$:

Definition 4.4.10 ($(-1)_{\mathcal{D}^\infty}, (+1)_{\mathcal{D}^\infty}$).

$$\begin{cases} (-1)_{\mathcal{D}^\infty} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ (-1)_{\mathcal{D}^\infty}(\mathsf{L}\colon x) = d_{-1} \\ (-1)_{\mathcal{D}^\infty}(\mathsf{M}\colon x) = \mathsf{L}\colon ((-1)_{\mathcal{D}^\infty}\ x) \\ (-1)_{\mathcal{D}^\infty}(\mathsf{R}\colon x) = \mathsf{L}\colon x \end{cases} \qquad \begin{cases} (+1)_{\mathcal{D}^\infty} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ (+1)_{\mathcal{D}^\infty}(\mathsf{L}\colon x) = \mathsf{R}\colon x \\ (+1)_{\mathcal{D}^\infty}(\mathsf{M}\colon x) = \mathsf{R}\colon ((+1)_{\mathcal{D}^\infty}\ x) \\ (+1)_{\mathcal{D}^\infty}(\mathsf{R}\colon x) = d_1 \end{cases}$$

Proposition 4.4.11 (totality of $(-1)_{\mathcal{D}^\infty}$ and $(+1)_{\mathcal{D}^\infty}$).

$$\forall x \in \mathcal{D}^\omega : \begin{cases} ((-1)_{\mathcal{D}^\infty}\ x) \in \mathcal{D}^\omega \\ ((+1)_{\mathcal{D}^\infty}\ x) \in \mathcal{D}^\omega \end{cases}$$

Proof. Left to the reader. □

We also define the counterpart (-1) and $(+1)$ functions over $\mathcal{J}$:

Definition 4.4.12 ($(-1)_{\mathcal{J}}, (+1)_{\mathcal{J}}$).

$$\begin{cases} (-1)_{\mathcal{J}} : \mathcal{J} \to \mathcal{J} \\ (-1)_{\mathcal{J}}[a, b] = [\max\{-1, a-1\}, \max\{-1, b-1\}] \end{cases}$$

$$\begin{cases} (+1)_{\mathcal{J}} : \mathcal{J} \to \mathcal{J} \\ (+1)_{\mathcal{J}}[a, b] = [\min\{1, a+1\}, \min\{1, b+1\}] \end{cases}$$

Proposition 4.4.13.

$$\forall x \in \mathcal{D}^\infty : \begin{cases} to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ x) \sqsubseteq (-1)_{\mathcal{J}}(to_{\mathcal{J}}\ x) \\ to_{\mathcal{J}}((+1)_{\mathcal{D}^\infty}\ x) \sqsubseteq (+1)_{\mathcal{J}}(to_{\mathcal{J}}\ x) \end{cases}$$

Proof. We just consider the case of $(-1)_{\mathcal{D}^\infty}$. The proof for $(+1)_{\mathcal{D}^\infty}$ is similar. In any case, the proof is done by induction over x:

1. case $x = [\,]$: we have

$$\begin{array}{rcl}
to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ x) & = & to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ [\,]) \\
(\text{definition of } (-1)_{\mathcal{D}^\infty}) & = & to_{\mathcal{J}}([\,]) \\
(\text{definition of } to_{\mathcal{J}}) & = & [-1,1] \\
 & \sqsubseteq & [-1,0] \\
(\text{definitions of } (-1)_{\mathcal{J}} \text{ and } to_{\mathcal{J}}) & = & (-1)_{\mathcal{J}}(to_{\mathcal{J}}[\,]) \\
 & = & (-1)_{\mathcal{J}}(to_{\mathcal{J}}\ x)
\end{array}$$

2. case $x = \mathsf{L}\colon x'$: we have

$$\begin{array}{rcl}
to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ x) & = & to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ (\mathsf{L}\colon x')) \\
(\text{definition of } (-1)_{\mathcal{D}^\infty}) & = & to_{\mathcal{J}}(d_{-1}) \\
(\text{definition of } d_{-1}) & = & [-1,1] \\
(\text{easy calculations}) & = & (-1)_{\mathcal{J}}\mathsf{Cons}_{[-1,0]}(to_{\mathcal{J}}x') \\
(\text{definition of } to_{\mathcal{J}}) & = & (-1)_{\mathcal{J}}(to_{\mathcal{J}}\ (\mathsf{L}\colon x')) \\
 & = & (-1)_{\mathcal{J}}\ (to_{\mathcal{J}}\ x)
\end{array}$$

3. case $x = \mathsf{M}\colon x'$: we have

$$\begin{array}{rcl}
to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ x) & = & to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ (\mathsf{M}\colon x')) \\
(\text{definition of } (-1)_{\mathcal{D}^\infty}) & = & to_{\mathcal{J}}(\mathsf{L}\colon ((-1)_{\mathcal{D}^\infty}\ x')) \\
(\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Cons}_{[-1,0]}\ (to_{\mathcal{J}}\ ((-1)_{\mathcal{D}^\infty}\ x')) \\
(\text{induction hypothesis}) & \sqsubseteq & \mathsf{Cons}_{[-1,0]}\ ((-1)_{\mathcal{J}}\ (to_{\mathcal{J}}\ x')) \\
(\text{cumbersome calculations}) & = & (-1)_{\mathcal{J}}\mathsf{Cons}_{[-1/2,1/2]}(to_{\mathcal{J}}\ x') \\
(\text{definition of } to_{\mathcal{J}}) & = & (-1)_{\mathcal{J}}(to_{\mathcal{J}}\ (\mathsf{M}\colon x')) \\
 & = & (-1)_{\mathcal{J}}\ (to_{\mathcal{J}}\ x)
\end{array}$$

4. case $x = \mathsf{R}\colon x'$: we have

$$\begin{array}{rcl}
to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ x) & = & to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ (\mathsf{R}\colon x')) \\
(\text{definition of } (-1)_{\mathcal{D}^\infty}) & = & to_{\mathcal{J}}(\mathsf{L}\colon x') \\
(\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Cons}_{[-1,0]}\ (to_{\mathcal{J}}\ x') \\
(\text{easy calculations}) & = & (-1)_{\mathcal{J}}\mathsf{Cons}_{[0,1]}(to_{\mathcal{J}}\ x') \\
(\text{definition of } to_{\mathcal{J}}) & = & (-1)_{\mathcal{J}}(to_{\mathcal{J}}\ (\mathsf{R}\colon x')) \\
 & = & (-1)_{\mathcal{J}}\ (to_{\mathcal{J}}\ x)
\end{array}$$

□

Combining Propositions 4.4.11 and 4.4.13 we get:

Corollary 4.4.14 (Correctness of $(-1)_{\mathcal{D}^\infty}$ and $(+1)_{\mathcal{D}^\infty}$). *For any $d_x \in \mathcal{D}^\omega$ representing a real number $x \in [-1, 1]$:*

$$\begin{cases} to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty}\ d_x) = \max\{-1, x-1\} \\ to_{\mathcal{J}}((+1)_{\mathcal{D}^\infty}\ d_x) = \min\{1, x+1\} \end{cases}$$

which leads to:

Corollary 4.4.15 (Extensionality of $(-1)_{\mathcal{D}^\infty}$ and $(+1)_{\mathcal{D}^\infty}$).

$$\forall d_x, e_x \in \mathcal{D}^\infty : d_x \simeq_{\mathcal{D}^\infty} e_x \Rightarrow \begin{cases} (-1)_{\mathcal{D}^\infty}\ d_x \simeq_{\mathcal{D}^\infty} (-1)_{\mathcal{D}^\infty}\ e_x \\ (+1)_{\mathcal{D}^\infty}\ d_x \simeq_{\mathcal{D}^\infty} (+1)_{\mathcal{D}^\infty}\ e_x \end{cases}$$

We are now in a stage where we can define our main functions:

Definition 4.4.16 ($\mathsf{L}^{-1}, \mathsf{M}^{-1}, \mathsf{R}^{-1} : \mathcal{D}^\infty \to \mathcal{D}^\infty$).

$$\begin{cases} \mathsf{L}^{-1} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{L}^{-1}(\mathsf{L} : x) = x \\ \mathsf{L}^{-1}(\mathsf{M} : x) = (+1)_{\mathcal{D}^\infty}\ x \\ \mathsf{L}^{-1}(\mathsf{R} : x) = d_1 \end{cases} \quad \begin{cases} \mathsf{M}^{-1} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{M}^{-1}(\mathsf{L} : x) = (-1)_{\mathcal{D}^\infty}\ x \\ \mathsf{M}^{-1}(\mathsf{M} : x) = x \\ \mathsf{M}^{-1}(\mathsf{R} : x) = (+1)_{\mathcal{D}^\infty}\ x \end{cases} \quad \begin{cases} \mathsf{R}^{-1} : \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{R}^{-1}(\mathsf{L} : x) = d_{-1} \\ \mathsf{R}^{-1}(\mathsf{M} : x) = (-1)_{\mathcal{D}^\infty}\ x \\ \mathsf{R}^{-1}(\mathsf{R} : x) = x \end{cases}$$

Lemma 4.4.17 (totality of L^{-1}, M^{-1} and R^{-1}).

$$\forall x \in \mathcal{D}^\omega : \begin{cases} \mathsf{L}^{-1} x \in \mathcal{D}^\omega \\ \mathsf{M}^{-1} x \in \mathcal{D}^\omega \\ \mathsf{R}^{-1} x \in \mathcal{D}^\omega \end{cases}$$

Proof. Immediate from the definition using Proposition 4.4.11. □

Lemma 4.4.18. *For all $x \in \mathcal{D}^\infty$:*

$$\begin{cases} to_{\mathcal{J}}(\mathsf{L}^{-1} x) \sqsubseteq \mathsf{Tail}_{[-1,0]}\ (to_{\mathcal{J}}(x)) \\ to_{\mathcal{J}}(\mathsf{M}^{-1} x) \sqsubseteq \mathsf{Tail}_{[-1/2,1/2]}\ (to_{\mathcal{J}}(x)) \\ to_{\mathcal{J}}(\mathsf{R}^{-1} x) \sqsubseteq \mathsf{Tail}_{[0,1]}\ (to_{\mathcal{J}}(x)) \end{cases}$$

Proof. We just consider the case of M^{-1}. The other two cases follow a similar procedure. In any case, the proof is done by induction over x:

1. case $x = [\,]$: Immediate as both sides evaluate to $[-1, 1]$.

2. case $x = \mathsf{L}\colon x'$: We have:

$$\begin{array}{rcl}
to_{\mathcal{J}}(\mathsf{M}^{-1}x) & = & to_{\mathcal{J}}(\mathsf{M}^{-1}(\mathsf{L}\colon x')) \\
(\text{definition of } \mathsf{M}^{-1}) & = & to_{\mathcal{J}}((-1)_{\mathcal{D}^\infty} x') \\
(\text{Proposition 4.4.13 on page 117}) & \sqsubseteq & (-1)_{\mathcal{J}}(to_{\mathcal{J}}\, x') \\
(\text{easy observation}) & = & \mathsf{Tail}_{[-1/2,1/2]}\mathsf{Cons}_{[-1,0]}(to_{\mathcal{J}}\, x') \\
(\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Tail}_{[-1/2,1/2]}(to_{\mathcal{J}}(\mathsf{L}\colon x')) \\
 & = & \mathsf{Tail}_{[-1/2,1/2]}(to_{\mathcal{J}}\, x)
\end{array}$$

3. case $x = \mathsf{M}\colon x'$: We have:

$$\begin{array}{rcl}
to_{\mathcal{J}}(\mathsf{M}^{-1}x) & = & to_{\mathcal{J}}(\mathsf{M}^{-1}(\mathsf{M}\colon x')) \\
(\text{definition of } \mathsf{M}^{-1}) & = & to_{\mathcal{J}}(x') \\
(\mathsf{Tail}_{[-1/2,1/2]}\mathsf{Cons}_{[-1/2,1/2]} = id_{\mathcal{J}}) & = & \mathsf{Tail}_{[-1/2,1/2]}\mathsf{Cons}_{[-1/2,1/2]}(to_{\mathcal{J}}\, x') \\
(\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Tail}_{[-1/2,1/2]}(to_{\mathcal{J}}(\mathsf{M}\colon x')) \\
 & = & \mathsf{Tail}_{[-1/2,1/2]}(to_{\mathcal{J}}\, x)
\end{array}$$

4. case $x = \mathsf{R}\colon x'$: We have:

$$\begin{array}{rcl}
to_{\mathcal{J}}(\mathsf{M}^{-1}x) & = & to_{\mathcal{J}}(\mathsf{M}^{-1}(\mathsf{R}\colon x')) \\
(\text{definition of } \mathsf{M}^{-1}) & = & to_{\mathcal{J}}((+1)_{\mathcal{D}^\infty} x') \\
(\text{Proposition 4.4.13 on page 117}) & \sqsubseteq & (+1)_{\mathcal{J}}(to_{\mathcal{J}}\, x') \\
(\text{easy observation}) & = & \mathsf{Tail}_{[-1/2,1/2]}\mathsf{Cons}_{[0,1]}(to_{\mathcal{J}}\, x') \\
(\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Tail}_{[-1/2,1/2]}(to_{\mathcal{J}}(\mathsf{R}\colon x')) \\
 & = & \mathsf{Tail}_{[-1/2,1/2]}(to_{\mathcal{J}}\, x)
\end{array}$$

□

Combining Lemmata 4.4.17 and 4.4.18 we get:

Corollary 4.4.19. *For any $d_x \in \mathcal{D}^\omega$ representing a real number $x \in [-1, 1]$:*

$$to_{\mathcal{J}}(\mathsf{L}^{-1}\ d_x) \quad = \quad \begin{cases} 2x + 1 & \text{if} \quad x \in [-1, 0] \\ 1 & \text{if} \quad x \in [0, 1] \end{cases}$$

$$to_{\mathcal{J}}(\mathsf{M}^{-1}\ d_x) \quad = \quad \begin{cases} -1 & \text{if} \quad x \in [-1, -1/2] \\ 2x & \text{if} \quad x \in [-1/2, 1/2] \\ 1 & \text{if} \quad x \in [1/2, 1] \end{cases}$$

$$to_{\mathcal{J}}(\mathsf{R}^{-1}\ d_x) \quad = \quad \begin{cases} -1 & \text{if} \quad x \in [-1, 0] \\ 2x - 1 & \text{if} \quad x \in [0, 1] \end{cases}$$

which leads to:

Corollary 4.4.20 (Extensionality of L^{-1}, M^{-1} and R^{-1}).

$$\forall d_x, e_x \in \mathcal{D}^\infty : d_x \simeq_{\mathcal{D}^\infty} e_x \Rightarrow \begin{cases} \mathsf{L}^{-1} d_x \simeq_{\mathcal{D}^\infty} \mathsf{L}^{-1} e_x \\ \mathsf{M}^{-1} d_x \simeq_{\mathcal{D}^\infty} \mathsf{M}^{-1} e_x \\ \mathsf{R}^{-1} d_x \simeq_{\mathcal{D}^\infty} \mathsf{R}^{-1} e_x \end{cases}$$

4.4.3 $\boxed{\textbf{qtoR}: \mathbb{N}_\perp \to \mathbb{N}_\perp \to \mathbb{N}_\perp \to \mathcal{D}^\infty}$

We need an operator over triplets of natural numbers such that on any

$$(p, m, n) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}$$

satisfying:

1. $n \neq 0$
2. $m/n \leq 1$

returns some maximal element of $\mathcal{D}^\infty$ representing the rational number:

1. m/n if $p > 0$.
2. $-m/n$ if $p = 0$.

