Electronic Notes in Theoretical Computer Science 10 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume10.html 16 pages

Effective and sequential definition by cases on the reals via infinite signed-digit numerals

Martín Hötzel Escardó [1]

LFCS, Department of Computer Science, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland

Abstract

The lexicographical and numerical orders on infinite signed-digit numerals are unrelated. However, we show that there is a computable normalization operation on pairs of signed-digit numerals such that for normal pairs the two orderings coincide. In particular, one can always assume without loss of generality that any two numerals that denote the same number are themselves the same. We apply the order-normalization operator to easily obtain an effective and sequential definition-by-cases scheme in which the cases consist of inequalities between real numbers.

Key words: Real number computation, parallel conditional.

1 Introduction

It is a surprising fact, discovered by the constructive mathematician Brouwer [5] in 1920, that potentially infinite decimal numerals are not suitable for effective (exact) real number computation—see Section 2.1 below. After this discovery, he and other mathematicians, logicians and computer scientists proposed many essentially equivalent concrete representations of real numbers that are suitable for constructive or mechanical computation—see Weihrauch [26] for some classical examples.

In this paper we consider a variation of Brouwer's solution, proposed by Wiedmer [29] in 1980 and further investigated by Boehm *et al.* [4], Di Gianantonio [8,10] and Weihrauch [26], among others, which consists of the use of *signed-digit* numerals. One learns from Di Gianantonio [9] that Leslie [14] considered signed-digit numerals in 1817 from a philosophical point of view, and from Sünderhauf [23] that Cauchy [6] observed in 1840 that signed-digit numerals simplify computations by hand. Avizienis [2] proposed in 1964 signed-digit numerals as an efficient internal representation of finite-precision numbers for hardware implementation of arithmetic.

[1] M.Escardo@ed.ac.uk

Elegant algorithms for exact Riemann integration and global maximum determination via signed-digit numerals were recently developed by Alex Simpson [21].

1.1 *Effective case analysis on the reals*

In all effective approaches to exact real number computation via concrete representations, the (in)equality relations are undecidable. In particular, this means that it is not possible to obtain exact algorithms from finite-precision algorithms by simply changing the underlying representation of numbers. The reason is that (in)equality tests are the basic ingredient for branching and looping.

Nevertheless, many definitions by cases consisting of inequalities, such as

$$\min(x,y) = \begin{cases} x & \text{if } x \leqslant y, \\ y & \text{otherwise,} \end{cases}$$

do produce computable functions. The point is that such functions cannot be computed by first evaluating the condition and then computing the corresponding branch.

But some general case-analysis schemes do exist. For example, if $f, g : \mathbb{R} \to \mathbb{R}$ are computable functions that agree at a computable number x_0, then the function $h : \mathbb{R} \to \mathbb{R}$ defined by

$$h(x) = \begin{cases} f(x) & \text{if } x \leqslant x_0, \\ g(x) & \text{if } x \geqslant x_0 \end{cases}$$

is also computable [26, Lemma 3.8]. In fact, as observed by Alex Simpson (personal communication), h can be implemented simply by

$$h(x) = f(\min(x, x_0)) + g(\max(x, x_0)) - f(x_0).$$

Here we consider a slightly more general computable case-analysis operator specified by

$$\text{cases}(x,t,y,z) = \begin{cases} y & \text{if } x < t \text{ or } y = z, \\ z & \text{if } x > t \text{ or } y = z, \end{cases}$$

with domain of definition $\{(x,t,y,z) \mid x = t \text{ implies } y = z\}$. Given this domain of definition, an equivalent specification is

$$\text{cases}(x,t,y,z) = \begin{cases} y & \text{if } x \leqslant t, \\ z & \text{if } x \geqslant t. \end{cases}$$

In order to see that this scheme is more general than the previous one, notice that

$$\min(x, y) = \text{cases}(x, y, x, y), \qquad \max(x, y) = \text{cases}(x, y, y, x)$$

and that h can be implemented by

$$h(x) = \text{cases}(x, x_0, f(x), g(x)).$$

In order to see that it is strictly more general, notice that the case-analysis operator cannot be implemented from the previous case-analysis scheme because it is a partial function and the functions used in the first implementation of h are all total. In fact, computable *partial* functions such as

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{if } x > 0 \end{cases}$$

can be implemented by

$$\text{sgn}(x) = \text{cases}(x, 0, -1, 1).$$

The previous scheme with $f(x) = -1$, $g(x) = 1$, and any choice of x_0 would produce an incorrect implementation, because

$$h(x) = f(\min(x, x_0)) + g(\max(x_0, x)) - f(x_0) = -1 + 1 - 1 = -1$$

and hence $h \neq \text{sgn}$.

Finally, notice that the case-analysis operator is a continuous map, defined on a subset of $\mathbb{R}^4$, which cannot be extended to a continuous map on any larger subset. In other words, the points (x, t, y, z) with $x = t$ but $y \neq z$ are singularities of the case-analysis operator. This means that the partial character of the case-analysis operator is due to topological rather than recursion-theoretic reasons.

1.2 Lexicographical order versus numerical order

In order to implement the above case-analysis operator (Section 4), we first investigate the connections between the lexicographical and numerical orders of infinite signed-digit numerals (Section 3). The main result is that there is a computable normalization operation on pairs of numerals such that for normal pairs the two orderings coincide. In particular, one can always assume without loss of generality that any two numerals that denote the same number are themselves the same. This is surprising because single numerals cannot have effectively determinable canonical forms—see Section 2.4.

1.3 Sequential case analysis on the reals

Sequentiality issues arise when one considers *partial elements*. For this part of the paper we assume some familiarity with domain theory [1,18] and the programming language PCF [16,11]. As opposed to other case-analysis operators considered in the literature [3,7], the above operator admits a sequential implementation.

One of the simplest examples of a "parallel" operation is the so-called *parallel-or* [20,16], defined by the following table:

	$\perp$	false	true
$\perp$	$\perp$	$\perp$	true
false	$\perp$	false	true
true	true	true	true

Notice that this is part of Kleene's three-valued logic [12]. Since the computational interpretation of $\perp$ is non-termination, this function requires some form of parallel evaluation. In fact, if the first argument is always evaluated first and it happens to be $\perp$, then the value of parallel-or at $(\perp, \text{true})$, namely true, cannot be correctly evaluated, because one never gets the chance of evaluating the second argument. If, on the other hand, the second argument is always evaluated first, the same problem arises with the evaluation of parallel-or at $(\text{true}, \perp)$. The only way out is to evaluate both arguments in a parallel fashion. In recursion theory, this kind of evaluation is referred to as *dovetailing* [19] and is usually achieved via Kleene's T-predicate [12].

Parallel-or is not definable in sequential languages such as PCF [16, consequence of Activity Lemma 4.2, pages 235–236]. The (extremely hard) problem of characterizing sequentiality in semantic terms is outside the scope of this paper. An operational formulation is given by the Activity Lemma. Roughly, it says that every unevaluated PCF expression has a unique "active subexpression", which is the one that has to be evaluated first. This property fails when one extends PCF with parallel-or or related constructions.

The case-analysis operator defined above is related to the so-called *parallel conditional*. For the flat domain of natural numbers, it is defined by

$$\text{pif } \ p \ \text{ then } \ x \ \text{ else } \ y = \begin{cases} x & \text{if } p = \text{true or } x = y, \\ y & \text{if } p = \text{false or } x = y, \\ \perp & \text{otherwise.} \end{cases}$$

As for parallel-or, no argument of the parallel conditional can be safely evaluated first, because

$$\begin{aligned} &\text{pif true then } x \text{ else } \bot = x, \\ &\text{pif false then } \bot \text{ else } y = y, \\ &\text{pif } \bot \text{ then } x \text{ else } x = x. \end{aligned}$$

The parallel conditional has a *pseudo-parallel conditional* as a relative, defined by

$$\text{ppif } p \text{ then } x \text{ else } y = \begin{cases} x & \text{if } p = \text{true or } x = y \text{, but } y \neq \bot, \\ y & \text{if } p = \text{false or } x = y \text{, but } x \neq \bot, \\ \bot & \text{otherwise.} \end{cases}$$

This construction *is* sequential, because one can first evaluate one of the branches, then the other, and finally, if necessary, the condition. In fact, it can be implemented from the usual sequential conditional by

$$\text{ppif } p \text{ then } x \text{ else } y = \text{if } (x = y) \text{ then } x \text{ else if } p \text{ then } x \text{ else } y.$$

The author doesn't know whether this has already been observed.

The parallel conditional generalizes from the flat domain of natural numbers to any cpo with binary meets such that the binary-meet operation is continuous (e.g. (algebraic and) continuous Scott domains) [17]:

$$\text{pif } p \text{ then } x \text{ else } y = \begin{cases} x & \text{if } p = \text{true}, \\ y & \text{if } p = \text{false}, \\ x \sqcap y & \text{if } p = \bot. \end{cases}$$

It is not clear to the author whether the pseudo-parallel conditional can also be generalized to a large class of domains. In this paper we consider a generalization to a domain of signed-digit numerals (Section 4).

Boehm and Cartwright [3] observed that the parallel conditional is useful in connection with a computable partial inequality test

$$(x <_\bot y) = \begin{cases} \text{true} & \text{if } x < y, \\ \text{false} & \text{if } x > y, \\ \bot & \text{otherwise.} \end{cases}$$

For instance, one can implement min by

$$\min(x, y) = \text{pif } x <_\bot y \text{ then } x \text{ else } y,$$

because when the test diverges the parallel conditional still converges as the two branches are equal:

$$\min(x, x) = \text{pif } \perp \text{ then } x \text{ else } x = x \sqcap x = x.$$

This was explored in detail by the author in the context of the interval domain [7].

Notice that the above observation makes sense only for *extensional* domains of real numbers. By "extensional" here we mean that each real number has a unique representation. An example is the interval domain. For *intensional* domains, the meet of two different representations of a number is not a third representation of the number in general. A counter-example is given by the domain of signed-digit numerals considered in this paper. It consists of finite and infinite sequences of signed-digits endowed with the prefix order. Hence meets are greatest common prefixes. However, if one takes meets after performing order-normalization, correct results are obtained.

In summary, our sequential implementation of the case-analysis operator (Section 4) relies on (1) order-normalization, (2) the partial inequality relation, (3) a pseudo-parallel conditional for the domain of numerals. This explicit factorization via the partial inequality relation and the pseudo-parallel conditional doesn't make sense when partial truth values and numerals are not available, so a direct construction is also given.