First assume that:

$$\widehat{\mathsf{qtoR}}\colon \mathbb{Q} \cap [-1, 1] \to \mathcal{D}^\infty$$

is some (effective or non-effective) function which does the same thing but acts directly on rational numbers, i. e. $\widehat{\mathsf{qtoR}}$ picks a canonical representation for each rational number in its domain. Therefore:

$$\forall q \in \mathbb{Q} \cap [-1, 1]\colon to_{\mathcal{J}}(\widehat{\mathsf{qtoR}}\ q) = q$$

It is easy to verify that:

1. $\forall q \in \mathbb{Q} \cap [0, 1]\colon q = to_{\mathcal{J}}(\mathsf{R}\colon (\widehat{\mathsf{qtoR}}(\mathsf{Tail}_{[0,1]}\ q)))$
2. $\forall q \in \mathbb{Q} \cap [-1, 0]\colon q = to_{\mathcal{J}}(\mathsf{L}\colon (\widehat{\mathsf{qtoR}}(\mathsf{Tail}_{[-1,0]}\ q)))$
3. $\forall q \in \mathbb{Q} \cap [-1/2, 1/2]\colon q = to_{\mathcal{J}}(\mathsf{M}\colon (\widehat{\mathsf{qtoR}}(\mathsf{Tail}_{[-1/2,1/2]}\ q)))$

In fact, we can *effectively* define such a function:

Definition 4.4.21 ($\widehat{\mathsf{qtoR}}$).

$$\widehat{\mathsf{qtoR}}\colon \mathbb{Q} \cap [-1, 1] \to \mathcal{D}^\infty$$

$$\widehat{\mathsf{qtoR}}\ q = \begin{cases} \mathsf{R}\colon (\widehat{\mathsf{qtoR}}(\mathsf{Tail}_{[0,1]}\ q))) & \text{if} \quad q > 0 \\ \mathsf{L}\colon (\widehat{\mathsf{qtoR}}(\mathsf{Tail}_{[-1,0]}\ q))) & \text{if} \quad q < 0 \\ \mathsf{M}\colon (\widehat{\mathsf{qtoR}}(\mathsf{Tail}_{[-1/2,1/2]}\ q))) & \text{otherwise} \end{cases}$$

Going back to the triplets of natural numbers, we define our main function as:

Definition 4.4.22 (qtoR). *The function* $\mathsf{qtoR}\colon \mathbb{N}_\perp \to \mathbb{N}_\perp \to \mathbb{N}_\perp \to \mathcal{D}^\infty$ *is defined as follows:*

$$\mathsf{qtoR}\ p\ m\ n = \begin{cases} [\,] & \text{if} \quad n = 0 \\ \widehat{\mathsf{qtoR}}\ 1 & \text{if} \quad (\frac{m}{n} > 1) \wedge (p > 0) \\ \widehat{\mathsf{qtoR}}\ (-1) & \text{if} \quad (\frac{m}{n} > 1) \wedge (p = 0) \\ \widehat{\mathsf{qtoR}}\ (\frac{m}{n}) & \text{if} \quad (\frac{m}{n} \le 1) \wedge (p > 0) \\ \widehat{\mathsf{qtoR}}\ (-\frac{m}{n}) & \text{if} \quad (\frac{m}{n} \le 1) \wedge (p = 0) \end{cases}$$

The following lemmata are now immediate:

Lemma 4.4.23 (correctness of qtoR).

$$\forall p, m, n \in \mathbb{N} \colon to_{\mathcal{J}}(\mathsf{qtoR}\ p\ m\ n) = \begin{cases} [-1,1] & \textit{if} \quad n = 0 \\ 1 & \textit{if} \quad (m/n > 1) \wedge (p > 0) \\ -1 & \textit{if} \quad (m/n > 1) \wedge (p = 0) \\ m/n & \textit{if} \quad (m/n \leq 1) \wedge (p > 0) \\ -m/n & \textit{if} \quad (m/n \leq 1) \wedge (p = 0) \end{cases}$$

Notation 4.4.24. *We say that a triplet* $(p, m, n) \in \mathbb{N}\times\mathbb{N}\times\mathbb{N}$ represents *the rational number* $q \in \mathbb{Q} \cap [-1, 1]$ *if:*

1. $n \neq 0$
2. $\begin{cases} (p > 0) \wedge (m/n > 1) & \textit{if} \quad q = 1 \\ (p = 0) \wedge (m/n > 1) & \textit{if} \quad q = -1 \\ (p > 0) \wedge (m/n = q) & \textit{if} \quad 0 \leq q \leq 1 \\ (p = 0) \wedge (-m/n = q) & \textit{if} \quad -1 \leq q \leq 0 \end{cases}$

Lemma 4.4.25 (totality of qtoR). *On triplets* (p, m, n) *representing rational numbers, i. e.* $n \neq 0$*,* (qtoR $p\ m\ n$) *is an infinite sequence of digits:*

$$\forall (p, m, n) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N} \colon n \neq 0 \Rightarrow \mathsf{qtoR}\ p\ m\ n \in \mathcal{D}^{\omega}$$

4.4.4 $\boxed{\mathsf{avg} \colon \mathcal{D}^{\infty} \to \mathcal{D}^{\infty} \to \mathcal{D}^{\infty}}$

We are going to present an implementation of the *average* function over $\mathcal{D}^{\infty}$, written in Haskell style to emphasize the sequential nature of the implementation. We essentially define the function over a few cases and then using commutativity law and the negation function, refer other cases to these few as well. But it is useful to emphasize that the definition could be done in such a way that no case was converted to any other case, only it would be much longer and more cumbersome than it is already.

Let us first define the auxiliary function neg:

Definition 4.4.26 (neg: $\mathcal{D}^{\infty} \to \mathcal{D}^{\infty}$).

$$\begin{aligned} \mathsf{neg}\ \ [\,] \ &= [\,] \\ \mathsf{neg}\ (\mathsf{L}\colon x) &= \mathsf{R}\colon (\mathsf{neg}(x)) \\ \mathsf{neg}\ (\mathsf{M}\colon x) &= \mathsf{M}\colon (\mathsf{neg}(x)) \\ \mathsf{neg}\ (\mathsf{R}\colon x) &= \mathsf{L}\colon (\mathsf{neg}(x)) \end{aligned}$$

Proposition 4.4.27. neg *satisfies the following properties:*

1. *(totality):* $\forall d \in \mathcal{D}^\omega : \mathsf{neg}(d) \in \mathcal{D}^\omega$
2. *(extensionality) :* $\forall d_x, d_y \in \mathcal{D}^\infty : d_x \simeq_{\mathcal{D}^\infty} d_y \Rightarrow \mathsf{neg}(d_x) \simeq_{\mathcal{D}^\infty} \mathsf{neg}(d_y)$
3. *(correctness) : For any* $d \in \mathcal{D}^\infty$*, let* $[a_d, b_d] = to_{\mathcal{J}}(d)$*. Therefore we have:*

$$to_{\mathcal{J}}(\mathsf{neg}(d)) = [-b_d, -a_d]$$

and in particular:

$$\forall d \in \mathcal{D}^\omega : to_{\mathcal{J}}(\mathsf{neg}(d)) = -to_{\mathcal{J}}(d)$$

Proof. Left to the reader. □

Definition 4.4.28 ($\mathsf{avg}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$). *The function* avg *is defined as in Table 4.2 on the facing page.*

Lemma 4.4.29 (continuity of avg). $\mathsf{avg}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$ *is continuous.*

Proof. Immediate from the definition. □

Lemma 4.4.30 (avg is total). *The result of applying* avg *to representations of* total *real numbers is the representation of a* total *real number, i. e.*

$$\forall x, y \in \mathcal{D}^\omega : \mathsf{avg}\ x\ y\ \in \mathcal{D}^\omega$$

Proof. (sketch) At each stage, avg only needs at most two digits from each of its arguments to produce one digit of the output. □

To ensure that avg, when interpreted as an operation over real numbers, correctly calculates the *average* of its arguments, we introduce the following definition:

Definition 4.4.31 ($\oplus\colon \mathcal{J} \to \mathcal{J} \to \mathcal{J}$).

$$\forall [x_1, y_1], [x_2, y_2] \in \mathcal{J} : [x_1, y_1] \oplus [x_2, y_2] := [(x_1 + x_2)/2, (y_1 + y_2)/2]$$

Lemma 4.4.32.

$$\forall a, b \in \mathcal{D}^\infty : to_{\mathcal{J}}\ (\ \mathsf{avg}\ a\ b\ ) \sqsubseteq_{\mathcal{J}} (to_{\mathcal{J}}\ a) \oplus (to_{\mathcal{J}}\ b)$$

Table 4.2 The function avg : $\mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$

avg	(L: x)	(L: y)	=	L: (avg x y)
avg	(L: x)	(R: y)	=	M: (avg x y)
avg	(L: L: x)	(M: L: y)	=	L: M: (avg x y)
avg	(L: L: x)	(M: M: y)	=	L: (avg (M: x) (R: y))
avg	(L: L: x)	(M: R: y)	=	L: R: (avg x y)
avg	(L: M: x)	(M: L: y)	=	L: (avg (M: x) (R: y))
avg	(L: M: x)	(M: M: y)	=	L: R: (avg x y)
avg	(L: M: x)	(M: R: y)	=	M: (avg (L: x) (M: y))
avg	(L: R: x)	(M: L: y)	=	L: R: (avg x y)
avg	(L: R: x)	(M: M: y)	=	M: (avg (M: x) (L: y))
avg	(L: R: x)	(M: R: y)	=	M: M: (avg x y)
avg	(M: x)	(M: y)	=	M: (avg x y)
avg	(M: x)	(L: y)	=	avg(L: y)(M: x)
avg	(M: x)	(R: y)	=	neg(avg (M: (neg(x))) (L: (neg(y))))
avg	(R: x)	(R: y)	=	R: (avg x y)
avg	(R: x)	(L: y)	=	M: (avg x y)
avg	(R: x)	(M: y)	=	avg (M: y) (R: x)

Proof. We start by proving the lemma for elements of $\mathcal{D}^*$, i. e. finite sequences of digits. This we can achieve using induction over length of a and b. So suppose $a, b \in \mathcal{D}^*$:

base case $a = b = [\,]$: In this case lemma holds because avg [ ] [ ] = [ ], therefore:

$$\begin{aligned} to_{\mathcal{J}}(\,\mathsf{avg}\,[\,]\,[\,]\,) &= to_{\mathcal{J}}([\,]) \\ &= [-1, 1] \\ &\sqsubseteq [-1, 1] \oplus [-1, 1] \\ &= to_{\mathcal{J}}([\,]) \oplus to_{\mathcal{J}}([\,]) \end{aligned}$$

induction case: Let $a = a_0 \ldots a_n$, $b = b_0 \ldots b_m$ and suppose that the lemma holds for sequences of digits c with *length* $c < \max\{m, n\}$. We need to really check each individual case in Table 4.2 above for avg. But here we merely content ourselves with verifying one case, leaving the others to the ambitious reader as they are similarly straightforward. As an example, let:

$$a = \mathsf{L}\colon \mathsf{L}\colon a_2 \ldots a_n$$
$$b = \mathsf{M}\colon \mathsf{L}\colon b_2 \ldots b_m$$

In other words:

$$a_0 = a_1 = \mathsf{L}$$

and

$$b_0 = \mathsf{M}, \qquad b_1 = \mathsf{L}$$

We also let $a'' = a_2 \ldots a_n$ and $b'' = b_2 \ldots b_m$. Thus we can write:

$$\begin{array}{rcl}
to_{\mathcal{J}}\,(\mathsf{avg}\ a\ b\ ) & = & to_{\mathcal{J}}\,(\mathsf{avg}\ (\mathsf{L}\colon \mathsf{L}\colon a'')\ (\mathsf{M}\colon \mathsf{L}\colon b'')\ ) \\
\text{(definition of avg)} & = & to_{\mathcal{J}}\,(\mathsf{L}\colon \mathsf{M}\colon (\ \mathsf{avg}\ a''\ b''\ )) \\
\text{(definition of } to_{\mathcal{J}}) & = & \mathsf{Cons}_{[-1,0]}\ \mathsf{Cons}_{[-1/2,1/2]}\ to_{\mathcal{J}}\ (\ \mathsf{avg}\ a''\ b''\ ) \\
\text{(induction hypothesis)} & \sqsubseteq & \mathsf{Cons}_{[-1,0]}\ \mathsf{Cons}_{[-1/2,1/2]}((to_{\mathcal{J}}\ a'') \oplus (to_{\mathcal{J}}\ b'')) \\
\text{(a little bit of calculation)} & = & (\mathsf{Cons}_{[-1,0]}\mathsf{Cons}_{[-1,0]}(to_{\mathcal{J}}\ a'')) \oplus \\
 & & (\mathsf{Cons}_{[-1/2,1/2]}\mathsf{Cons}_{[-1,0]}(to_{\mathcal{J}}\ b'')) \\
\text{(definition of } to_{\mathcal{J}}) & = & (to_{\mathcal{J}}\,(\mathsf{L}\colon \mathsf{L}\colon a'')) \oplus (to_{\mathcal{J}}\,(\mathsf{M}\colon \mathsf{L}\colon b'')) \\
 & = & (to_{\mathcal{J}}\ a) \oplus (to_{\mathcal{J}}\ b)
\end{array}$$

Now we extend the result to the whole $\mathcal{D}^\infty$. For any $x \in \mathcal{D}^\infty$, we let $x_{<n}$ denote the initial segment of x of length not more than n. So in particular:

$$length\ x < n \Rightarrow x_{<n} = x$$

For any $x \in \mathcal{D}^\infty$, we have:

$$x = \bigsqcup_{n\in\mathbb{N}} x_{<n}$$

Therefore one can write:

$$\begin{array}{rcl}
to_{\mathcal{J}}\,(\ \mathsf{avg}\ a\ b\ ) & = & to_{\mathcal{J}}\,(\ \mathsf{avg}\ (\bigsqcup_{n\in\mathbb{N}} a_{<n})\ (\bigsqcup_{n\in\mathbb{N}} b_{<n})) \\
\text{(continuity of avg)} & = & to_{\mathcal{J}}\,(\ \bigsqcup_{m,n\in\mathbb{N}} \mathsf{avg}\ a_{<m}\ b_{<n}) \\
\text{(continuity of } to_{\mathcal{J}}) & = & \bigsqcup_{m,n\in\mathbb{N}} (to_{\mathcal{J}}\,(\ \mathsf{avg}\ a_{<m}\ b_{<n}\ )) \\
\text{(by first part of the proof)} & \sqsubseteq & \bigsqcup_{m,n\in\mathbb{N}} ((to_{\mathcal{J}}\ a_{<m}) \oplus (to_{\mathcal{J}}\ b_{<n})) \\
\text{(continuity of } \oplus) & = & (\ \bigsqcup_{m\in\mathbb{N}} to_{\mathcal{J}}\ a_{<m}\ ) \oplus (\ \bigsqcup_{n\in\mathbb{N}} to_{\mathcal{J}}\ b_{<n}\ ) \\
\text{(continuity of } to_{\mathcal{J}}) & = & to_{\mathcal{J}}(\bigsqcup_{n\in\mathbb{N}} a_{<n}) \oplus to_{\mathcal{J}}(\bigsqcup_{n\in\mathbb{N}} b_{<n}) \\
 & = & to_{\mathcal{J}}(a) \oplus to_{\mathcal{J}}(b)
\end{array}$$

□

Combining lemmata 4.4.30 and 4.4.32 results in:

Corollary 4.4.33 (correctness of avg). *For elements $a, b \in \mathcal{D}^\omega$, i. e. infinite sequences of digits:*

$$to_{\mathcal{I}}\ (\ \mathsf{avg}\ a\ b\ ) = (to_{\mathcal{I}}\ a) \oplus (to_{\mathcal{I}}\ b)$$

Apart from correctness, the above corollary demonstrates the extensionality of avg at total real numbers as well. We already discussed that avg is not necessarily extensional at all partial real numbers (see equation (4.2) on page 113).

Now assume $d_x, e_x \in \mathcal{D}^\omega$ are two representations of the same number $x \in [-1, 1]$, and $d_y, e_y \in \mathcal{D}^\omega$ that of $y \in [-1, 1]$. In other words:

$$\begin{aligned} to_{\mathcal{I}}\ d_x &= to_{\mathcal{I}}\ e_x = x \\ to_{\mathcal{I}}\ d_y &= to_{\mathcal{I}}\ e_y = y \end{aligned}$$

By corollary 4.4.33 above, we have:

$$\begin{aligned} to_{\mathcal{I}}\ (\ \mathsf{avg}\ d_x\ d_y\ ) &= (to_{\mathcal{I}}\ d_x) \oplus (to_{\mathcal{I}}\ d_y) \\ &= x \oplus y \\ &= (to_{\mathcal{I}}\ e_x) \oplus (to_{\mathcal{I}}\ e_y) \\ &= to_{\mathcal{I}}\ (\ \mathsf{avg}\ e_x\ e_y\ ) \end{aligned}$$

Corollary 4.4.34 (extensionality of avg). avg *preserves the equivalence relation $\simeq_{\mathcal{D}^\infty}$, i. e. for all $d_x, e_x, d_y, e_y \in \mathcal{D}^\infty$:*

$$\left.\begin{array}{l} d_x \simeq_{\mathcal{D}^\infty} e_x \\ d_y \simeq_{\mathcal{D}^\infty} e_y \end{array}\right\} \Rightarrow (\ \mathsf{avg}\ d_x\ d_y\ ) \simeq_{\mathcal{D}^\infty} (\ \mathsf{avg}\ e_x\ e_y\ )$$

4.4.5 $\boxed{\mathsf{mult} : \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty}$

Picking up pen and paper, trying to test one's own skills at implementing operations on real numbers in signed-binary format, the average operation should cause no trouble unless the lengthy mess of checking too many cases appears cumbersome, otherwise it is straightforward. As for *multiplication*, it is a rather different story. The implementation is not at all trivial. The one we present here was worked out by Plume [1998].