One has to be careful about what is meant by "evaluate first" when infinite objects (such that infinite sequences or functions) are involved. It is surely not meant that the object is completely evaluated, because this doesn't even make sense. Rather, it is meant that a canonical "part" of the object is evaluated first. For infinite sequences, this part will usually be the first term of the sequence (the "head").

1.4 Organization

For the reader's convenience, we begin by recalling the basic facts about real number computation via infinite signed-digit numerals in Section 2. We then investigate the connections between the lexicographical and numerical orders in Section 3. This is applied to obtain extremely simple constructions of the case-analysis operator for total and partial numerals in Section 4.

2 Computation via signed-digit numerals

For simplicity, in this paper we consider *binary* numerals. We first consider standard binary numerals and recall their main drawback from the point of view of computability (Section 2.1). We then briefly show how signed-digit binary numerals overcome the difficulties (Section 2.2) and define computability (Section 2.3). Finally, we discuss the issue of canonical forms (Section 2.4).

2.1 Standard numerals

A *numeral* is an infinite sequence over the digit alphabet $\mathbf{2} = \{0,1\}$. A numeral $\alpha \in \mathbf{2}^\omega$ denotes the number

$$[\![\alpha]\!] = \sum_{i \geqslant 0} \alpha_i \cdot 2^{-(i+1)} \in [0,1].$$

If one wants to extend binary notation to the whole real line, one can use a mantissa-exponent representation—see e.g. [9]. Since this adds little to our considerations, we restrict ourselves to numerals over a closed and bounded interval.

The map $\alpha \mapsto [\![\alpha]\!]$ is a surjection $q : \mathbf{2}^\omega \twoheadrightarrow [0,1]$. It is not an injection because the dyadic rationals $m/2^n \in (0,1)$ have two binary notations. We say that a function $\phi : \mathbf{2}^\omega \to \mathbf{2}^\omega$ *realizes* a function $f : [0,1] \to [0,1]$ if the following diagram commutes:

$$\begin{tikzcd} \mathbf{2}^\omega \arrow[r, "\phi"] \arrow[d, "q"'] & \mathbf{2}^\omega \arrow[d, "q"] \\ {[0,1]} \arrow[r, "f"'] & {[0,1].} \end{tikzcd}$$

Of course, here we are interested in *computable* realizers. A necessary condition for a function $\phi : \mathbf{2}^\omega \to \mathbf{2}^\omega$ being computable is that finite subsequences of $\phi(\alpha)$ depend only on finite subsequences of α. In this case we say that ϕ is of *finite character*.

Proposition 2.1 *A function $\phi : \mathbf{2}^\omega \to \mathbf{2}^\omega$ is of finite character iff it is continuous.*

Here $\mathbf{2}$ is endowed with the discrete topology and $\mathbf{2}^\omega$ with the product topology, so that $\mathbf{2}^\omega$ is Cantor space—see e.g. [22].

Proposition 2.2 *The surjection $q : \mathbf{2}^\omega \twoheadrightarrow [0,1]$ is a topological quotient map.*

Simply because it is continuous, and a continuous surjection of compact Hausdorff spaces is always a quotient map.

Corollary 2.3 *A function with a realizer of finite character is necessarily continuous.*

Unfortunately, the converse is not true.

Proposition 2.4 *There are continuous functions with no realizer of finite character, for example $f(x) = 3x/4$.*

See e.g. [29,10]. Corollary 2.3, Proposition 2.4 and Corollary 2.5 are essentially due to Brouwer [5].

Corollary 2.5 *Simple functions such as $f(x) = 3x/4$ fail to be computable in standard digital notation.*

This is of course independent of the choice of a base—but different counter-examples are needed for different bases. However, Brouwer showed that the use of the non-standard base 2/3 with digits 0 and 1 solves this problem. Turing [25] didn't notice the problem when he defined computable numbers. He corrected his definition in [24], adopting Brouwer's solution. Here we consider a variation of this solution, consisting of the use of the standard base 2 with negative digits.

2.2 Signed-digit numerals

A *signed-digit numeral* is an infinite sequence over the signed-digit alphabet $\mathbf{3} = \{\bar{1}, 0, 1\}$, where $\bar{1}$ stands for -1. A numeral $\alpha \in \mathbf{3}^\omega$ denotes the number

$$[\![\alpha]\!] = \sum_{i \geqslant 0} \alpha_i \cdot 2^{-(i+1)} \in [-1, 1].$$

As before, the surjection $\alpha \mapsto [\![\alpha]\!]$ is a quotient map $q : \mathbf{3}^\omega \twoheadrightarrow [-1, 1]$. The above definitions and facts for standard numerals apply to signed-digit numerals, except that now one has that, in contrast to Proposition 2.4,

Proposition 2.6 *Every continuous function $f : [-1, 1] \to [-1, 1]$ has a realizer $\phi : \mathbf{3}^\omega \to \mathbf{3}^\omega$ of finite character.*

The same holds for functions of several arguments, with realizers defined in the obvious way.

Proof. Müller [15], and Weihrauch and Kreitz [28], showed that the quotient map $q : \mathbf{3}^\omega \twoheadrightarrow [-1, 1]$ is *admissible* (or maximal) in the following sense. For every quotient map $q' : \mathbf{3}^\omega \twoheadrightarrow [-1, 1]$, there is a (far from unique) continuous map $t : \mathbf{3}^\omega \to \mathbf{3}^\omega$ which translates from q'-notation to q-notation, meaning that $q' = q \circ t$. The same argument shows that, more generally, for every continuous map $g : \mathbf{3}^\omega \to [-1, 1]$ there is a continuous map $\phi : \mathbf{3}^\omega \to \mathbf{3}^\omega$ such that $g = q \circ \phi$. (In other words, the space $\mathbf{3}^\omega$ is projective over the quotient map q.) Then the result follows by taking $g = f \circ q$. □

2.3 Computability

For the purposes of this paper, a function $f : [-1, 1] \to [-1, 1]$ is computable if it has a computable realizer $\phi : \mathbf{3}^\omega \to \mathbf{3}^\omega$. Computability on $\mathbf{3}^\omega$ can be defined e.g. via Turing machines with oracles [13,27,26]. In practice, an intuitive understanding of computability on $\mathbf{3}^\omega$ is enough for most purposes.

2.4 The lexicographical order and canonical forms

We order numerals by the lexicographical order

$\alpha < \beta$ iff there is an integer k such that $\alpha_k < \beta_k$ and $\alpha_i = \beta_i$ for each $i < k$.

Each number has two canonically associated signed-digit numerals:

Proposition 2.7 *Every number is denoted by a smallest and by a largest signed-digit numeral.*

Proof. The preimage of a point by the quotient map $q : \mathbf{3}^\omega \to [-1,1]$ is a closed set because q is continuous. But topologically closed subsets of $\mathbf{3}^\omega$ are closed under non-empty infima and suprema in the lexicographical order (in fact, this characterizes topologically closed subsets). □

However, there are no effectively determinable canonical forms:

Proposition 2.8 *There is no continuous, denotation-preserving idempotent map $c : \mathbf{3}^\omega \to \mathbf{3}^\omega$ such that $c(q^{-1}(\{x\}))$ is a singleton for each $x \in [-1,1]$.*

Proof. This would show that $[-1,1]$ is a retract of $\mathbf{3}^\omega$. But $\mathbf{3}^\omega$ is a totally disconnected space, and such spaces are closed under retracts. On the other hand, $[-1,1]$ is connected. □

3 Lexicographical order versus numerical order

If $\alpha, \beta \in \mathbf{2}^\omega$ are standard numerals then

$\alpha \leqslant \beta$ implies $[\![\alpha]\!] \leqslant [\![\beta]\!]$.

This property fails for signed-digit numerals (for example, for $\alpha = \bar{1}1^\omega$ and $\beta = 0\bar{1}^\omega$ one has $\alpha < \beta$ but $[\![\alpha]\!] = 0 \not\leqslant -1/2 = [\![\beta]\!]$). Moreover, its converse fails for both standard and signed-digit numerals (for example, for $\alpha = 10^\omega$ and $\beta = 01^\omega$ one has that $[\![\alpha]\!] = [\![\beta]\!] = 1/2$ but $\alpha \not\leqslant \beta$). However, it turns out that signed-digit numerals admit a very strong order-normalization property that standard numerals don't.

3.1 Order-normalization

Definition 3.1 *A pair (α, β) of numerals is* order-normal *iff*

(i) $\alpha < \beta$ *and* $[\![\alpha]\!] < [\![\beta]\!]$, *or*

(ii) $\alpha = \beta$, *or*

(iii) $\alpha > \beta$ *and* $[\![\alpha]\!] > [\![\beta]\!]$. □

Thus, for order-normal pairs (α, β), the condition $[\![\alpha]\!] = [\![\beta]\!]$ implies $\alpha = \beta$. In order to emphasize that order-normality is a property of pairs of numerals and not of single numerals, we observe that the pair (α, α) is always order-normal.

Theorem 3.2 *There is a computable, denotation-preserving idempotent map* $\mathrm{onorm} : \mathbf{3}^\omega \times \mathbf{3}^\omega \to \mathbf{3}^\omega \times \mathbf{3}^\omega$ *whose fixed-points are order-normal pairs.*

That is, $\mathrm{onorm}(\alpha, \beta) = (\alpha', \beta')$ implies that

(i) $([\![\alpha]\!], [\![\beta]\!]) = ([\![\alpha']\!], [\![\beta']\!])$,

(ii) (α', β') is order-normal, and

(iii) $\mathrm{onorm}(\alpha', \beta') = (\alpha', \beta')$.

We don't make use of the idempotency condition (iii) in our applications.

In order to prove Theorem 3.2, it is convenient to introduce a weak version of order-normality. A pair (α, β) of numerals is *head-normal* if

(i) $\alpha < \beta$ and $[\![\alpha]\!] < [\![\beta]\!]$, or

(ii) $\alpha_0 = \beta_0$, or

(iii) $\alpha > \beta$ and $[\![\alpha]\!] > [\![\beta]\!]$.

Lemma 3.3 *There is a computable, denotation-preserving idempotent map* $\mathrm{hnorm} : \mathbf{3}^\omega \times \mathbf{3}^\omega \to \mathbf{3}^\omega \times \mathbf{3}^\omega$ *whose fixed points are head-normal pairs.*

With this we can prove the order-normalization theorem.