Again, we need a few definitions first:

Definition 4.4.35 (dp). *Consider the set $\mathcal{D} = \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$ of digits and the following pair of operations:*

$$\begin{array}{ll} e\colon \mathcal{D} \to \mathbb{Z} & r\colon \mathbb{Z} \to \mathcal{D} \\ e(\mathsf{L}) = -1 & \\ e(\mathsf{M}) = 0 & r(x) = \begin{cases} \mathsf{L} & \textit{if} \quad x < 0 \\ \mathsf{M} & \textit{if} \quad x = 0 \\ \mathsf{R} & \textit{if} \quad x > 0 \end{cases} \\ e(\mathsf{R}) = 1 & \end{array}$$

The product $\mathsf{dp}\colon \mathcal{D} \to \mathcal{D} \to \mathcal{D}$ over the digits is defined by:

$$\forall d_1, d_2 \in \mathcal{D}\colon \mathsf{dp}\ d_1\ d_2 = r(e(d_1) \times e(d_2))$$

where $\times$ is the ordinary product over the integers. This can be summarized in:

dp	L	M	R
L	R	M	L
M	M	M	M
R	L	M	R

Definition 4.4.36 (dsp). *A special product of digits and sequences is defined by:*

$$\begin{array}{l} \mathsf{dsp}\colon \mathcal{D} \to \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{dsp}\ \mathsf{L}\ d = \mathsf{neg}(d) \\ \mathsf{dsp}\ \mathsf{M}\ d = d_0 \\ \mathsf{dsp}\ \mathsf{R}\ d = d \end{array}$$

where $d_0 \in \mathcal{D}^\infty$ is some representation of the real number zero*, say:*

$$d_0 = \mathsf{M}\colon \mathsf{M}\colon \mathsf{M}\colon \cdots$$

Now we are ready to present the definition of mult (again) in Haskell style:

Definition 4.4.37 (mult). *The function $\mathsf{mult}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$ is defined by:*

$$\mathsf{mult}\ (a_0\colon a_1\colon x)\ (b_0\colon b_1\colon y) = \mathsf{avg}\ p\ q$$

where:

$$\begin{cases} p & = & \mathsf{avg}\ p_1\ p_2 \\ p_1 & = & (\mathsf{dp}\ a_0\ b_1)\colon (\mathsf{avg}\ p_{11}\ p_{12}) \\ p_{11} & = & \mathsf{dsp}\ b_1\ x \\ p_{12} & = & \mathsf{dsp}\ a_1\ y \\ p_2 & = & \mathsf{avg}\ p_{21}\ p_{22} \\ p_{21} & = & \mathsf{dsp}\ b_0\ x \\ p_{22} & = & \mathsf{dsp}\ a_0\ y \end{cases}$$

and

$$\begin{cases} q & = & c_0 : c_1 : c_2 : (\mathsf{mult}\ x\ y) \\ c_0 & = & \mathsf{dp}\ a_0\ b_0 \\ c_1 & = & \mathsf{dp}\ a_1\ b_0 \\ c_2 & = & \mathsf{dp}\ a_1\ b_1 \end{cases}$$

Again — much like the case of avg — continuity is immediate:

Lemma 4.4.38 (continuity of mult). $\mathsf{mult}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$ *is continuous.*

The difficulty with implementing multiplication is that one has to make sure the algorithm produces output once provided with enough input digits. The obvious implementations do not satisfy this property, whereas Plume's algorithm does. Let us consider the following two infinite sequences:

$$\begin{array}{ll} a = a_0 : a_1 : x & (x \in \mathcal{D}^\omega) \\ b = b_0 : b_1 : y & (y \in \mathcal{D}^\omega) \end{array}$$

According to the definition of mult:

$$\mathsf{mult}\ a\ b \ = \ \mathsf{avg}\ p\ q$$

where p is readily available as it makes no recursive calls to mult itself, and q is of the form:

$$q = c_0 : c_1 : c_2 : (\mathsf{mult}\ x\ y)$$

in which c_0, c_1 and c_2 are expressed in terms of a_i's and b_j's ($i, j \in \{0, 1\}$), i. e. the first three digits of q can be worked out without any recursive calls made to mult.

As avg needs at most two digits from each of its arguments before generating at least one digit of the output, the above argument demonstrates that mult a b produces at least one digit of the output before making any recursive calls to mult again. This leads to:

Lemma 4.4.39 (mult is total). *The result of applying* mult *to representations of* total *real numbers is the representation of a* total *real number, i. e.*

$$\forall x, y \in \mathcal{D}^\omega : \mathsf{mult}\ x\ y \ \in \mathcal{D}^\omega$$

The correctness of mult can be proved following the same line as that of avg. Therefore, we omit the details and merely present the following:

Definition 4.4.40 ($\otimes\colon \mathcal{J} \to \mathcal{J} \to \mathcal{J}$).

$$\forall [x_1, y_1], [x_2, y_2] \in \mathcal{J}\colon\ [x_1, y_1] \otimes [x_2, y_2] := [\min \mathrm{EP}, \max \mathrm{EP}]$$

where:

$$\mathrm{EP} = \{x_1x_2, x_1y_2, y_1x_2, y_1y_2\}$$

Lemma 4.4.41.

$$\forall a, b \in \mathcal{D}^\infty\colon\ to_{\mathcal{J}}\ (\ \mathsf{mult}\ a\ b\ ) \sqsubseteq_{\mathcal{J}} (to_{\mathcal{J}}\ a) \otimes (to_{\mathcal{J}}\ b)$$

Combining lemmata 4.4.39 and 4.4.41 one gets:

Corollary 4.4.42 (correctness of mult). *For elements $a, b \in \mathcal{D}^\omega$, i. e. infinite sequences of digits:*

$$to_{\mathcal{J}}\ (\ \mathsf{mult}\ a\ b\ ) = (to_{\mathcal{J}}\ a) \otimes (to_{\mathcal{J}}\ b)$$

Corollary 4.4.43 (extensionality of mult). mult *preserves the equivalence relation $\simeq_{\mathcal{D}^\infty}$ of Definition 4.4.1 on page 113, i. e. for all $d_x, e_x, d_y, e_y \in \mathcal{D}^\infty$:*

$$\left.\begin{array}{l} d_x \simeq_{\mathcal{D}^\infty} e_x \\ d_y \simeq_{\mathcal{D}^\infty} e_y \end{array}\right\} \;\Rightarrow\; (\ \mathsf{mult}\ d_x\ d_y\ ) \simeq_{\mathcal{D}^\infty} (\ \mathsf{mult}\ e_x\ e_y\ )$$

Remark 4.4.44. *We have chosen the implementation of multiplication worked out by Plume [1998] while there are other alternatives available as well. In fact much earlier, Trivedi and Ercegovac [1977] presented implementations of multiplication and division in a slightly more general setting. However, it is interesting to note that their division algorithm fails for* signed-digit binary *format, though once the base is raised to anything above 2, it works fine.*

4.4.6 $\boxed{\mathsf{abs}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty}$

Here we include a function to help us with the *absolute value* operation over real numbers. The reason we do this is merely to simplify Definition 4.4.64 on page 140 of the is_Str_Cau function which will be part of the implementation of the limit function (subsection 4.4.8). Of course, limit could be implemented without explicit resort to abs, in which case — as we will see in section 4.7 — abs is definable from other constants of SHRAD. In that respect, abs is not an essential part of SHRAD, and we are not going to dwell on it too much. Needless to say that absolute value operation is a much-used popular operation over real numbers — one prominent use of which is to get the distance between pairs of reals — thus we saw no harm in including it as a basic constant of the language.

We define the denotation of the constant *abs* — i. e. $\mathcal{A}_{\mathcal{SH}}(abs)$ — as:

Definition 4.4.45 ($\mathsf{abs}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty$).

$$\begin{aligned}
\mathsf{abs}\,(R\colon x) &= R\colon x \\
\mathsf{abs}\,(M\colon x) &= M\colon (\mathsf{abs}\,(x)) \\
\mathsf{abs}\,(L\colon x) &= R\colon (\mathsf{neg}(x))
\end{aligned}$$

where $\mathsf{neg}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty$ *is the negation function defined in Definition 4.4.26 on page 123.*

Lemma 4.4.46 (abs is total).

$$\forall x \in \mathcal{D}^\omega\colon \mathsf{abs}\,(x) \in \mathcal{D}^\omega$$

Proof. Left to the reader. □

Lemma 4.4.47 (correctness of abs). *For each $d \in \mathcal{D}^\infty$, let $[a_d.b_d] = to_{\mathcal{J}}(d)$. Then*

$$to_{\mathcal{J}}(\mathsf{abs}\,(d)) \sqsubseteq [r, s]$$

where

$$[r,s] = \begin{cases} [a_d, b_d] & \textit{if} \quad 0 \le a_d \\ [-b_d, -a_d] & \textit{if} \quad b_d \le 0 \\ [0, b_d] & \textit{if} \quad a_d \le 0 \le b_d \end{cases}$$

Proof. Left to the reader. □

Corollary 4.4.48 (extensionality of abs).

$$\forall x, y \in \mathcal{D}^\infty\colon \quad x \simeq_{\mathcal{D}^\infty} y \Rightarrow \mathsf{abs}\,(x) \simeq_{\mathcal{D}^\infty} \mathsf{abs}\,(y)$$

4.4.7 $\boxed{\mathsf{isNeg}\colon \mathcal{D}^\infty \to \mathbb{B}_\perp}$

Let us present the denotation of the only constant which is the link *from* continuous types *to* discrete ones. The major role this predicate is going to play for us — much like the function abs — is in simplifying Definition 4.4.64 on page 140 of is_Str_Cau which is in turn part of the implementation of limit (subsection 4.4.8).

This time however, *isNeg* is an essential constant of SHRAD, as it would not be possible to define isNeg from other constants. Moreover, although in our framework there is no possibility for defining *total* functions over real numbers by cases — as opposed to using pif_I in Real-PCF — it is still useful to have such a constant as part of the language. After all, now and again the user would like to compare the result of his calculations with some other value. Thus for instance, to see if x is less than some value y, one can run the test

$$\mathsf{isNeg}\,((x-y))$$

Therefore, we include this constant in our language.

Before coming to the denotation of *isNeg*, i. e. isNeg, we define two simple predicates. The first one checks to see if all the digits of its input are L, in which case it runs forever. Otherwise it terminates and returns *ff*:

Definition 4.4.49 (is_all_L : $\mathcal{D}^\infty \to \mathbb{B}_\perp$).

$$\begin{aligned}
\mathsf{is_all_L}\,(\mathsf{L}:\,x) &= \mathsf{is_all_L}\,(x)\\
\mathsf{is_all_L}\,(\mathsf{M}:\,x) &= \mathit{ff}\\
\mathsf{is_all_L}\,(\mathsf{R}:\,x) &= \mathit{ff}
\end{aligned}$$

The second predicate is some sort of dual to the previous one, in that it runs forever if all the digits are R, otherwise terminates with value *tt*:

Definition 4.4.50 (not_all_R : $\mathcal{D}^\infty \to \mathbb{B}_\perp$).

$$\begin{aligned}
\mathsf{not_all_R}\,(\mathsf{L}:\,x) &= \mathit{tt}\\
\mathsf{not_all_R}\,(\mathsf{M}:\,x) &= \mathit{tt}\\
\mathsf{not_all_R}\,(\mathsf{R}:\,x) &= \mathsf{not_all_R}\,(x)
\end{aligned}$$

Having defined these two predicates, we present isNeg as follows:

Definition 4.4.51 (isNeg : $\mathcal{D}^\infty \to \mathbb{B}_\perp$).

$$\begin{aligned}
\mathsf{isNeg}\,(\mathsf{L}:\,x) &= \mathsf{not_all_R}\,(x)\\
\mathsf{isNeg}\,(\mathsf{M}:\,x) &= \mathsf{isNeg}\,(x)\\
\mathsf{isNeg}\,(\mathsf{R}:\,x) &= \mathsf{is_all_L}\,(x)
\end{aligned}$$

Unlike previous functions, isNeg *is not a total one*, and the sources of partiality are representations of 0. Thus, by abusing notation, we can formulate the following lemma:

Lemma 4.4.52 (isNeg is total over $[-1,1] \setminus \{0\}$).

$$\forall x \in \mathcal{D}^{\omega}: \quad \mathsf{isNeg}\,(x) \in \{tt, \mathit{ff}\} \iff to_{\mathcal{J}}(x) \neq 0$$

Proof. Left to the reader. □

Lemma 4.4.53 (correctness of isNeg). *For any $d \in \mathcal{D}^{\infty}$, let $[a_d, b_d] = to_{\mathcal{J}}(d)$. Thus:*

$$\mathsf{isNeg}\,(d) = \begin{cases} tt & \text{if} \quad b_d < 0 \\ \mathit{ff} & \text{if} \quad 0 < a_d \\ \bot & \text{if} \quad a_d \leq 0 \leq b_d \end{cases}$$

Proof. Left to the reader. □

Corollary 4.4.54 (extensionality of isNeg).

$$\forall x, y \in \mathcal{D}^{\infty}: \quad x \simeq_{\mathcal{D}^{\infty}} y \Rightarrow \mathsf{isNeg}\,(x) = \mathsf{isNeg}\,(y)$$

4.4.8 $\boxed{\mathsf{limit}\colon [\mathbb{N}_{\bot} \to \mathcal{D}^{\infty}] \to \mathcal{D}^{\infty}}$

In the realm of (classical) mathematical analysis, the class of all sequences over (say) $\mathbb{R}$ is partitioned into two subclasses: *divergent* and *convergent*, with the latter considered to be the relatively more tame and easy-to-handle one. Enter *computable* (or *intuitionistic*) analysis, and yet even this one appears to be too wild to handle.

Assume that we are to implement a general limit operator:

$$\mathsf{gen\_limit}\colon \mathbb{R}^{\mathbb{N}} \rightharpoonup \mathbb{R}$$

such that given *any* convergent sequence $s\colon \mathbb{N} \to \mathbb{R}$ we get:

$$\mathsf{gen\_limit}\ s = \lim_{n\to\infty} s(n)$$

For each $k \in \mathbb{N}$, let $u_k\colon \mathbb{N} \to \mathbb{R}$ be the sequence defined by:

$$\forall i \in \mathbb{N}\colon u_k(i) := \begin{cases} 1 & \text{if } i \leq k \\ 0 & \text{if } i > k \end{cases}$$

and define the sequence $u\colon \mathbb{N} \to \mathbb{R}$ by:

$$\forall i \in \mathbb{N}\colon u(i) := 1$$

Obviously, u together with all u_k's are convergent with:

$$\forall k \in \mathbb{N}\colon \lim_{n\to\infty} u_k(n) = 0$$

and

$$\lim_{n\to\infty} u(n) = 1$$

Hence, for gen_limit to work correctly, it has to output correct results over these sequences. If one fixes an approximation range, say

$$\epsilon = 1/4$$

and applies gen_limit over u, it must be possible to get an interval a_u of length not more than ϵ, which contains $\lim_{n\to\infty} u(n)$, i. e.

$$1 \in a_u$$

and the whole process must end in some finite time.

Assume that we are in the middle of the computation of (gen_limit u). At any stage, say after n steps, at most n terms of u are evaluated by gen_limit. Let k_0 be the greatest subscript of these terms. If no such term exists[6] then let $k_0 = 0$. The problem gen_limit faces at this stage is that there is no way in which it can be assured of the sequence it is dealing with — i. e. u — not being (say) u_{k_0}, whose immediate next term is 0, and whose limit is also 0. Thus, gen_limit is unable to output the expected interval a_u.

Remark 4.4.55. *The tacit assumption that the flow of input/output information in time is* incremental *plays a crucial role in the previous argument. This means that once an interval is produced as partial result, not only the final result is inside it, but also all further refinements of the result occur there.*

The way out of this is, of course, to narrow the class of *all* convergent sequences to some smaller manageable subclass. There are different choices at hand, but the relevant ones all prove to be *equivalent*, in the sense that the model of the real numbers $\mathbb{R}$ and the function space $C(\mathbb{R})$ arising from each choice is the same.

Here we pick the set Cau_{2^n} of all the Cauchy sequences $s\colon \mathbb{N} \to \mathbb{R}$ such that:

$$\forall m, n \in \mathbb{N}\colon n \leq m \Rightarrow |s(n) - s(m)| < 2^{-n}$$

and as up to now we have restricted the discussion to real numbers in $[-1, 1]$, the following definition makes the statement of some of the results more concise:

[6] i. e. gen_limit has not evaluated any of the terms of u so far...

Definition 4.4.56 (Strong Cauchy Sequences: $\mathbf{Cau}_{2^n}(C)$). *For any subset C of the real numbers $\mathbb{R}$ we let $\mathrm{Cau}_{2^n}(C)$ be the set of all Cauchy sequences $(r_n)_{n\in\mathbb{N}}$ such that:*

1. $\forall n \in \mathbb{N}\colon r_n \in C$
2. $\forall m, n \in \mathbb{N}\colon n \leq m \Rightarrow |r_n - r_m| < 2^{-n}$

In order to demonstrate that $\mathrm{Cau}_{2^n}([-1,1])$ is in fact the right choice, we embark on presenting the implementation of the function

$$\mathsf{limit}\colon [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathcal{D}^\infty$$

where given any $(s_n)_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}([-1,1])$ and any $d_s\colon \mathbb{N}_\perp \to \mathcal{D}^\infty$ representing it, i. e.

$$\forall i \in \mathbb{N}\colon to_{\mathcal{J}}(d_s(i)) = s_i$$

we get:

$$to_{\mathcal{J}}(\mathsf{limit}\ d_s) = \lim_{n\to\infty} s_n$$

As usual, we need a few definitions first:

Definition 4.4.57 ($\widetilde{\ }\colon \{\mathsf{L}, \mathsf{M}, \mathsf{R}\} \to \mathcal{J}$). *For each digit $d \in \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$ we define:*

$$\widetilde{d} = to_{\mathcal{J}}([d])$$

Thus, we have:

$$\widetilde{\mathsf{L}} = [-1, 0], \qquad \widetilde{\mathsf{M}} = [-1/2, 1/2], \qquad \widetilde{\mathsf{R}} = [0, 1]$$

Definition 4.4.58 ($x_{<n}$). *Let X be a set and $x \in X^\infty$ a (finite or infinite) sequence over it. For each natural number $n \in \mathbb{N}$, by $x_{<n}$ we mean the initial segment of x of length not more than n. In particular:*

$$length(x) \leq n \iff x_{<n} = x$$

Definition 4.4.59 ($stretch_{\mathcal{J}}\colon \mathcal{J} \to \mathbb{R} \to \mathcal{J}$). *For any interval $a = [\underline{a}, \overline{a}] \in \mathcal{J}$ and any real number $r \in \mathbb{R}$ we define:*

$$stretch_{\mathcal{J}}\ a\ r \ := \begin{cases} [\max\{-1, \underline{a} - r\}, \min\{\overline{a} + r, 1\}] & \text{if } 0 \leq r \\ a & \text{otherwise} \end{cases}$$

Proposition 4.4.60. *For any $d \in \mathcal{D}^\infty$:*

1. $\mu(to_{\mathcal{J}}(d)) > 0 \iff d \in \mathcal{D}^* \iff \mathit{length}(d) \in \mathbb{N}$, *in which case:*

$$\mu(to_{\mathcal{J}}(d)) = 2^{1-\mathit{length}(d)}$$

   where $\mu(a)$ is the length of the interval a, as in Definition 3.3.7 on page 76.