Proof of Theorem 3.2. A recursive algorithm is given by

$$\begin{aligned} \mathrm{onorm}(\alpha, \beta) &= \mathrm{onorm}'(\mathrm{hnorm}(\alpha, \beta)), \\ \mathrm{onorm}'(\alpha, \beta) &= (\alpha, \beta) \text{ if } \alpha_0 \neq \beta_0, \\ \mathrm{onorm}'(d\alpha, d\beta) &= (d\alpha', d\beta') \text{ where } (\alpha', \beta') = \mathrm{onorm}'(\mathrm{hnorm}(\alpha, \beta)). \end{aligned}$$

The idea is that the input of onorm′, when onorm′ is called from onorm or itself, is always head-normal. If $[\![\alpha]\!] = [\![\beta]\!]$ then the last equation is always applied. Otherwise we eventually reach the previous equation and the process comes to an end. □

In order to prove the head-normalization lemma, let $\equiv$ be the equivalence relation on numerals induced by the denotation function:

$\alpha \equiv \beta$ iff $[\![\alpha]\!] = [\![\beta]\!]$.

We state the following two easily established facts as lemmas:

Lemma 3.4 *The following identities hold for all $\alpha \in \mathbf{3}^\star$ and $\beta \in \mathbf{3}^\omega$:*

$\alpha 0\bar{1}\beta \equiv \alpha\bar{1}1\beta, \qquad \alpha 01\beta \equiv \alpha 1\bar{1}\beta.$

Here $\mathbf{3}^\star$ is the set of finite sequences over $\mathbf{3}$. Notice that these identities are symmetric, in that each one arises from the other by multiplying digits by -1.

Lemma 3.5 *For all $x \in [-1, 1]$, $d \in \mathbf{3}$ and $\alpha \in \mathbf{3}^\omega$,*

(i) $x \leqslant [\![\alpha]\!]$ *implies* $(d + x)/2 \leqslant [\![d\alpha]\!]$,

(ii) $[\![\alpha]\!] \leqslant x$ *implies* $[\![d\alpha]\!] \leqslant (d + x)/2$.

In practice, we successively apply these entailments to the inequality $-1 \leqslant [\![\alpha]\!] \leqslant 1$ after examining finitely many digits of α.

Proof of Lemma 3.3. In order to significantly cut down the number of cases in the definition of hnorm, given below, it is convenient to add the following natural conditions to its specification: If $\mathrm{hnorm}(\alpha, \beta) = (\alpha', \beta')$ then

(i) $\mathrm{hnorm}(\beta, \alpha) = (\beta', \alpha')$ (symmetry by twisting), and

(ii) $\mathrm{hnorm}(-\alpha, -\beta) = (-\alpha', -\beta')$ (symmetry by reflection).

Here $-\alpha$ is the digit-wise complement (multiplication by -1) of the numeral 'α, which of course realizes the complement function on numbers.

We let $\mathrm{hnorm}(\alpha, \beta)$ be (α', β'), where α' and β' are defined by cases as follows:

(i) $\alpha_0 = \beta_0$: We can let (α', β') be (α, β) because the latter is head-normal.

(ii) $\alpha_0 = \bar{1}$ and $\beta_0 = 0$: Then $[\![\alpha]\!] \leqslant 0$ and $-1/2 \leqslant [\![\beta]\!]$.

(a) $\alpha_1 = \bar{1}$: Then $[\![\alpha]\!] \leqslant -1/2$.

- $\alpha_2 \in \{\bar{1}, 0\}$ or $\beta_1 \in \{0, 1\}$: Then $[\![\alpha]\!] < -1/2$ or $-1/2 < [\![\beta]\!]$. Hence $[\![\alpha]\!] < [\![\beta]\!]$. Thus (α, β) is head-normal because $\alpha < \beta$. Therefore we can let (α', β') be (α, β).
- $\alpha_2 = 1$ and $\beta_1 = \bar{1}$: In order to get α' and β' we apply the identities of Lemma 3.4 to β so that α' and β' share the first digit and hence are head-normal: $\alpha = \bar{1}\bar{1}1\cdots =: \alpha'$ and $\beta = 0\bar{1}\cdots \equiv \bar{1}1\cdots =: \beta'$.

(b) $\alpha_1 = 0$: Then $[\![\alpha]\!] \leqslant -1/4$.

- $\beta_1 = \bar{1}$: $\alpha = \bar{1}0\cdots =: \alpha'$ and $\beta = 0\bar{1}\cdots \equiv \bar{1}1\cdots =: \beta'$.
- $\alpha_2 = 1$ (and $\beta_1 \neq \bar{1}$): $\alpha = \bar{1}01\cdots \equiv \bar{1}1\bar{1}\cdots \equiv 0\bar{1}\bar{1}\cdots =: \alpha'$ and $\beta = 0\cdots =: \beta'$.
- $\alpha_2 \in \{\bar{1}, 0\}$ and $\beta_1 \in \{0, 1\}$: Then $[\![\alpha]\!] < -1/4 \leqslant [\![\beta]\!]$. Hence we can take $(\alpha', \beta') = (\alpha, \beta)$.

(c) $\alpha_1 = 1$: $\alpha = \bar{1}1\cdots \equiv 0\bar{1}\cdots =: \alpha'$ and $\beta = 0\cdots =: \beta'$.

(iii) $\alpha_0 = \bar{1}$ and $\beta_0 = 1$: Then $[\![\alpha]\!] \leqslant 0 \leqslant [\![\beta]\!]$.