2. $\mu(to_{\mathcal{J}}\, d) = 0 \iff d \in \mathcal{D}^\omega \iff \mathit{length}(d) \notin \mathbb{N}$.

Therefore by allowing $2^{-\infty} = 0$, we can write:

$$\forall d \in \mathcal{D}^\infty : \mu(to_{\mathcal{J}}(d)) = 2^{1-\mathit{length}(d)}$$

Proof. Left to the reader. □

Consider any interval $a = [\underline{a}, \overline{a}] \in \mathcal{J}$ with $\mu(a) \leq 1/2$. One can easily argue that:

1. $0 \leq \underline{a} \Rightarrow [0, 1] \sqsubseteq_{\mathcal{J}} a$
2. $\overline{a} \leq 0 \Rightarrow [-1, 0] \sqsubseteq_{\mathcal{J}} a$
3. otherwise $[-1/2, 1/2] \sqsubseteq_{\mathcal{J}} a$

In other words:

$$\forall a \in \mathcal{J} : \mu(a) \leq 1/2 \Rightarrow \exists X \in \{\mathsf{L}, \mathsf{M}, \mathsf{R}\} : to_{\mathcal{J}}\,([X]) \sqsubseteq_{\mathcal{J}} a$$

A generalization of this fact (to be expressed in the following proposition) is the essence of generating successive digits of the limit. First, we fix a notation:

Notation 4.4.61. *Let X be a set, $w \in X^*$ and $v \in X^\infty$ two sequences over X. By*

$$w +\!\!\!+\, v$$

we mean the concatenation *of w and v.*

Proposition 4.4.62. *If $a \in \mathcal{J}$ and $d \in \mathcal{D}^*$ satisfy:*

1. $(to_{\mathcal{J}}\, d) \sqsubseteq_{\mathcal{J}} a$
2. $\mu(a) \leq 2^{-\mathit{length}(d)-1}$

then:

$$\exists X \in \{\mathsf{L}, \mathsf{M}, \mathsf{R}\} \colon to_{\mathcal{J}}\,(d +\!\!+ [X]) \sqsubseteq_{\mathcal{J}} a$$

Proof. By induction over the length of d:

base case $d = [\ ]$: In this case $\mu(a) \leq 2^{-1}$ and we have already discussed this case in the argument preceding this proposition.

inductive case : Suppose $d = d_0 \colon d'$. We first observe that as $to_{\mathcal{J}}(d) \sqsubseteq a$ we get:

$$a = \mathsf{Cons}_{\widetilde{d_0}}\ \mathsf{Tail}_{\widetilde{d_0}}\ a \tag{4.3}$$

Now we let $a' = \mathsf{Tail}_{\widetilde{d_0}}\ a$ and we see that:

$$\begin{array}{rcl} to_{\mathcal{J}}(d') & = & to_{\mathcal{J}}({d_0}^{-1} d) \\ \text{(by Lemma 4.4.18 on page 119)} & \sqsubseteq & \mathsf{Tail}_{\widetilde{d_0}}(to_{\mathcal{J}}(d)) \\ (\mathsf{Tail}_{\widetilde{d_0}} \text{ is monotone}) & \sqsubseteq & \mathsf{Tail}_{\widetilde{d_0}}\ a \\ & = & a' \end{array}$$

On the other hand:

$$\begin{array}{rcl} \mu(a') & = & \mu(\mathsf{Tail}_{\widetilde{d_0}}\ a) \\ & \leq & 2 \times \mu(a) \\ & \leq & 2 \times 2^{-length(d)-1} \\ & = & 2^{-length(d)} \\ & = & 2^{-(length(d')+1)} \\ & = & 2^{-length(d')-1} \end{array}$$

Now we are in the right position to apply the induction hypothesis to d' and a' in order to get the digit X which makes:

$$to_{\mathcal{J}}(d' +\!\!+ [X]) \sqsubseteq a'$$

Finally we see that:

$$\begin{array}{rcl} to_{\mathcal{J}}(d +\!\!+ [X]) & = & to_{\mathcal{J}}((d_0 \colon d') +\!\!+ [X]) \\ & = & to_{\mathcal{J}}(d_0 \colon (d' +\!\!+ [X])) \\ (\text{definition of } to_{\mathcal{J}}) & = & \mathsf{Cons}_{\widetilde{d_0}}(to_{\mathcal{J}}(d' +\!\!+ [X])) \\ (\mathsf{Cons}_{\widetilde{d_0}} \text{ is monotone}) & \sqsubseteq & \mathsf{Cons}_{\widetilde{d_0}}(a') \\ (\text{definition of } a') & = & \mathsf{Cons}_{\widetilde{d_0}}.\mathsf{Tail}_{\widetilde{d_0}}(a) \\ (\text{Equation (4.3) above}) & = & a \end{array}$$

□

Take any $(r_n)_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}([-1,1])$ with $r = \lim_{n\to\infty} r_n$. As $[-1,1]$ is a closed subset of $\mathbb{R}$, we get $r \in [-1,1]$. Let $f : \mathbb{N}_\perp \to \mathcal{D}^\omega$ be *any* representation of $(r_n)_{n\in\mathbb{N}}$. We want to outline the process which leads to the computation of some $f_r \in \mathcal{D}^\omega$ representing the real number $r \in [-1,1]$.

To produce the first digit of f_r, by Proposition 4.4.62 on page 136, we should look for an interval $a \in \mathcal{J}$ such that:

1. $r \in a$
2. $\mu(a) \leq 1/2$

and of course, all the information we have is the sequence f and the fact that $(r_n)_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}([-1,1])$. Yet, this information is enough to show that:

1. $|r - r_3| \leq 1/8$
2. $r_3 \in to_{\mathcal{J}}\,(f(3)_{<3})$
3. $\mu(to_{\mathcal{J}}\,(f(3)_{<3})) = 2^{1-3} = 1/4$

which leads to:

$$r \in stretch_{\mathcal{J}}\ \ (to_{\mathcal{J}}\,(f(3)_{<3}))\ \ 1/8$$

So, by considering $a_1 = stretch_{\mathcal{J}}\ \ (to_{\mathcal{J}}\,(f(3)_{<3}))\ \ 1/8$ we get the interval we were looking for, as

$$(r \in a_1) \wedge (\mu(a_1) \leq (1/4 + 1/8 + 1/8) = 1/2)$$

This way, we can produce the first digit of f_r. The way to proceed follows the same basic method implemented recursively. So assume that at some stage, we have produced the first n digits of f_r, which we write as:

$$(f_r)_{<n} = d_0 \ldots d_{n-1}$$

To get the next digit d_n, all we need to do is to consider the interval $a_n \in \mathcal{J}$ defined by:

$$a_n := b_n \cap (to_{\mathcal{J}}((f_r)_{<n})) \tag{4.4}$$

where

$$b_n = stretch_{\mathcal{J}}\ \ (to_{\mathcal{J}}\,(f(n+3)_{<n+3}))\ \ 2^{-(n+3)}$$

as we have:

1. $r \in to_{\mathcal{J}}((f_r)_{<n})$
2. $|r - r_{n+3}| \leq 2^{-(n+3)}$
3. $r_{n+3} \in to_{\mathcal{J}}(f(n+3)_{<n+3})$
4. $\mu(to_{\mathcal{J}}(f(n+3)_{<n+3})) = 2^{1-(n+3)} = 2^{-(n+2)}$

therefore by 1, 2 and 3:

$$r \in a_n$$

and by 4:

$$\begin{array}{rcl} \mu(a_n) & \leq & \mu(b_n) \\ & = & 2^{-(n+2)} + 2 \times 2^{-(n+3)} \\ & < & 2^{-(n+1)} \\ & = & 2^{-length((f_r)_{<n})-1} \end{array}$$

Hence, by Proposition 4.4.62 on page 136 we are able to produce the next digit $d_n \in \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$ such that:

$$to_{\mathcal{J}}((f_r)_{<n+1}) = to_{\mathcal{J}}(d_0 \ldots d_{n-1} d_n) \sqsubseteq_{\mathcal{J}} a_n$$

Combined with the fact that $r \in a_n$ we get:

$$r \in to_{\mathcal{J}}((f_r)_{<n+1})$$

Thus we have gone through the algorithm which we are about to apply in defining the function $\mathsf{limit}\colon [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathcal{D}^\infty$ promised before. The representations of strong Cauchy sequences are just a fraction of the whole objects in the domain $[\mathbb{N}_\perp \to \mathcal{D}^\infty]$. Hence in implementing limit, alongside generating output digits, we run a *test* over successive terms of the input in order to ensure that the sequence is legitimate. The unfortunate fact is that this test is quite expensive; nonetheless, without it we lose one of the main characteristics of our framework, i. e. *extensionality*. In Example 4.6.7 on page 152 we will illustrate why this expensive test is inevitable and there we discuss the whole issue further.

According to the algorithm above, for each $n \geq 3$, the $(n-2)$-nd digit of the limit — i. e. d_{n-3} — is generated based on the n-th term of the sequence — i. e. f_n. Now for each $n > 3$, we first check

$$\begin{array}{rcl} |\, r_3 - r_n \,| & < & 2^{-3} \\ \vdots & & \vdots \\ |\, r_{n-1} - r_n \,| & < & 2^{-(n-1)} \end{array}$$

and embark on generating the digit d_{n-3}, only if all these $n-3$ inequalities hold[7]. For this we define the predicate:

$$\mathsf{is\_Str\_Cau} \colon [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathbb{N}_\perp \to \mathbb{B}_\perp$$

such that given any function $f \colon \mathbb{N}_\perp \to \mathcal{D}^\infty$ representing the sequence $(r_n)_{n\in\mathbb{N}}$, and any $n > 3$:

$$\mathsf{is\_Str\_Cau}\,(f)\,(n) = \ tt \iff \forall k \in \{3, \ldots, n-1\} \colon |r_k - r_n| < 2^{-k}$$

which in turn uses the predicate

$$\mathsf{cau\_Test}\,(x, y, k) = |x - y| < 2^{-k}$$

the implementation of which necessitated the introduction of the constant *abs* in the language:

Definition 4.4.63 ($\mathsf{cau\_Test} \colon (\mathcal{D}^\infty, \mathcal{D}^\infty, \mathbb{N}_\perp) \to \mathbb{B}_\perp$).
For all $d_1, d_2 \in \mathcal{D}^\infty$ and $k \in \mathbb{N}_\perp$:

$$\mathsf{cau\_Test}\,(d_1, d_2, k) = \mathsf{isNeg}\,(\mathsf{sub}\,(\mathsf{abs}\,(\mathsf{sub}\,(d_1)\,(d_2)))\,(\mathsf{qtoR}\,(1, 1, 2^k)))$$

where
$$\begin{cases} \mathsf{sub} \colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty \\ \mathsf{sub}\,(x)\,(y) = \mathsf{M}^{-1}\,(\mathsf{avg}\ x\,(\mathsf{mult}\ d_{-1}\ y)) \end{cases}$$
and $to_{\mathcal{I}}(d_{-1}) = -1$.

Definition 4.4.64 ($\mathsf{is\_Str\_Cau} \colon [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathbb{N}_\perp \to \mathbb{B}_\perp$).

$$\mathsf{is\_Str\_Cau}\,(f)\,(n) = \begin{cases} tt & \text{if } \ n \leq 3 \\ \mathsf{aux\_Is\_Str\_Cau}\,(f)\,(3)\,(n) & \text{if } \ n > 3 \end{cases}$$

where $\mathsf{aux\_Is\_Str\_Cau}\,(f)\,(k)\,(n)$ *is:*

$$\begin{cases} tt & \text{if} \quad k = n \\ \mathsf{aux\_Is\_Str\_Cau}\,(f)\,(k+1)\,(n) & \text{if} \quad \mathsf{cau\_Test}\,(f(k), f(n), k) = tt \\ ff & \text{if} \quad \mathsf{cau\_Test}\,(f(k), f(n), k) = ff \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

[7]It is worth emphasizing here that *inequality* over real numbers is semi-decidable, as opposed to *equality*.

Going back to the implementation of limit, we first define a function $\mathsf{bcna}\colon \mathcal{D}^\infty \to \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$ such that given any $d \in \mathcal{D}^\infty$ for which a digit $X \in \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$ exists satisfying:

$$to_{\mathcal{J}}(X) \sqsubseteq stretch_{\mathcal{J}}\ to_{\mathcal{J}}(d)\ 1/8$$

we get

$$\mathsf{bcna}(d) = X$$

One tentative definition could be:

$$\begin{aligned} &\mathsf{bcna}\colon \mathcal{D}^\infty \to \{\mathsf{L}, \mathsf{M}, \mathsf{R}\} \\ &\mathsf{bcna}\ d = \begin{cases} \mathsf{R} & \text{if} \quad \underline{a} \geq 0 \\ \mathsf{L} & \text{if} \quad \overline{a} \leq 0 \\ \mathsf{M} & \text{if} \quad (-1/2 \leq \underline{a}) \wedge (\overline{a} \leq 1/2) \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

where $[\underline{a}, \overline{a}] = to_{\mathcal{J}}(stretch_{\mathcal{J}}\ (d_{<3})\ 1/8))$.

Although the definition works fine, we would rather not resort to rational numbers via $to_{\mathcal{J}}$ and $stretch_{\mathcal{J}}$. Instead we pretend to be fussy and present a purely symbolic definition for bcna using pattern matching, although it requires a long list of cases to be checked:

Definition 4.4.65 ($\mathsf{bcna}\colon \mathcal{D}^\infty \to \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$). *See Table 4.3 on the next page.*

Next we define the function $\mathsf{cna}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$ such that given $\mathsf{inner} \in \mathcal{D}^\infty$ and $\mathsf{outer} := d_0\ d_1\ \ldots\ d_n \in \mathcal{D}^*$ with:

1. $length(\mathsf{outer}) + 3 \leq length(\mathsf{inner})$
2. $to_{\mathcal{J}}(\mathsf{outer}) \sqsubseteq to_{\mathcal{J}}(\mathsf{inner})$

we get:

$$to_{\mathcal{J}}(\mathsf{outer} +\!\!+ [X]) \sqsubseteq stretch_{\mathcal{J}}\ (to_{\mathcal{J}}(\mathsf{inner}))\ 2^{-(n+3)}$$

where:

1. $X = \mathsf{cna\ inner\ outer}$
2. $n = length(\mathsf{outer})$

Of course for practical purposes, we have to relax the condition:

$$to_{\mathcal{J}}(\mathsf{outer}) \sqsubseteq to_{\mathcal{J}}(\mathsf{inner})$$

Table 4.3 The function bcna : $\mathcal{D}^\infty \to \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$

bcna	(L: L: x)	=	L
bcna	(L: M: x)	=	L
bcna	(L: R: L: x)	=	L
bcna	(L: R: M: x)	=	M
bcna	(L: R: R: x)	=	M
bcna	(M: L: L: x)	=	L
bcna	(M: L: M: x)	=	M
bcna	(M: L: R: x)	=	M
bcna	(M: M: x)	=	M
bcna	(M: R: L: x)	=	M
bcna	(M: R: M: x)	=	R
bcna	(M: R: R: x)	=	R
bcna	(R: L: L: x)	=	M
bcna	(R: L: M: x)	=	R
bcna	(R: L: R: x)	=	R
bcna	(R: M: x)	=	R
bcna	(R: R: x)	=	R

and only stick to the first condition:

$$length(\mathsf{outer}) + 3 \leq length(\mathsf{inner})$$

The reason is best observed by a second look at the definition of a_n (Equation (4.4) on page 138) where parts of the interval:

$$b_n = stretch_{\mathcal{J}}\ \ (to_{\mathcal{J}}\,(f(n+3)_{<n+3}))\ \ 2^{-(n+3)}$$

might well lie outside the (nonetheless *overlapping*) interval:

$$to_{\mathcal{J}}((f_r)_{<n})$$

However, we are only interested in the overlapping segment containing the limit, i. e. r.