(a) $\alpha_1 \in \{\bar{1}, 0\}$ or $\beta_1 \in \{0, 1\}$: Then $[\![\alpha]\!] < 0$ or $0 < [\![\beta]\!]$. Hence we can take $(\alpha', \beta') = (\alpha, \beta)$.

(b) $\alpha_1 = 1$ and $\beta_1 = \bar{1}$: We can take $\alpha = \bar{1}1\cdots \equiv 0\bar{1}\cdots =: \alpha'$ and $\beta = 1\bar{1}\cdots \equiv 01\cdots =: \beta'$.

(iv) $\alpha_0 = 0$ and $\beta_0 = \bar{1}$: Symmetric to case (ii) by twisting.

(v) $\alpha_0 = 0$ and $\beta_0 = 1$: Symmetric to case (iv) by reflection.

(vi) $\alpha_0 = 1$ and $\beta_0 = \bar{1}$: Symmetric to case (iii) by twisting.

(vii) $\alpha_0 = 1$ and $\beta_0 = 0$: Symmetric to case (v) by twisting.

By construction, hnorm is denotation-preserving and its range consists of head-normal pairs. A simple inspection shows that it is also idempotent. (Notice that we need to look to at most three digits of each input of the head-normalization operator in order to generate the whole output.) $\square$

This shows that denotational equivalence, for *signed-digit* numerals, is a relation of finite character. Let $\sim$ be the least equivalence relation such that

$$\alpha 0\bar{1}\beta \sim \alpha\bar{1}1\beta, \qquad \alpha 01\beta \sim \alpha 1\bar{1}\beta.$$

It follows from Lemma 3.4 that $\alpha \sim \beta$ implies $\alpha \equiv \beta$. Of course, the converse is not true. However, a weak form of the converse follows from Lemma 3.3.

Corollary 3.6 *If $\alpha \equiv \beta$ then for every natural number k there are $\alpha' \sim \alpha$ and $\beta' \sim \beta$ with $\alpha'_i = \beta'_i$ for each $i < k$.*

3.2 Weak order-normalization

The following weak form of order-normalization is enough for many purposes. We say that a pair (α, β) of numerals is *weakly order-normal* iff

$\alpha \leqslant \beta$ and $[\![\alpha]\!] \leqslant [\![\beta]\!]$, or $\alpha \geqslant \beta$ and $[\![\alpha]\!] \geqslant [\![\beta]\!]$.

As above, in order to obtain a weak normalization operator, it is enough to consider the following (further) weakening. A pair (α, β) of numerals is *weakly head-normal* if

$\alpha \leqslant \beta$ and $[\![\alpha]\!] \leqslant [\![\beta]\!]$, or $\alpha_0 = \beta_0$, or $\alpha \geqslant \beta$ and $[\![\alpha]\!] \geqslant [\![\beta]\!]$.

A simplification of the construction given in Lemma 3.3 produces a more efficient computable, denotation-preserving idempotent map whnorm : $\mathbf{3}^\omega \times \mathbf{3}^\omega \to \mathbf{3}^\omega \times \mathbf{3}^\omega$ whose fixed points are weakly head-normal pairs. We let whnorm(α, β) be (α', β'), where α' and β' are defined by cases as follows:

(i) $\alpha_0 = \beta_0$: Take $(\alpha', \beta') = (\alpha, \beta)$.

(ii) $\alpha_0 = \bar{1}$ and $\beta_0 = 0$: Then $[\![\alpha]\!] \leqslant 0$ and $-1/2 \leqslant [\![\beta]\!]$.

- (a) $\alpha_1 = \bar{1}$: Then $[\![\alpha]\!] \leqslant -1/2$. Hence (α, β) is weakly head-normal because $\alpha < \beta$ and $[\![\alpha]\!] \leqslant [\![\beta]\!]$. We can thus take $(\alpha', \beta') = (\alpha, \beta)$.
- (b) $\alpha_1 = 0$: Then $[\![\alpha]\!] \leqslant -1/4$.
  - $\beta_1 = \bar{1}$: $\alpha = \bar{1}0 \cdots =: \alpha'$ and $\beta = 0\bar{1} \cdots \equiv \bar{1}1 \cdots =: \beta'$.
  - $\beta_1 \in \{0, 1\}$: Then $1/4 \leq [\![\beta]\!]$ and $\alpha < \beta$, and hence we can take $(\alpha', \beta') = (\alpha, \beta)$.

(iii) $\alpha_0 = \bar{1}$ and $\beta_0 = 1$: Then $[\![\alpha]\!] \leqslant 0 \leqslant [\![\beta]\!]$ and we can take $(\alpha', \beta') = (\alpha, \beta)$.

The remaining cases are symmetric to the others by reflection or twisting as in the proof of Lemma 3.3. From this construction we see that, for the weak head-normalization operator, we only need to look to at most two digits of each input in order to generate the whole output.

4 Definition by cases on the reals

As a first application of order-normalization, we obtain a simple proof of an essentially well-known result.

Proposition 4.1 *There is a computable partial function* cases : $[-1, 1]^4 \rightharpoonup [-1, 1]$ *such that*

$$\mathrm{cases}(x, t, y, z) = \begin{cases} y & \textit{if } x < t \textit{ or } y = z, \\ z & \textit{if } x > t \textit{ or } y = z, \end{cases}$$

with domain of definition $\{(x, t, y, z) \mid x = t \textit{ implies } y = z\}$.