Definition 4.4.66 ($\mathsf{cna}\colon \mathcal{D}^\infty \to \mathcal{D}^\infty \to \{\mathsf{L}, \mathsf{M}, \mathsf{R}\}$).

$$\mathsf{cna\ inner\ outer} = \begin{cases} \mathsf{bcna\ inner} & \textit{if} \quad \textit{length}(\mathsf{outer}) = 0 \\ \textit{undefined} & \textit{if} \quad \textit{length}(\mathsf{inner}) = 0 \\ \mathsf{cna}\ (d_0^{-1}(\mathsf{inner}))\ \mathsf{outer}' & \textit{otherwise} \end{cases}$$

where $\mathsf{outer} = d_0 : \mathsf{outer}'$.

Now we can present our main function:

Definition 4.4.67 (limit).

$$\begin{cases} \mathsf{limit}\colon [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathcal{D}^\infty \\ \mathsf{limit}\ f = \mathsf{aux\_limit}\ f\ 3\ [\,] \end{cases}$$

where the auxiliary function $\mathsf{aux\_limit}$ *is defined by:*

$$\mathsf{aux\_limit}\colon [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathbb{N}_\perp \to \mathcal{D}^\infty \to \mathcal{D}^\infty$$

$$\mathsf{aux\_limit}\ f\ n\ a = \begin{cases} d : (\mathsf{aux\_limit}\ f\ (n+1)\ (a \mathbin{+\!\!+} [d])) & \textit{if} \quad \mathsf{is\_Str\_Cau}\,(f)\,(n) \\ [\,] & \textit{otherwise} \end{cases}$$
$$\textit{where} \qquad d = \mathsf{cna}\ (f(n)_{<n})\ a$$

We have already gone through the algorithm for limit, implicit in which is the proofs for the following lemmas:

Lemma 4.4.68 (limit is total over strong Cauchy sequences).
For any $f : \mathbb{N}_\perp \to \mathcal{D}^\infty$*:*

$$\boxed{\mathsf{limit}\ f \in \mathcal{D}^\omega \iff f \textit{ represents some strong Cauchy sequence}}$$

Lemma 4.4.69. *Let* $f : \mathbb{N}_\perp \to \mathcal{D}^\omega$ *represent the strong Cauchy sequence*

$$(r_n)_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}([-1, 1])$$

with $r = \lim_{n\to\infty} r_n \in [-1, 1]$*. Then:*

$$\forall n \in \mathbb{N} : r \in to_{\mathcal{I}}((\mathsf{limit}\ f)_{<n})$$

Combining lemmata 4.4.68 and 4.4.69 we get:

Corollary 4.4.70 (correctness of limit). *Let* $f\colon \mathbb{N}_\perp \to \mathcal{D}^\omega$ *represent the strong Cauchy sequence* $(r_n)_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}([-1,1])$*. Then* $(\mathsf{limit}\ f)$ *represents the total real number* $\lim_{n\to\infty} r_n \in [-1,1]$*, i. e.*

$$(\forall i \in \mathbb{N}\colon\, to_{\mathcal{J}}(f(i)) = r_i) \Rightarrow (to_{\mathcal{J}}(\mathsf{limit}\ f) = \lim_{n\to\infty} r_n)$$

Extensionality of limit over strong Cauchy sequences is also guaranteed by Corollary 4.4.70:

Corollary 4.4.71 (extensionality of limit over strong Cauchy sequences). *Let* $f, g\colon \mathbb{N} \to \mathcal{D}^\omega$ *be representations of the same strong Cauchy sequence*

$$(r_n)_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}([-1,1])$$

Then:

$$\mathsf{limit}\ f \simeq_{\mathcal{D}^\infty} \mathsf{limit}\ g$$

4.5 Operational Semantics

We continue with our style of defining operational semantics via *immediate* reduction rules. The PCF fragment of SHRAD follows that of standard PCF as in Table 2.2 on page 39. As for the proper SHRAD terms, one needs to know how to recover digits by reduction. But this can already be extracted from the denotational semantics in a straightforward way. Take the constant $avg : r \to r \to r$ for instance. We have:

$$[\![avg]\!] = \mathsf{avg}$$

From Table 4.2 on page 125 we learn:

$$\mathsf{avg}\ (\mathsf{L}\colon\ x)\ (\mathsf{L}\colon\ y) = \mathsf{L}\colon\ (\mathsf{avg}\ x\ y)$$

Therefore one can postulate:

$$avg\ (L\colon\ x)\ (L\colon\ y) \to L\colon\ (avg\ x\ y)$$

Other constants follow suit, though at times not as easy as that of *avg*. For instance we already know what a lengthy path one has to take in order to generate one digit out of a limit computation. Nonetheless, the principle remains the same and we skip the details.

4.6 Extensionality

By now, not only we have given the precise definition of what it means for a function over real numbers to be represented by a Shrad-definable object via $to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}}$ of Definition 4.4.3 on page 113, but also we have a good intuition of the concept as a result of section 4.4. Here as we will be dealing with the subject exclusively, perhaps a more handy terminology and notation is not a bad idea. Thus we define:

Definition 4.6.1.

1. *A partial function* $g\colon [-1,1]^m \rightharpoonup [-1,1]$ *is said to be* Shrad-definable *if it is representable by some* Shrad*-definable* $\tilde{g}\colon (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$*, i. e.*

$$to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}}(\tilde{g}) = g$$

2. *A function* $f\colon (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$ *is said to* represent *a function over real numbers if some* $\hat{f}\colon [-1,1]^m \rightharpoonup [-1,1]$ *exists for which*

$$to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}}(f) = \hat{f}$$

Take any Shrad-definable $f\colon (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$. We demonstrate that for any two $\boldsymbol{d}, \boldsymbol{e} \in (\mathcal{D}^\omega)^m$ representing the same $m-$tuple of real numbers, i. e.

$$\forall i \in \{1, \ldots, m\}\colon to_{\mathcal{J}}(d_i) = to_{\mathcal{J}}(e_i)$$

it is *impossible* for all the following to hold at the same time:

i. $f(\boldsymbol{d}) \in \mathcal{D}^\omega$

ii. $f(\boldsymbol{e}) \in \mathcal{D}^\omega$

iii. $to_{\mathcal{J}}(f(\boldsymbol{d})) \neq to_{\mathcal{J}}(f(\boldsymbol{d}))$

We simply refer to this property of the (functions definable in the) language as *extensionality*.

To prove extensionality for Shrad, we resort to logical relations (see subsection 2.4.3).

4.6.1 The Logical Relation $\mathcal{X}$

Definition 4.6.2 (Binary Logical Relation $\mathcal{X}$). *Let $\cup\{\mathbb{D}_\sigma \mid \sigma$ a* SHRAD*-type$\}$ be the collection of domains for* SHRAD *as in Definition 4.3.9 on page 110. The binary logical relation $\mathcal{X}$ is defined over the pairs of objects in this model by:*

1. *For $o \in \{bool, nat\}$:*

$$\forall x, y \in \mathbb{D}_o : x \; \mathcal{X}^o \; y \iff x \uparrow y$$

2. $\forall d, e \in \mathcal{D}^\infty : d \; \mathcal{X}^r \; e \iff to_{\mathcal{J}}(d) \uparrow to_{\mathcal{J}}(e)$

where $a \uparrow b$ stands for 'a and b are consistent' (Definition 2.2.3 on page 33). $\mathcal{X}$ is the logical relation built upon the above ground type cases.

Proposition 4.6.3. *Over the ground types $o \in \{bool, nat\}$, we can rephrase the logical relation as:*

1. $\forall x, y \in \mathbb{D}_o : \quad x \; \mathcal{X}^o \; y \iff (x \sqsubseteq y) \vee (y \sqsubseteq x)$

2. $\forall x, y \in \mathbb{D}_o : \quad x \; \mathcal{X}^o \; y \iff (x = y) \vee (x = \bot) \vee (y = \bot)$

Proof. Easy. □

Proposition 4.6.4. *Over the ground type r:*

$$\forall x, y \in \mathcal{D}^\infty : \quad x \; \mathcal{X}^r \; y \iff to_{\mathcal{J}}(x) \cap to_{\mathcal{J}}(y) \neq \emptyset$$

Proof. Easy. □

Thus we freely use any of the equivalent reformulations of $\mathcal{X}$ over ground types as expressed in the previous couple of propositions.

Next we demonstrate that $\mathcal{X}$ is a C-logical relation, where C is the set of SHRAD-constants:

1. For constants inherited from PCF it is straightforward to show that they preserve $\mathcal{X}$. Hence we skip the details.

2. For proper SHRAD-constants (except *limit* and Y_σ's) we have already established the property in section 4.4. For instance, by Lemma 4.4.32 on page 124, *avg* preserves consistency. The cases for other such constants are similarly dealt with, which we summarize in Table 4.4 on the next page.

3. The fix-point constants Y_σ's and *limit* are the ones that require further attention. Thus in the sequel we focus on these constants.

Table 4.4 $\mathcal{X}$ and SHRAD-constants

Constant	Corresponding Statement	Page
L, M, R	Proposition 4.4.5	115
L^{-1}, M^{-1}, R^{-1}	Lemma 4.4.18	119
$qtoR$	Lemma 4.4.23	123
avg	Lemma 4.4.32	124
$mult$	Lemma 4.4.41	130
abs	Lemma 4.4.47	131
$isNeg$	Lemma 4.4.53	133

The Fix-point Constants

In Chapter 3 we showed how fix-point constants preserved certain logical relations[8]. The technique is similar here and we do not get into the details anymore. Again the interesting bit is to demonstrate that Y_r is invariant under $\mathcal{X}$, which in turn boils down to:

Proposition 4.6.5. *The set of $\mathcal{X}^r$ invariant elements forms an* inclusive predicate[9] *in $\mathcal{D}^\infty \times \mathcal{D}^\infty$.*

Proof.

1. $$\begin{aligned}[-1,1] \cap [-1,1] \neq \emptyset &\Rightarrow to_{\mathcal{J}}([\,]) \uparrow to_{\mathcal{J}}([\,]) \\ &\Rightarrow \bot_{\mathcal{D}^\infty}\ \mathcal{X}^r\ \bot_{\mathcal{D}^\infty}\end{aligned}$$

2. Let $\{d_i \mid i \in \mathbb{N}\}$ and $\{e_j \mid j \in \mathbb{N}\}$ be two ascending chains in $\mathcal{D}^\infty$ such that:

   $$\forall i \in \mathbb{N}\colon d_i\ \mathcal{X}^r\ e_i$$

   Note that as $to_{\mathcal{J}}$ is monotone, $\{to_{\mathcal{J}}(d_i) \mid i \in \mathbb{N}\}$ and $\{to_{\mathcal{J}}(e_i) \mid i \in \mathbb{N}\}$ are two ascending chains in $\mathcal{J}$ as well. Fixing any $i \in \mathbb{N}$ we observe that:

   (a) as $\{to_{\mathcal{J}}(e_j) \mid j \in \mathbb{N}\}$ is ascending:

   $$\forall j \leq i\colon to_{\mathcal{J}}(e_j) \sqsubseteq to_{\mathcal{J}}(e_i)$$

   which combined with the fact that $d_i\ \mathcal{X}^r\ e_i$ results in:

   $$\forall j \leq i\colon d_i\ \mathcal{X}^r\ e_j$$

[8] e. g. Proposition3.2.6 on page 68.
[9] Definition 3.2.5 on page 67

(b) as $\{to_{\mathcal{J}}(d_j) \mid j \in \mathbb{N}\}$ is ascending:

$$\forall j \geq i:\ to_{\mathcal{J}}(d_i) \sqsubseteq to_{\mathcal{J}}(d_j)$$

which combined with the fact that $d_j\ \mathcal{X}^r\ e_j$ results in:

$$\forall j \geq i:\ d_i\ \mathcal{X}^r\ e_j$$

From (a) and (b) on pages 147–148 it follows that for each $i \in \mathbb{N}$

$$\{to_{\mathcal{J}}(d_i) \cap to_{\mathcal{J}}(e_j) \mid j \in \mathbb{N}\}$$

forms an ascending chain of non-empty compact intervals in $\mathcal{J}$, which has to have a non-empty intersection. Thus:

$$\begin{aligned} to_{\mathcal{J}}(d_i) \cap (\cap\{to_{\mathcal{J}}(e_j) \mid j \in \mathbb{N}\}) &= \cap\{to_{\mathcal{J}}(d_i) \cap to_{\mathcal{J}}(e_j) \mid j \in \mathbb{N}\} \\ &\neq \emptyset \end{aligned}$$

Now let us fix the non-empty interval

$$a = \cap\{to_{\mathcal{J}}(e_j) \mid j \in \mathbb{N}\}$$

By the preceding argument, for any $i \in N$

$$to_{\mathcal{J}}(d_i) \cap a$$

is non-empty. As a result $\{to_{\mathcal{J}}(d_i) \cap a \mid i \in \mathbb{N}\}$ forms an ascending chain of non-empty compact intervals in $\mathcal{J}$, which again has to have a non-empty intersection. Therefore

$$\begin{aligned} to_{\mathcal{J}}(\sqcup\{d_i \mid i \in \mathbb{N}\}) \cap to_{\mathcal{J}}(\sqcup\{e_j \mid j \in \mathbb{N}\}) &= (\sqcup\{to_{\mathcal{J}}(d_i) \mid i \in \mathbb{N}\}) \cap \\ &\qquad to_{\mathcal{J}}(\sqcup\{e_j \mid j \in \mathbb{N}\}) \\ (\text{as } a = to_{\mathcal{J}}(\sqcup\{e_j \mid j \in \mathbb{N}\})) &= \sqcup\{to_{\mathcal{J}}(d_i) \cap a \mid i \in \mathbb{N}\} \\ (\text{preceding argument}) &\neq \emptyset \end{aligned}$$

□

The Constant *limit*

Here we demonstrate how *limit* preserves the logical relation $\mathcal{X}$. The argument we follow is based on the algorithm presented in subsection 4.4.8 and we assume familiarity with that material throughout, without repeated cross-referencing anymore.

Let $f, g : \mathbb{N}_\perp \to \mathcal{D}^\infty$ satisfy:

$$f \; \mathcal{X}^{nat \to r} \; g$$

which in particular implies:

$$\forall n \in \mathbb{N}: \quad to_{\mathcal{I}}(f(n)) \cap to_{\mathcal{I}}(g(n)) \neq \emptyset$$

We must prove that $(\mathsf{limit}\, f)\ \mathcal{X}^r\ (\mathsf{limit}\, g)$. For this we prove the following statement:

$$\forall i, j \in \mathbb{N}: \quad to_{\mathcal{I}}((\mathsf{limit}\, f)_{<i}) \cap to_{\mathcal{I}}((\mathsf{limit}\, g)_{<j}) \neq \emptyset$$

which suffices to lead to:

$$to_{\mathcal{I}}(\mathsf{limit}\, f) \cap to_{\mathcal{I}}(\mathsf{limit}\, g) \neq \emptyset$$

Let us first fix a couple of notations and assume:

$$\begin{aligned} \mathsf{limit}\, f &= a_0 : a_1 : a_2 : \ldots \\ \mathsf{limit}\, g &= b_0 : b_1 : b_2 : \ldots \end{aligned}$$

Now in case either of $\mathsf{limit}\, f$ or $\mathsf{limit}\, g$ produces no digits, the result follows trivially. Therefore we assume that a_0 and b_0 have been output. We know that:

(i) $to_{\mathcal{I}}([a_0]) \sqsubseteq to_{\mathcal{I}}(f(3))$

(ii) $to_{\mathcal{I}}([b_0]) \sqsubseteq to_{\mathcal{I}}(g(3))$

(iii) $to_{\mathcal{I}}(f(3)) \cap to_{\mathcal{I}}(g(3)) \neq \emptyset$

If both $\mathsf{limit}\, f$ and $\mathsf{limit}\, g$ stop generating any further digits we have nothing more to talk about. On the other hand (without loss of generality) let $\mathsf{limit}\, g$ generate a further digit b_1 (Figure 4.1 on the following page). This necessitates the following condition to hold:

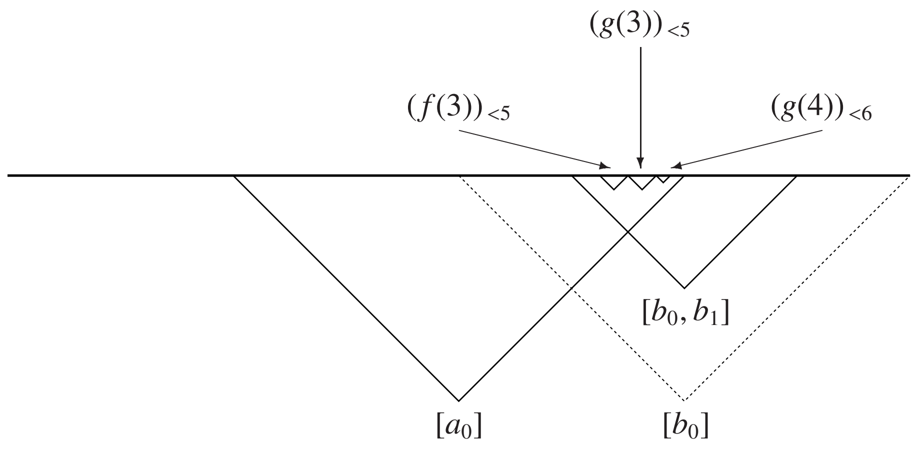
$$\mathsf{is\_Str\_Cau}\,(g)\,(4) = tt$$

which in turn implies:

$$\mu\left(to_{\mathcal{I}}(g(3)) \sqcap to_{\mathcal{I}}(g(4))\right) < \frac{1}{8} \tag{4.5}$$

Figure 4.1 Extensionality of the limit

Note: in order to save space in this figure by $d \in \mathcal{D}^\infty$ we mean $to_{\mathcal{J}}(d) \in \mathcal{J}$.