Proof. We first consider an analogous lexicographical case-analysis operator lexcases : $(\mathbf{3}^\omega)^4 \rightharpoonup \mathbf{3}^\omega$ such that

$$\text{lexcases}(\alpha,\beta,\gamma,\delta) = \begin{cases} \gamma & \text{if } \alpha < \beta \text{ or } \gamma = \delta, \\ \delta & \text{if } \alpha > \beta \text{ or } \gamma = \delta, \end{cases}$$

with domain of definition $\{(\alpha,\beta,\gamma,\delta) \mid \alpha = \beta \text{ implies } \gamma = \delta\}$. In order to compute lexcases$(\alpha,\beta,\gamma,\delta)$, we can first output the greatest common prefix of γ and δ. If it is finite then $\gamma \neq \delta$ and hence we must have $\alpha < \beta$ or $\alpha > \beta$. In the first case we output the remainder of γ and in the second the remainder of δ. Now, a realizer for the numerical case-analysis operator is given by $\phi(\alpha,\beta,\gamma,\delta) = \text{lexcases}(\alpha',\beta',\gamma',\delta')$ where $(\alpha',\beta') = \text{onorm}(\alpha,\beta)$ and $(\gamma',\delta') = \text{onorm}(\gamma,\delta)$. (Notice that the first occurrence of the the order-normalization operator can be replaced by the more efficient weak order-normalization operator, but that the second cannot.) □

We now work with the domain $\mathbf{3}^\infty = \mathbf{3}^\star \cup \mathbf{3}^\omega$ of finite and infinite numerals ordered by prefix. The idea here is that finite numerals denote processes that produce finitely many digits and then diverge. In order to emphasize this point of view, finite numerals will be referred to as *partial numerals*. If one wishes allow processes that produce finitely many digits and then terminate, one can consider a termination symbol—but this possibility is outside the scope of the present discussion. In this framework, partial functions are modelled by total functions which map some total numerals to partial numerals.

If an index i is out of range in an expression such as α_i, we adopt the convention that the value of the expression is $\perp$. This is in accordance with the interpretation of finite numerals as partial numerals. We also adopt the notation $\alpha^{(n)}$ defined by $(\alpha^{(n)})_i = \alpha_{n+i}$. In particular, $\alpha^{(1)}$ is the "tail" of α.

Lemma 4.2 *The order-normalization operator constructed in Theorem 3.2 has a sequentially computable extension* $\mathbf{3}^\infty \times \mathbf{3}^\infty \to \mathbf{3}^\infty \times \mathbf{3}^\infty$.

We are not particularly interested in the precise behaviour of the operator at partial numerals. The same is true for the operators defined below. What is important here is that it is extended in such a way that one obtains a sequentially computable function.

Proof. The recursive definition of the order-normalization operator given in the proof of Theorem 3.2 clearly applies to the case in which one admits partial numerals, and it is manifestly sequential. The same applies to the construction of the auxiliary function hnorm defined in the proof of Lemma 3.3, as it consists of a case analysis which examines digits of the first and second argument in an alternated and hence sequential fashion. □

Let $\mathcal{B}$ be the flat domain $\{\text{true}, \text{false}\}_\perp$ of truth values.

Lemma 4.3 *There is a sequentially computable function* $(\alpha,\beta) \mapsto (\alpha <_\perp \beta)$:

$\mathbf{3}^\infty \times \mathbf{3}^\infty \to \mathcal{B}$ *such that for all* total *numerals* α *and* β,

$$(\alpha <_\perp \beta) = \begin{cases} \text{true} & \textit{if } [\![\alpha]\!] < [\![\beta]\!], \\ \text{false} & \textit{if } [\![\alpha]\!] > [\![\beta]\!]. \end{cases}$$

Proof. Since we can first apply the sequential order-normalization operator, it is enough to define a sequential lexicographical inequality relation:

$$(\alpha <_\perp \beta) = \text{if } (\alpha_0 = \beta_0) \text{ then } \alpha^{(1)} <_\perp \beta^{(1)} \text{ else } \alpha_0 < \beta_0. \quad \square$$

Lemma 4.4 *There is a sequentially computable map* ppif : $\mathcal{B} \times \mathbf{3}^\infty \times \mathbf{3}^\infty \to \mathbf{3}^\infty$ *such that for all* total *numerals* α *and* β,

$$[\![\mathrm{ppif}(p, \alpha, \beta)]\!] = \begin{cases} [\![\alpha]\!] & \textit{if } p = \text{true } \textit{or } [\![\alpha]\!] = [\![\beta]\!], \\ [\![\beta]\!] & \textit{if } p = \text{false } \textit{or } [\![\alpha]\!] = [\![\beta]\!]. \end{cases}$$

Proof. Since we can first order-normalize (α, β), we can assume w.l.o.g. that $[\![\alpha]\!] = [\![\beta]\!]$ iff $\alpha = \beta$, and hence define ppif by

$$\mathrm{ppif}(p, \alpha, \beta) = \mathrm{if}(\alpha_0 = \beta_0)\,\mathrm{then}\,\alpha_0 \cdot \mathrm{ppif}\,(p, \alpha^{(1)}, \beta^{(1)})\,\mathrm{else\,if}\,p\,\mathrm{then}\,\alpha\,\mathrm{else}\,\beta. \quad \square$$

Corollary 4.5 *A sequentially computable extension of the the numerical case-analysis operator to partial numerals can be defined by*

$$\mathrm{cases}(x, t, y, z) = \mathrm{ppif}\;\; x <_\perp t \;\; \mathrm{then} \;\; y \;\; \mathrm{else} \;\; z.$$

5 Concluding remarks

We developed a general effective and sequential cases-analysis operator for which the cases consist of (in)equalities between real numbers. The construction is based on order-normalization, a partial inequality relation, and a pseudo-parallel conditional. We have to admit that the pseudo-parallel conditional is probably as inefficient as the truly parallel conditional, because both branches have to be evaluated and compared always. However, parallelism (or dovetailing) is completely avoided, so that implementations in sequential programming languages are possible.