(μ is defined in Definition 3.3.7 on page 76).

Equation (4.5) on the page before simply says that $to_{\mathcal{J}}(g(3))$ and $to_{\mathcal{J}}(g(4))$ both lie inside an open interval of length $1/8$. This in particular requires $g(3)$ (and $g(4)$ alike) to be defined up to at least 5 terms.

Going back to cna[10] of Definition 4.4.66 on page 143 we know that a_0 is produced in such a way that the interval $to_{\mathcal{J}}([a_0])$ contains not only $to_{\mathcal{J}}(f(3))$, but also a $1/8$ neighbourhood of it too. This, combined with:

(a) $to_{\mathcal{J}}(f(3)) \cap to_{\mathcal{J}}(g(3)) \neq \emptyset$

(b) $\mu(to_{\mathcal{J}}(g(3)) \sqcap to_{\mathcal{J}}(g(4))) < 1/8$

implies:

$$to_{\mathcal{J}}([a_0]) \sqsubseteq to_{\mathcal{J}}(g(4)) \tag{4.6}$$

Moreover, once the test

$$\mathsf{is\_Str\_Cau}\,(g)\,(4) = tt$$

[10]in this particular case in fact bcna of Definition 4.4.65 on page 141.

is passed, b_1 is output in such a way that:

$$to_{\mathcal{J}}([b_0, b_1]) \sqsubseteq to_{\mathcal{J}}(g(4)) \tag{4.7}$$

Equations (4.6) and (4.7) on pages 150–151 imply

$$to_{\mathcal{J}}([a_0]) \cap to_{\mathcal{J}}([b_0, b_1]) \neq \emptyset$$

hence $(\mathsf{limit}\, f)_{<1}\ \mathcal{X}^r\ (\mathsf{limit}\, g)_{<2}$.

The preceding argument can be extended to the general case in a straightforward manner:

Proposition 4.6.6. *Let $f, g : \mathbb{N}_\perp \to \mathcal{D}^\infty$ satisfy $f\ \mathcal{X}^{nat\to r}\ g$. Then:*

$$\forall i, j \in \mathbb{N} : to_{\mathcal{J}}\left((\mathsf{limit}\, f)_{<i}\right) \cap to_{\mathcal{J}}\left((\mathsf{limit}\, g)_{<j}\right) \neq \emptyset$$

Proof. [induction] We have already established the case for $i + j = 3$. Now assume that for some $i, j \in \mathbb{N}$ we have:

$$to_{\mathcal{J}}\left((\mathsf{limit}\, f)_{<i}\right) \cap to_{\mathcal{J}}\left((\mathsf{limit}\, g)_{<j}\right) \neq \emptyset$$

If both $\mathsf{limit}\, f$ and $\mathsf{limit}\, g$ stop generating any further digits then we are done. Otherwise (without loss of generality) let $\mathsf{limit}\, g$ output the next digit b_j. This requires

$$\mathsf{is\_Str\_Cau}\,(g)\,(j+3) = tt$$

to hold. Let a_{i_0} be the last digit of $(\mathsf{limit}\, f)_{<i}$[11]. Then we have:

1. $f\ \mathcal{X}^{nat\to r}\ g \quad\Rightarrow\quad f(i_0+3)\ \mathcal{X}^r\ g(i_0+3)$
   $\qquad\qquad\qquad\Rightarrow\quad to_{\mathcal{J}}(f(i_0+3)) \cap to_{\mathcal{J}}(g(i_0+3)) \neq \emptyset$

2. $\mathsf{is\_Str\_Cau}\,(g)\,(j+3) = tt \quad\Rightarrow$
   $\qquad\mu\left(to_{\mathcal{J}}(g(i_0+3)) \sqcap to_{\mathcal{J}}(g(j+3))\right) < 2^{-(i_0+3)}$

Combined with the fact that $to_{\mathcal{J}}\left((\mathsf{limit}\, f)_{<i}\right)$ contains not only $to_{\mathcal{J}}\left(f(i_0+3)\right)$ but also the $2^{-(i_0+3)}$ neighbourhood of it as well, we get:

$$to_{\mathcal{J}}\left((\mathsf{limit}\, f)_{<i}\right) \sqsubseteq to_{\mathcal{J}}\left(g(j+3)\right) \tag{4.8}$$

On the other hand once b_j is generated, we are assured of

$$to_{\mathcal{J}}\left((\mathsf{limit}\, g)_{<j+1}\right) \sqsubseteq to_{\mathcal{J}}\left(g(j+3)\right) \tag{4.9}$$

Equations (4.8) and (4.9) result in:

$$to_{\mathcal{J}}\left((\mathsf{limit}\, f)_{<i}\right) \cap to_{\mathcal{J}}\left((\mathsf{limit}\, g)_{<j+1}\right) \neq \emptyset$$ □

[11] Note that if $(\mathsf{limit}\, f)_{<i}$ has not stopped outputting digits yet then $i_0 = i - 1$.

Extensionality and the Necessity of is_Str_Cau

In retaining the extensionality of the whole framework, we had to hang on to the expensive is_Str_Cau predicate which definitely takes its toll on the efficiency of the computations. For the moment assume that there were no such predicate intermingled into the implementation of limit. So let limit′ be the variant of limit in which the expensive predicate is not employed. To obtain limit′ all one needs to do is to replace aux_limit of definition 4.4.67 on page 143 by aux_limit′ defined as follows:

$$\begin{aligned}
&\mathsf{aux_limit}' : [\mathbb{N}_\perp \to \mathcal{D}^\infty] \to \mathbb{N}_\perp \to \mathcal{D}^\infty \to \mathcal{D}^\infty \\
&\mathsf{aux_limit}'\ f\ n\ a = d : (\mathsf{aux_limit}'\ f\ (n+1)\ (a +\!\!+ [d])) \\
&\qquad \text{where} \quad d = \mathsf{cna}\ (f(n)_{<n})\ a
\end{aligned}$$

Under this condition, we manufacture a computation scenario which results in non-extensionality. Remember from Definition 4.4.9 on page 117 that d_{-1} and d_1 are representation of -1 and 1, respectively.

Example 4.6.7. *Consider $f, g : \mathbb{N}_\perp \to \mathcal{D}^\infty$ defined by:*

1. $\forall n \leq 3$:
   - $f(n) = \mathsf{M} : \mathsf{M} : d_{-1}$
   - $g(n) = \mathsf{L} : \mathsf{M} : d_1$
2. $\forall n > 3 : f(n) = g(n) = \mathsf{M} : \mathsf{M} : d_1$

Note that:

$$\begin{array}{lcl}
\forall n \leq 3 & : & to_{\mathcal{J}}(f(n)) = to_{\mathcal{J}}(g(n)) = -1/4 \\
\forall n > 3 & : & to_{\mathcal{J}}(f(n)) = to_{\mathcal{J}}(g(n)) = 1/4
\end{array}$$

Thus in particular $f\ \mathcal{X}^{nat \to r}\ g$. On the other hand it is not difficult to check that without is_Str_Cau*, we get:*

$$\begin{cases}
\mathsf{limit}'\ f = \mathsf{M} : \mathsf{R} : d_0 \\
\mathsf{limit}'\ g = \mathsf{L} : d_1
\end{cases}$$

where $d_0 = \mathsf{M} : \mathsf{M} : \mathsf{M} : \ldots$, which means:

$$\begin{cases}
to_{\mathcal{J}}(\mathsf{limit}'\ f) = 1/4 \\
to_{\mathcal{J}}(\mathsf{limit}'\ g) = 0
\end{cases}$$

Therefore without $\mathsf{is\_Str\_Cau}$ the extensionality is lost. In this particular case, if we apply limit instead of limit' we get:

$$\begin{cases} \mathsf{limit}\, f = [\mathsf{M}] \\ \mathsf{limit}\, g = [\mathsf{L}] \end{cases}$$

hence:

$$\mathsf{limit}\, f \ \mathcal{X}^r \ \mathsf{limit}\, g$$

The reason is that neither of f or g represent a legitimate strongly Cauchy sequence as we have:

$$\begin{cases} |to_{\mathcal{J}}(f(3)) - to_{\mathcal{J}}(f(4))| > 2^{-2} \\ |to_{\mathcal{J}}(g(3)) - to_{\mathcal{J}}(g(4))| > 2^{-2} \end{cases}$$

so according to our implementation of limit, after the first digit of the limit is generated, f and g fail to pass the strong Cauchyness test, i. e. $\mathsf{is\_Str\_Cau}$, and the process comes to a halt.

Of course it all comes back to careless programming. So in case the programmer is sensible enough to *only* implement strongly Cauchy sequences, this expensive test can be omitted from the actual implementation of SHRAD in order to raise the efficiency.

4.6.2 Extensionality at First-order

The material presented in subsection 4.6.1 can be summarized in:

Proposition 4.6.8. *$\mathcal{X}$ is a C-logical relation, where C is the set of* SHRAD-*constants.*

Combined with the fundamental lemma of logical relations (Lemma 2.4.5 on page 51) one gets:

Corollary 4.6.9. *For any* SHRAD-*term $M : \sigma$:*

$$[\![M]\!] \ \mathcal{X}^\sigma \ [\![M]\!]$$

The question arises:

What has this property got to do with extensionality?

Assume that a function $f : \mathcal{D}^\infty \to \mathcal{D}^\infty$ is defined in SHRAD, representing some $\hat{f} : [-1, 1] \rightharpoonup [-1, 1]$. Then take some $x \in [-1, 1]$ together with one of its representations $d_x \in \mathcal{D}^\omega$ and finally suppose that $f(d_x)$ represents a *total* real number, i. e. $f(d_x)$ is an infinite sequence. Now in case $e_x \in \mathcal{D}^\omega$ is any (other) representation of x, by Corollary 4.6.9 on the preceding page we are assured that:

1. If $f(e_x)$ is an *infinite* sequence, then it represents the same total real number as $f(d_x)$ does, i. e.
$$to_{\mathcal{J}}(f(e_x)) = to_{\mathcal{J}}(f(d_x))$$
2. If $f(e_x)$ is only a *finite* sequence, then:
$$to_{\mathcal{J}}(f(e_x)) \sqsubseteq to_{\mathcal{J}}(f(d_x))$$

As a result:

> It is *impossible* for *different* representations of the same total real number as input to result in representations of *different* total real numbers as output.

But does the fact that $f(d_x)$ is an infinite sequence automatically guarantee $f(e_x)$ to be infinite too? The answer is 'no'. It is still possible to write terms (say) f violating this property, in which case — by definition of $to_{\mathcal{J}^m_{max} \to \mathcal{J}_{max}}$ (Definition 4.4.3 on page 113) — we rule out x as being an element of $dom(\hat{f})$. This is because such a case arises when f is 'carelessly' implemented. To clarify this issue, we make the following further distinction:

Definition 4.6.10 (Strong versus Weak Extensionality).

- *Let $m \geq 1$ and $f : (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$. Then $\hat{f} : [-1, 1]^m \rightharpoonup [-1, 1]$ (or f itself for that matter) is said to be*

  1. strongly extensional *at $\boldsymbol{x} \in [-1, 1]^m$ if for any two $\boldsymbol{d_x}, \boldsymbol{e_x} \in (\mathcal{D}^\omega)^m$ representing $\boldsymbol{x}$*
  $$to_{\mathcal{J}}(f(\boldsymbol{d_x})) = to_{\mathcal{J}}(f(\boldsymbol{e_x})) \in [-1, 1]$$
  2. weakly extensional *at $\boldsymbol{x} \in [-1, 1]^m$ if for any two $\boldsymbol{d_x}, \boldsymbol{e_x} \in (\mathcal{D}^\omega)^m$ representing $\boldsymbol{x}$*
  $$to_{\mathcal{J}}(f(\boldsymbol{d_x})) \cap to_{\mathcal{J}}(f(\boldsymbol{e_x})) \neq \emptyset$$

- $\hat{f}$ *(or f) is* strongly (weakly) extensional *if it is strongly (weakly) extensional over all* $\boldsymbol{x} \in [-1, 1]^m$.

Proposition 4.6.11. *Strong extensionality implies the weak one (pointwise and overall alike).*

Proof. Follows from the fact that:

$$to_{\mathcal{J}}(f(\boldsymbol{d_x})) = to_{\mathcal{J}}(f(\boldsymbol{e_x})) \quad \Rightarrow \quad to_{\mathcal{J}}(f(\boldsymbol{d_x})) \cap to_{\mathcal{J}}(f(\boldsymbol{e_x})) \neq \emptyset \qquad \square$$

Definition 4.6.12. *For any type σ and each $m \geq 0$, the type expression $\sigma^m \to \sigma$ is defined recursively as:*

$$\begin{aligned} \sigma^0 \to \sigma &= \sigma \\ \sigma^{(m+1)} \to \sigma &= \sigma \to (\sigma^m \to \sigma) \end{aligned}$$

As a direct consequence of Corollary 4.6.9 on page 153, the main extensionality property of Shrad is stated as:

Theorem 4.6.13 (weak extensionality). *Let $m \geq 1$ and $f : r^m \to r$ be any term written in* Shrad. *Then $[\![f]\!] : (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$ is weakly extensional.*

In section 4.4 we demonstrated that all constants preserve $\simeq_{\mathcal{D}^\infty}$ of Definition 4.4.1 on page 113, except for *limit* and the Y_σ's. In section 4.8 we will introduce primitive recursion which for our purposes can be used instead of fix-point constants. On the other hand in Corollary 4.4.71 on page 144 we showed that in case the input to *limit* represents a strongly Cauchy sequence even *limit* preserves $\simeq_{\mathcal{D}^\infty}$. Therefore as long as this last condition is cautiously cared for, strong extensionality is guaranteed. Of course, the problem here is that this property does not fully characterize strong extensionality and after all, *it is not a syntactic characterization*. Yet from a certain viewpoint it can make sense. In section 4.9 we will explain how to write a function over real numbers by virtue of the *effective Weierstraß theorem*[12]. According to that procedure, each function is written either

[12]Theorem 4.7.8 on page 159.

as a polynomial, or the limit of a (suitably fast converging) sequence of polynomials. Once a term (say) $f : r^m \to r$ is written according to that procedure, strong extensionality (of $[\![f]\!] : (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$) is guaranteed.

Practice aside, the importance of weak extensionality is best seen in the theoretical side. In designing the language, by denying the user access to individual terms of the sequences (in $\mathcal{D}^\infty$), we have succeeded in obtaining weak extensionality. This is in contrast to other works based on (variations of) signed-digit binary representation available in the literature, such as [Di Gianantonio, 1993; Ménissier-Morain, 1996; Böhm et al., 1986].

4.7 Expressivity

In designing a framework for computation involving a language and its model, expressivity is one of the most important features. Naturally we like to be able to define more and more objects. Yet it should come as no surprise that more does not always mean better. For instance, take PCF as in [Plotkin, 1977]. The nagging problem of full abstraction leads to the augmentation of the language with parallel-or. Thus the cpo model becomes fully abstract with respect to PCF + $\{por\}$ (Theorem 2.3.11 on page 46) which is more expressive than PCF, nonetheless at the cost of embracing parallelism into the (up-to-then) sequential setting.

Now consider Weihrauch's approach [Weihrauch, 2000] to computability over real numbers, the so-called *Type-2 Theory of Effectivity* (TTE). We take TTE as the standard framework for computability over real numbers as — relative to other approaches — it scores high on both *expressivity* and *effectivity*. In fact a Type-2 machine has been realized [Weihrauch, 2000, Fig. 2.2, page 16] confirming the realistic nature of TTE. Here we do not get into the details of the subject and content ourselves with the following brief description.

Although the framework admits various representations of real numbers (resulting in different notions of computability), for our purposes it suffices to consider the one with real numbers being represented by shrinking infinite sequences of intervals with rational end-points (rational intervals). Now assume that for some suitable alphabet Σ and with some intermediate encoding, the set Σ^ω of all infinite sequences over Σ represents the set of all shrinking sequences of rational intervals which in turn represent real numbers. Thus one can define a representation $\nu : \Sigma^\omega \to \mathbb{R}$. This way we have functions over $\mathbb{R}$ on one hand, and *Type-2 machines* dealing with Σ^ω on the other hand. We do not present the definition of these machines here[13] as a mere intuition of computable string functions would suffice. The basic idea is that $f : \mathbb{R} \to \mathbb{R}$ is computable iff a string function F exists such that:

$$\begin{tikzcd} \Sigma^\omega \arrow[r, "F"] \arrow[d, "\nu"'] & \Sigma^\omega \arrow[d, "\nu"] \\ \mathbb{R} \arrow[r, "f"'] & \mathbb{R} \end{tikzcd}$$

commutes. The definition for $f : \mathbb{R}^n \to \mathbb{R}$ with $(n > 1)$ is similar. This way a rich library of functions over real numbers is obtained containing all the "standard" computable functions such as addition, multiplication, division, exponentiation, etc.

The problem with this framework is that there are string functions which do not realize any functions over real numbers. Moreover, TTE operates at the lowest possible level and offers no language at all. Therefore, in this thesis, we thought of selecting a number of functions as primitives and aiming to somewhat control the set of string functions definable while retaining a good deal of expressivity. We have already shown which primitives are selected (Definition 4.2.2 on page 108) and have presented weak extensionality (Theorem 4.6.13 on page 155) as the measure of control over definable string functions. Now, let us see how much expressivity we have managed to retain.

4.7.1 Effective Weierstraß Theorem

Notation 4.7.1. *For any $x \in \mathbb{R}$, by $|x|$ we mean the* absolute value *of x, i. e.*

$$|x| = \begin{cases} x & \textit{if} \quad x \geq 0 \\ -x & \textit{if} \quad x \leq 0 \end{cases}$$

Definition 4.7.2 (maximum norm). *The function $\|\,.\,\|_n : \mathbb{R}^n \to \mathbb{R}$ is defined by:*

$$\forall (x_1, \ldots, x_n) \in \mathbb{R}^n : \quad \|(x_1, \ldots, x_n)\|_n = \max\{|x_1|, \ldots, |x_n|\}$$

The subscript n may be dropped when it is easily understood from the context.

Let $n \geq 1$ and take some compact $X \subseteq \mathbb{R}^n$. It is true — both classically and constructively — that any continuous $f : X \to \mathbb{R}$ is uniformly continuous, i. e.

$$\forall \epsilon > 0\, \exists \delta > 0\, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n : \quad \|\boldsymbol{x} - \boldsymbol{y}\| < \delta \Rightarrow |f(\boldsymbol{x}) - f(\boldsymbol{y})| < \epsilon$$

[13] see [Weihrauch, 2000, section 2.1].

Thus whenever we talk about a continuous $f : X \to \mathbb{R}$, uniformity is implicitly implied as well.

Definition 4.7.3 ($C(X, Y)$). *Let $X \subseteq \mathbb{R}^n$ be compact and $Y \subseteq \mathbb{R}$ arbitrary. We define:*

$$C(X, Y) = \{f : X \to Y \mid f \text{ is continuous}\}$$

As we mostly deal with $[-1, 1]$, whenever $Y = [-1, 1]$ we simply drop it. Thus in particular, $C([-1, 1])$ is the set of all continuous $f : [-1, 1] \to [-1, 1]$.

For any compact $X \subseteq \mathbb{R}^n$, the set $C(X, \mathbb{R})$ contains functions of different shape. But the whole set is somewhat under control, and that is due to the presence of a dense subset consisting of "well-behaved" functions called polynomials. This is best expressed by the so-called Weierstraß theorem which holds both in classical[14] and constructive analysis[15].

Theorem 4.7.4 (Weierstraß Approximation Theorem [Bridges, 1979, page 94]).

Let X be a compact subset of $\mathbb{R}^n$, $f \in C(X, \mathbb{R})$ and $\epsilon > 0$. Then there exists a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ such that

$$\forall x \in X : |f(x) - p(x)| < \epsilon$$

This subject was taken even further independently by Hauck [1976] and Caldwell and Pour-El [1975] where it was proven that the above theorem holds even in a computable setting. Weihrauch [2000, pages 159 –161] adopted the idea in defining a representation for functions over (suitable compact intervals of) real numbers using fast converging Cauchy sequences of polynomials with rational coefficients[16].

Definition 4.7.5 (Strong Cauchy Sequences of Functions). *Let X be any set and $Y \subseteq \mathbb{R}$. By $\mathrm{Cauf}_{2^n}(X, Y)$ we mean the set of all sequences $\{f_n : X \to Y \mid n \in \mathbb{N}\}$ such that for any $x \in X$, the sequence $(f_n(x))_{n\in\mathbb{N}}$ is strongly Cauchy in Y (see Definition 4.4.56 on page 135), i. e.*

$$\forall x \in X : \quad (f_n(x))_{n\in\mathbb{N}} \in \mathrm{Cau}_{2^n}(Y)$$

[14] See [Rudin, 1976, page 159] or [Rudin, 1973, page115] for classical versions.

[15] See [Bridges, 1979, page 94] or [Bishop and Bridges, 1985, page 109] for constructive versions.

[16] In [Weihrauch, 2000, page 159] the *rational polygons* are originally used, but later (on page 160 of the same book) it is mentioned that rational polygons can be substituted by *rational polynomials*.

Definition 4.7.6 (Cauchy Representation of Functions). *Let $X \subseteq \mathbb{R}^n$ be arbitrary. We say that the sequence*

$$\{p_n : \mathbb{R}^n \to \mathbb{R} \mid n \in \mathbb{N}\}$$

of polynomials with rational coefficients represents $f : X \to \mathbb{R}$ *if and only if:*

1. $(p_n)_{n\in\mathbb{N}} \in \mathrm{Cauf}_{2^n}(X, \mathbb{R})$

2. $\forall x \in X : f(x) = \lim_{n\to\infty} p_n(x)$

In case the sequence can be effectively generated, we say that $(p_n)_{n\in\mathbb{N}}$ effectively *represents f.*

The following definition is derived from [Weihrauch, 2000]:

Definition 4.7.7 (Computable Cube). *The cube $[a_1, b_1] \times \cdots \times [a_n, b_n] \subseteq \mathbb{R}^n$ is called* computable *if and only if each a_i and b_i $(1 \leq i \leq n)$ is a TTE-computable real number.*

Now we are in the right position to present the effective Weierstraß theorem. Of course, the version we present here is adjusted to suit our own framework, otherwise it is equivalent to the one in [Weihrauch, 2000, Lemma 6.1.10, page 160]:

Theorem 4.7.8 (Effective Weierstraß Theorem). *Let $n \geq 1$ and $X \subseteq \mathbb{R}^n$ be a computable cube. Then for any $f : X \to \mathbb{R}$ the following are equivalent:*

1. *f is TTE-computable.*

2. *There exists a sequence $(p_n)_{n\in\mathbb{N}}$ of polynomials with rational coefficients effectively representing f.*

In the light of the theorem above, we embark on demonstrating how to define functions over real numbers using Shrad.

4.8 Primitive Recursion

As we need to define effective sequences of polynomials, perhaps it is more convenient to define *primitive recursion* exclusively.

Definition 4.8.1 ($\boldsymbol{rec_\sigma}$, $\boldsymbol{A_{rec_\sigma}}$). *For each type σ, i. e. including PCF-types, we define a term*

$$A_{rec_\sigma} : \gamma \to \gamma$$

where

$$\gamma := \sigma \to (nat \to \sigma \to \sigma) \to (nat \to \sigma)$$

as follows:

$$A_{rec_\sigma} := \lambda T a_0 f n.\ if\ Zero(n)\ a_0\ (f\, n\, (T\ a_0\ f\ (n-1))) \tag{4.10}$$

where:

1. $T : \sigma \to (nat \to \sigma \to \sigma) \to (nat \to \sigma)$
2. $a_0 : \sigma$
3. $f : nat \to \sigma \to \sigma$
4. $n : nat$

based on which the constant rec_σ is defined in terms of Y_γ by:

$$rec_\sigma = Y_\gamma\ A_{rec_\sigma} \tag{4.11}$$

Bearing in mind the standard interpretation of the fix-point constant we have:

Definition 4.8.2 (rec_σ).

$$\mathcal{A}_{\mathcal{SH}}(rec_\sigma) := \mathsf{rec}_\sigma = \sqcup\{[\![A_{rec_\sigma}]\!]^n(\bot) \mid n \in \mathbb{N}\}$$

Let us fix a type σ and consider a starting point $a_0 \in \mathbb{D}_\sigma$ together with a generating function $f : \mathbb{N}_\bot \to \mathbb{D}_\sigma \to \mathbb{D}_\sigma$. We define a sequence $(x_n)_{n\in\mathbb{N}}$ of elements of $\mathbb{D}_\sigma$ by:

$$\begin{array}{rcl} x_0 & = & a_0 \\ x_1 & = & f\ 1\ x_0 \\ \vdots & & \\ x_{n+1} & = & f\ (n+1)\ x_n \\ \vdots & & \end{array}$$

and a sequence of functions $(g_n)_{n\in\mathbb{N}} \subseteq [\mathbb{N}_\bot \to \mathbb{D}_\sigma]$ by:

$$\forall n \in \mathbb{N} \colon g_n := ([\![A_{rec_\sigma}]\!]^n(\bot))\ a_0\ f$$

Proposition 4.8.3.

$$\forall n \geq 1\ \forall m < n\colon\ g_n(m) = x_m$$

Proof. Easy induction:

base case $(n = 1, m = 0)$:

$$\begin{array}{rcl} g_1(0) & = & ([\![A_{rec_\sigma}]\!]^n(\bot))\ a_0\ f\ 0 \\ (\text{definition of } A_{rec_\sigma}) & = & a_0 \end{array}$$

inductive case $(n > 1)$:

$$\begin{array}{rcl} g_n(m) & = & ([\![A_{rec_\sigma}]\!]^n(\bot))\ a_0\ f\ m \\ & = & (([\![A_{rec_\sigma}]\!][\![A_{rec_\sigma}]\!]^{n-1})(\bot))\ a_0\ f\ m \\ & = & [\![A_{rec_\sigma}]\!]([\![A_{rec_\sigma}]\!]^{n-1}(\bot))\ a_0\ f\ m \\ (\text{definition of } A_{rec_\sigma}) & = & \begin{cases} a_0 & \text{if } m = 0 \\ f\ m\ ([\![A_{rec_\sigma}]\!]^{n-1}(\bot)\ a_0\ f\ (m-1)) & \text{if } m > 0 \end{cases} \\ (\text{definition of } (g_n)_{n\in\mathbb{N}}) & = & \begin{cases} a_0 & \text{if } m = 0 \\ f\ m\ (g_n(m-1)) & \text{if } m > 0 \end{cases} \\ (\text{ind. hyp.}) & = & \begin{cases} a_0 & \text{if } m = 0 \\ f\ m\ x_{m-1} & \text{if } m > 0 \end{cases} \\ (\text{definition of } (x_n)_{n\in\mathbb{N}}) & = & \begin{cases} a_0 & \text{if } m = 0 \\ x_m & \text{if } m > 0 \end{cases} \\ (x_0 = a_0) & = & \begin{cases} x_0 & \text{if } m = 0 \\ x_m & \text{if } m > 0 \end{cases} \\ & = & x_m \end{array}$$

□

On the other hand we have:

$$\begin{array}{rcl} \mathsf{rec}_\sigma\ a_0\ f & = & (\sqcup\{[\![A_{rec_\sigma}]\!]^n(\bot) \mid n \in \mathbb{N}\})\ a_0\ f \\ & = & \sqcup\{[\![A_{rec_\sigma}]\!]^n(\bot)\ a_0\ f \mid n \in \mathbb{N}\} \\ (\text{definition of } (g_n)_{n\in\mathbb{N}}) & = & \sqcup\{g_n \mid n \in \mathbb{N}\} \end{array}$$

Therefore, we can summarize our observation of the behaviour of rec_σ in the following lemma:

Lemma 4.8.4. *Let σ be any type and consider $a_0 \in \mathbb{D}_\sigma$ and $f \in [\mathbb{N}_\perp \to \mathbb{D}_\sigma \to \mathbb{D}_\sigma]$. Define the sequence $(x_n)_{n\in\mathbb{N}}$ by:*

$$\begin{array}{rcl} x_0 & = & a_0 \\ x_1 & = & f\ 1\ x_0 \\ \vdots & & \\ x_{n+1} & = & f\ (n+1)\ x_n \\ \vdots & & \end{array}$$

then the following holds:

$$\forall n \in \mathbb{N}\colon \mathsf{rec}_\sigma\ a_0\ f\ n = x_n$$

4.8.1 Primitive Recursion and Strong Extensionality

Primitive recursion obviously preserves $\simeq_{\mathcal{D}^\infty}$ of Definition 4.4.1 on page 113. Moreover, in section 4.4 we demonstrated that all constants except *limit* and Y_σ's preserve $\simeq_{\mathcal{D}^\infty}$ too. Therefore:

Theorem 4.8.5 (strong extensionality). *Assume that $\mathcal{L}$ is the language derived from* Shrad *by omitting limit and replacing Y_σ's by rec_σ's at each type σ. Let $m \geq 1$ and $f : r^m \to r$ be any term written in $\mathcal{L}$. Then $[\![f]\!] : (\mathcal{D}^\infty)^m \to \mathcal{D}^\infty$ is strongly extensional.*

4.9 How to Define a Function

In our setting — by virtue of the effective Weierstraß theorem — the procedure for defining a function starts with defining a suitable effective sequence of polynomials with rational coefficients, which we simply call *rational polynomials*. As we are merely sketching the procedure, let us focus on functions with one argument over real numbers. Also, for now suppose that everything we talk about happens inside $[-1, 1]$. We will later on show how this can be extended to cover the general case.

4.9.1 Polynomials

First things first. In order to define the polynomial

$$p = \sum_{i=0}^{n} a_i x^i$$

one needs:

multiplication We already have a primitive for this operation in the language.

addition Well, we are half-way through as we have average as a primitive. Thus we define:

$$\begin{aligned} add &: r \to r \to r \\ add &= \lambda xy.\, M^{-1}(avg\, x\, y) \end{aligned}$$

Now if we allow the infix binary operator $\boxplus : [-1,1] \to [-1,1] \to [-1,1]$ to be

$$\boxplus := to_{\mathcal{I}^2_{max} \to \mathcal{I}_{max}} [\![add]\!]$$

it should not be difficult to see that over total real numbers in $[-1,1]$:

$$\forall x, y \in [-1,1] : \quad x \boxplus y = \max\{-1, \min\{x + y, 1\}\}$$

rational coefficients Rational numbers in $[-1,1]$ are provided via *qtoR*, but we must be careful as even when we are approximating some $f : [-1,1] \to [-1,1]$ with rational polynomials, the coefficients need not be restricted to $[-1,1]$. Thus, we implement an operation:

$$\begin{aligned} \text{qsplit} : \mathbb{Q} &\to \mathbb{N} \times \mathbb{Q} \\ q &\mapsto (n, q') \end{aligned}$$

where $q' \in [-1,1]$ and $2^n \times q' = q$. The implementation (in PCF) is fairly straightforward and we leave it to the reader. The only thing to remember is that we represented rational numbers as *triplets of natural numbers* (see subsection 4.4.3) and as we do not have product types in SHRAD (Definition 4.2.1 on page 108) a proper implementation requires *four* components (say)

$$qsplit_i : \underbrace{nat \to nat \to nat}_{\mathbb{Q}} \to nat$$

for $i \in \{1, 2, 3, 4\}$.

Notation 4.9.1. *The type nat → nat → nat is abbreviated by* ***rat*** *(for* rational*).*

Now let us put the pieces together. As the rational numbers ended up being decomposed into pairs of natural and rational numbers, accordingly we need to equip ourselves with new operations allowing us multiply out-of-range rational numbers with real numbers in $[-1, 1]$, the same for addition. But these are common-sense practices in programming. For example, one can implement the following operation:

$$\begin{aligned} \mathrm{qmult} : \mathbb{Q} \times [-1,1] &\to [-1,1] \\ (q,x) &\mapsto \max\{-1, \min\{s,1\}\} \\ \text{where} \quad s &= 2^n (q' x) \\ (n,q') &= \mathrm{qsplit}\, q \end{aligned}$$

by:

$$\begin{aligned} qmult &: rat \to r \to r \\ qmult &= \lambda qx.\, (M^{-1})^n\, (mult\ qtoR(q')\ x) \\ &\text{where } (n,q') = \mathrm{qsplit}\, q \end{aligned}$$

For adding (out-of-range) rational numbers to real numbers in $[-1, 1]$ one may define:

$$\begin{aligned} qadd &: rat \to r \to r \\ qadd &= \lambda qx.\, (M^{-1})^n\, (add\ qtoR(q')\ M^n(x)) \\ &\text{where } (n,q') = \mathrm{qsplit}\, q \end{aligned}$$

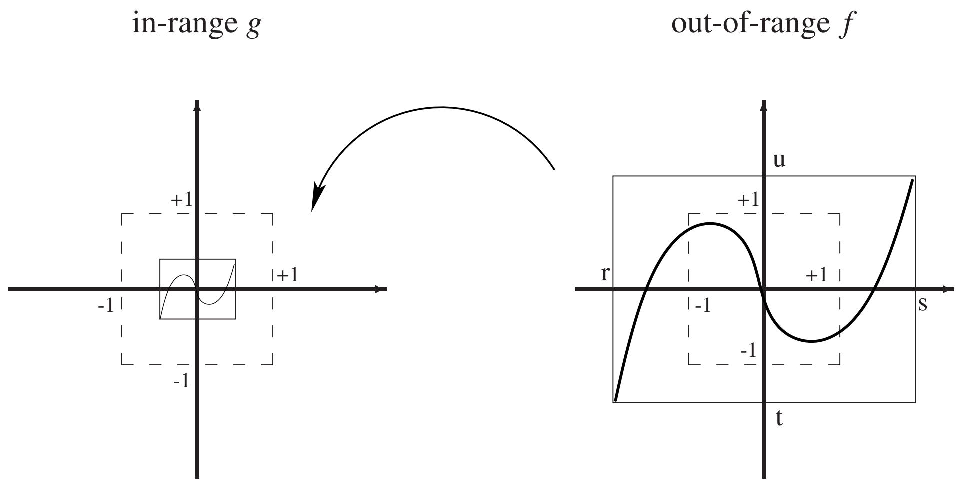
4.9.2 Out-of-range Functions

Our framework deals only with real numbers in $[-1, 1]$. Hence, any function working outside this range must be scaled and moved inside this interval.

Convention 4.9.2. *From now on, by* computable *we mean* TTE-computable*.*

Assume that for some computable $r, s \in \mathbb{R}$, a computable $f : [r, s] \to \mathbb{R}$ has caught our eyes and we want to implement it in SHRAD. As computability implies continuity, the range of f must be some compact subset of $\mathbb{R}$ — in fact some compact interval (say) $[t, u]$. In case both $dom(f)$ and $range(f)$ happen to be subsets of $[-1, 1]$, no scaling is needed and we can proceed directly to the implementation of f in SHRAD. If on the other hand, any of them goes beyond

Figure 4.2 Handling out-of-range functions



the limit, we must engage in some scaling. For that, let k be — say, the least — natural number such that:

$$\forall x \in dom\,(f) \cup range\,(f) : |x| < 2^k$$

If we define $g : \mathbb{R} \rightharpoonup \mathbb{R}$ by:

$$g(x) = \frac{f(2^k x)}{2^k}$$

it can easily be verified that:

$$\begin{aligned} dom\,(g) &= [r/2^k, s/2^k] \\ range\,(g) &= [t/2^k, u/2^k] \end{aligned}$$

hence both $dom\,(g)$ and $range\,(g)$ are subsets of $[-1, 1]$ (Figure 4.2) and as computability of g easily follows that of f, we can proceed to implement g in Shrad. Once this has been done, f can be calculated by re-scaling, i. e.

$$\forall x \in [r, s] : f(x) = 2^k g(x/2^k)$$

It is easily seen that this procedure can be generalized to the case of functions with more than one argument.

4.9.3 Test of Practice

The procedure outlined so far is theoretically appealing. But how feasible is the whole process in specific concrete cases? The first example that we discuss is the case of *exponential function* over $[-1, 1]$. As the range stretches beyond $[-1, 1]$, we instead consider $f : [-1, 1] \to [-1, 1]$ defined by:

$$f(x) = \frac{1}{4}\exp(x)$$

This f is "in-the-range", i. e.

$$\mathit{range}(f) \subseteq [-1, 1]$$

From [Weihrauch, 2000, Example 4.3.3, page 111] we learn that exp is computable, hence there exists an effective representation of f with rational polynomials. In fact we may just resort to the *Taylor* series

$$f(x) = \frac{1}{4}\sum_{n=0}^{\infty}\frac{x^n}{n!}$$

and define the sequence $(p_n)_{n\in\mathbb{N}}$ by:

$$\forall n \in \mathbb{N} : p_n(x) = \frac{1}{4}\sum_{i=0}^{n}\frac{x^i}{i!}$$

Incidentally $(p_n)_{n\in\mathbb{N}}$ is a strongly Cauchy sequence of polynomials over $[-1, 1]$, so all we need to do is to apply *rec* of section 4.8 combined with the procedure outlined in subsection 4.9.1 for implementing rational polynomials to get $(p_n)_{n\in\mathbb{N}}$. As the last step, we apply limit and we are done.

Even *sine* and *cosine* have Taylor series whose coefficients diminish factorially which makes them easily implementable in Shrad.

$$\begin{aligned}
\sin(x) &= \sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!} \\
\cos(x) &= \sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}
\end{aligned}$$

Yet there are important functions with rather *slow* converging Taylor series. After all, we admit that Taylor series were in no way tailor made for Shrad. Take *natural* *logarithm* for instance. For $|x| < 1$ we have:

$$\log(1 + x) = \sum_{j=1}^{\infty} (-1)^{j+1} \frac{x^j}{j}$$

Also notice that $\log(1 + x)$ tends to $-\infty$ as $x \to -1$. So each time we need to content ourselves with some interval

$$[-1 + \frac{1}{2^n}, 1] \qquad \text{(for some } n \in \mathbb{N}\text{)}$$

and scale the range of $\log(1 + x)$ on this domain down to $[-1, 1]$ and then try to come up with a sequence of rational polynomials which effectively represents $\log(1 + x)$ on this domain. The case for the function

$$x \mapsto \frac{1}{x}$$

is more or less the same.

Part III

Concluding Material

Chapter 5

Summary of the Results

Since Di Gianantonio [1993] introduced his semantics for exact real number computation, there has always been a struggle to maintain data abstraction and efficiency as much as possible. The interval domain model — or its variations — can be regarded as the standard setting to obtain maximum data abstraction. As for efficiency there has been much focus on sequentiality to the extent that these two terms have become almost synonymous. Escardó et al. [1998, 2004] demonstrated that there is not much one can get by sequential computation in the interval domain model. In Chapter 3 we reinforced this result by exposing the limited power of (some extensions of) the sequential fragment of Real-PCF.

The previous argument suggests some sort of compromise in the beauty of the model in order to keep efficiency. One way forward is to try to sacrifice *extensionality*. This is exactly what we did in Chapter 4. There we succeeded in presenting a framework for exact real number computation which satisfies the following all at the same time:

1. It is sequential.
2. Multi-valuedness is carefully avoided.
3. A 'good degree' of expressivity is retained.

Chapter 6

Future Work

6.1 Piece-wise Affinity

In Theorem 3.3.16 on page 86 we derived a necessary condition for wPR-definability of total unary first-order functions over real numbers. We believe that by imposing computability property over the parameters r, s, c_1 and c_2 a sufficient condition can be obtained as well:

Conjecture 6.1.1. *Let f be as in the statement of Theorem 3.3.16 on page 86, then f is wPR-definable if and only if the parameters r, s, c_1 and c_2 are computable.*

In Definition 3.3.8 on page 77 we defined the set of logical relations R_k which incorporated the meaning of piece-wise affinity and as stated in Theorem 3.3.16 on page 86 we admit the fact that these relations are far from characterizing piece-wise affinity, as they are not preserved by decreasing affine functions. Yet they offer a good chance of leading to some kind of full characterization of piece-wise affinity via logical relations. Thus we propose:

Problem 6.1.2. *Is it possible to fully characterize piece-wise affinity via logical relations?*

Conjecture 6.1.3. *Let $\mathcal{PA}$ be a set of constants such that for any $c \in \mathcal{PA}$, $[\![c]\!]$ is piece-wise affine and denote the extension of* wRPCF *with $\mathcal{PA}$ by* $\mathrm{wRPCF}_{\mathcal{PA}}$. *Then any first-order* $\mathrm{wRPCF}_{\mathcal{PA}}$*-definable function is piece-wise affine.*

6.2 The Language Shrad

In Chapter 4 we presented the language Shrad and demonstrated some of its key properties. Yet it should (hopefully) only be the beginning. The future directions may be categorized as follows:

Theory: On the more theoretical side, we draw attention to the emergence of a new family of mathematical objects used as models of data types. We modelled Shrad inside a *category of cpo's equipped with a special logical relation* and demonstrated that all Shrad-definable objects preserve the logical relation. Looking from the opposite angle, we managed to purge the original category of cpo's (without the logical relation) from those unwanted objects which do not preserve the logical relation. But then the question of universality comes up:

Problem 6.2.1. *How is it possible to extend* Shrad *to a language which is universal with respect to the specific category of cpo's $\{\mathbb{D}_\sigma \mid \sigma \in \mathbb{T}_{\mathcal{SH}}\}$ of Definition 4.3.9 on page 110 together with the logical relation $\mathcal{X}$ of Definition 4.6.2 on page 146.*

This in turn necessitates a more comprehensive study of this model which is as of now not so well-known.

We proved the weak-extensionality of the setting in Theorem 4.6.13 on page 155. Nevertheless one may wonder if this can be strengthened to strong extensionality by e. g. coming up with a new implementation of limit operator and replacing fix-point constants with their weaker counterparts, i. e. primitive recursion.

We can push this matter even further. In Section 4.4 we demonstrated that avg is extensional over the *total* real numbers while it is not extensional over the *partial* ones.[1] Now it is perfectly legitimate to ask whether based on the same model as ours it is possible to come up with other forms of implementations for average, multiplication, etc which are extensional over *both* partial and total real numbers. Our feeling is negative though we do not yet have a proof. Thus we propose:

Conjecture 6.2.2. *There exists no function* $\mathsf{avg}' : \mathcal{D}^\infty \to \mathcal{D}^\infty \to \mathcal{D}^\infty$ *such that:*

1. $\forall x, y \in \mathcal{D}^\omega : to_{\mathcal{J}}(\mathsf{avg}'\ x\ y) = (to_{\mathcal{J}}(x) + to_{\mathcal{J}}(y))/2$

2. $\forall x_1, x_2, y_1, y_2 \in \mathcal{D}^\infty : [(to_{\mathcal{J}}(x_1) = to_{\mathcal{J}}(x_2)) \wedge (to_{\mathcal{J}}(y_1) = to_{\mathcal{J}}(y_2))] \Rightarrow (to_{\mathcal{J}}(\mathsf{avg}'\ x_1\ y_1) = to_{\mathcal{J}}(\mathsf{avg}'\ x_2\ y_2))$

On the other hand, one may think of studying variations of Shrad obtained by adding/replacing the primitives. One immediate choice is to add a constant for integration and then employ the implementation by Longley [1998, 1999].

Another interesting primitive is cases by Escardó [2000] which can be used for definition by cases over reals in a signed-digit binary setting. A way forward is to study the languages obtained by:

1. Adding cases to Shrad.

2. Adding cases to Shrad while dropping *limit*.

Practice: On a more practical side we need to analyze the efficiency of the framework. Although parallelism is avoided, the complexity issue needs to be taken care of. It would be interesting to know the complexity of a few basic operations such as limit, exponential, etc.

[1] See Equation (4.2) on page 113 and Corollary 4.4.34 on page 127.

Bibliography

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.

R. Amadio and P.-L. Curien. *Domains and Lambda Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998. ISBN 0521622778.

H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

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

R. Bird. *Introduction to Functional Programming using Haskell*. Prentice Hall Europe, second edition, 1998.

Errett Bishop and Douglas S. Bridges. *Constructive Analysis*, volume 279 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin, 1985.

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

H.-J. Böhm and R. Cartwright. Exact real arithmetic: Formulating real numbers as functions. In D. Turner, editor, *Research Topics in Functional Programming*, pages 43–64. Addison Wesley, 1990.

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

Vasco Brattka. Recursive characterization of computable real-valued functions and relations. *Theoretical Computer Science*, 162:45–77, 1996.

Douglas S. Bridges. *Constructive Functional Analysis*. Number 28 in Research Notes in Mathematics. Pitman, London, 1979.

J. Caldwell and Marian Boykan Pour-El. On a simple definition of computable functions of a real variable — with applications to functions of a complex variable. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21: 1–19, 1975.

Paul J. Cohen. The independence of the continuum hypotheis. *Proceedings of the National Academy of Sceinces of the United States of America*, 50:1143–1148, 1963.

Paul J. Cohen. The independence of the continuum hypotheis. *Proceedings of the National Academy of Sceinces of the United States of America*, 51:105–110, 1964.

Nigel Cutland. *Computability : An Introduction to Recursice Function Theory*. Cambridge University Press, 1980.

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

Pietro Di Gianantonio. An abstract data type for real numbers. *Theoretical Computer Science*, 221:295–326, 1999.

A. Edalat and R. Heckmann. Computing with real numbers: (i) LFT approach to real computation (ii) domain-theoretic model of computational geometry. In G. Barthe, P. Dybjer, L. Pinto, and J. Saraiva, editors, *Applied Semantics: Advanced Lectures*, volume 2395 of *Lecture Notes in Computer Science*, pages 193–267. Springer Verlag, 2002.

A. Edalat, P. J. Potts, and Ph. Sünderhauf. Lazy computation with exact real numbers. In *Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming*, pages 185–194. ACM, 1998.

Abbas Edalat and Peter John Potts. A new representation for exact real numbers. In S. Brookes and M. Mislove, editors, *Electronic Notes in Theoretical Computer Science*, volume 6. Elsevier, 2000.

Herbert B. Enderton. *Elements of Set Theory.* Academic Press, New York (etc.); London, 1977.

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

M. H. Escardó, M. Hofmann, and T. Streicher. On the non-sequential nature of the interval-domain model of exact real-number computation. *Mathematical Structures in Computer Science*, 2004. Accepted for publication.

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

Martín Hötzel Escardó. *PCF Extended with Real Numbers: A Domain-Theoretic Approach to Higher-Order Exact real Number Computation*. PhD thesis, The University of Edinburgh, Department of Computer Science, November 1997.

Martín Hötzel Escardó. Effective and sequential definition by cases on the reals via infinite signed-digit numerals. In Abbas Edalat, Achim Jung, Klaus Keimel, and Marta Kwiatkowska, editors, *Electronic Notes in Theoretical Computer Science*, volume 13. Elsevier, 2000.

Amin Farjudian. Conservativity of wRPCF over PCF. Unpublished Manuscript, 2003a.

Amin Farjudian. Sequentiality and piecewise affinity in fragments of Real-PCF. To Appear in ENTCS, 2003b.

Kurt Gödel. The consistency of the axiom of choice and of the generalized continuum hypothesis. *Proceedings of the National Academy of Sceinces of the United States of America*, 24:556–557, 1938.

Kurt Gödel. The consistency of the axiom of choice and of the generalized continuum hypothesis. *Annals of Mathematical Studies*, 3, 1940a.

Kurt Gödel. *The Consistency of the Axiom of Choice and of the Generalized Continuum Hypothesis with the Axioms of Set Theory*. Princeton University Press, Princeton, NJ, 1940b.

Jürgen Hauck. Berechenbare reelle Funktionenfolgen. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 22:265–282, 1976.

Reinhold Heckmann. The appearance of big integers in exact real arithmetic based on linear fractional transformations. In *Foundations of Software Science and Computation Structures*, volume 1378, pages 172–188. Springer, 1998.

Reinhold Heckmann. Big integers and complexity issues in exact real arithmetic. In Abbas Edalat, Achim Jung, Klaus Keimel, and Marta Kwiatkowska, editors, *Electronic Notes in Theoretical Computer Science*, volume 13. Elsevier, 2000a.

Reinhold Heckmann. How many argument digits are needed to produce n result digits? In Abbas Edalat, David Matula, and Philipp Sünderhauf, editors, *Electronic Notes in Theoretical Computer Science*, volume 24. Elsevier, 2000b.

Reinhold Heckmann. Translation of taylor series into LFT expansions. *Symbolic Algebraic Methods and Verification Methods*, 2001.

Reinhold Heckmann. Contractivity of linear fractional transformations. *Theoretical Computer Science*, 279, (1):65–82, 2002.

A. Jung and J. Tiuryn. A new characterization of lambda definability. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 245–257. Springer Verlag, 1993.

Michal Konečný. *Many-valued real functions computable by finite transducers using IFS-representations*. PhD thesis, School of Computer Science, The University of Birmingham, 2000. Dissertation.

Michal Konečný. Real functions computable by finite automatons using affine representations. *Theoretical Computer Science*, 284(2):373–396, July 2002.

John Longley. When is a functional program not a functional program?: A walkthrough introduction to the sequentially realizable functionals., 1998. ML source file, available from `http://www.dcs.ed.ac.uk/hom/jrl/`.

John Longley. When is a functional program not a functional program? In *Proceedings of the Fourth ACM SIGPLAN International Conference on Functional Programming*, pages 1–7, New York, NY, USA, September 1999. ACM Press.

S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer Verlag, 1971.

J. R. Marcial-Romero and M. H. Escardó. Semantics of a sequential language for exact real-number computation. To appear in LICS, 2004.

C. McLarty. *Elementary Categories, Elementary Toposes*, volume 21 of *Oxford Logic Guides*. Oxford University Press, 1992.

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

Norbert Müller. The iRRAM: Exact arithmetic in C++. In *Workshop on Constructivity and Complexity in Analysis*, Swansea, 2000.

Andrew M. Pitts. Nominal logic: A first order theory of names and binding. In N.Kobayashi and B. C. Pierce, editors, *LNCS*, volume 2215, pages 219–242. Springer-Verlag Berlin Heidelberg, 2001.

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

Dave Plume. A calculator for exact real number computation. 4th Year Project Report, Department of Computer Science and Artificial Intelligence, University of Edinburgh, 1998.

P. Potts, A. Edalat, and M. H. Escardó. Semantics of exact real arithmetic. In *Twelfth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1997.

Walter Rudin. *Functional Analysis*. McGraw Hill Series in Higher Mathematics. McGraw Hill, New York, 1973.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw Hill, Auckland; London, third edition, 1976.

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

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

Kishor S. Trivedi and Miloš D. Ercegovac. On-line algorithms for division and multiplication. *IEEE Transactions on Computers*, C-26(7):681–687, July 1977.

J. Vuillemin. Syntaxe, Sèmantique et Axiomatique d'un Langage de Programmation Simple, 1974. Thèse de Doctorat d'Etat, Université Paris VII.

Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

Index