The approach to real number computation via signed-digit numerals is highly *intensional*. The author conjectures that parallelism is unavoidable in *extensional* domain-theoretic approaches to real number computation such as the one investigated in [7].

The results on order-normalization used to obtain the case-analysis operator are interesting in their own right. In particular, they show that one can always assume without loss of generality that any two numerals that denote the same number are themselves the same.

Acknowledgement
I benefited from remarks by Alex Simpson on a previous version of this paper.

References

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

[2] A. Avizienis. Binary-compatible signed-digit arithmetic. In *AFIPS Conference Proceedings*, volume 1 of *26*, pages 663–672, 1964.

[3] H.J. Boehm and R. Cartwright. Exact real arithmetic: Formulating real numbers as functions. In Turner. D., editor, *Research Topics in Functional Programming*, pages 43–64. Addison-Wesley, 1990.

[4] H.J. Boehm, R. Cartwright, M. Riggle, and M.J. O'Donnel. Exact real arithmetic: A case study in higher order programming. In *ACM Symposium on Lisp and Functional Programming*, 1986.

[5] L.E.J. Brouwer. Besitzt jede reelle Zahl eine Dezimalbruchentwicklung? *Math Ann*, 83:201–210, 1920.

[6] A. Cauchy. Sur les moyens d'éviter les erreurs dans les calculs numériques. *Comptes Rendus 11*, pages 789–798, Paris 1840. Republished in: Augustin Cauchy, Œvres complètes, 1ére série, Tome V, pp 431–442.

[7] M.H. Escardó. *PCF extended with real numbers: A domain-theoretic approach to higher-order exact real number computation.* PhD thesis, Imperial College of the University of London, November 1996. Technical Report ECS-LFCS-97-374, Department of Computer Science, University of Edinburgh. Available at `http://www.dcs.ed.ac.uk/lfcsreps/EXPORT/97/ECS-LFCS-97-374/index.html`.

[8] P. Di Gianantonio. *A functional approach to computability on real numbers.* PhD thesis, University of Pisa, 1993. Technical Report TD 6/93.

[9] P. Di Gianantonio. A golden ratio notation for the real numbers. Technical Report CS-R9602, CWI Amsterdam, 1996.

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

[11] C.A. Gunter. *Semantics of Programming Languages—Structures and Techniques.* The MIT Press, London, 1992.

[12] S.C. Kleene. *Introduction to Metamathematics.* North-Holland, Amsterdam, 1952.

[13] Ker-I Ko. *Complexitity Theory of Real Functions.* Birkhauser, Boston, 1991.

[14] J. Leslie. *The Philosophy of Arithmetic.* Edinburgh, 1817.

[15] N.Th. Müller. Subpolynomial complexity classes of real functions and real numbers. In Laurent Kott, editor, *Proceedings of the 13th International Colloquium on Automata, Languages, and Programming*, volume 226 of *Lecture Notes in Computer Science*, pages 284–293, Berlin, 1986. Springer.

[16] G.D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(1):223–255, 1977.

[17] G.D. Plotkin. $\mathbb{T}^\omega$ as a universal domain. *Journal of Computer and System Sciences*, 17:209–236, 1978.

[18] G.D. Plotkin. Domains. Post-graduate lectures in advanced domain theory, Department of Computer Science, University of Edinburgh. Available at http://hypatia.dcs.qmw.ac.uk/sites/other/domain.notes.other, 1983.

[19] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[20] D. S. Scott. A type-theoretical alternative to CUCH, ISWIM and OWHY. *Theoretical Computer Science*, 121:411–440, 1993. Reprint of a manuscript produced in 1969.

[21] A. Simpson. Lazy functional algorithms for exact real functionals. In *Mathematical Foundations of Computer Science 1998.* To appear. Available at `http://www.dcs.ed.ac.uk/home/als/Research/`.

[22] M.B. Smyth. Topology. In S. Abramsky, D. M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 641–761. Clarendon Press, Oxford, 1992.

[23] Ph. Sünderhauf. *Discrete Approximation of Spaces.* PhD thesis, Technishe Hochschule Darmstadt, 1994.

[24] A. Turing. A correction. *The London Mathematical Society*, 43:544–546, 1937.

[25] A. Turing. On computable numbers, with an application to the Entscheindungproblem. *The London Mathematical Society*, 42:230–265, 1937.

[26] K. Weihrauch. A simple introduction to computable analysis. Technical Report 171–7/1995, FernUniversität, 1995.

[27] K. Weihrauch. A foundation for computable analysis. In Douglas S. Bridges, Cristian S. Calude, Jeremy Gibbons, Steve Reeves, and Ian H. Witten, editors, *Combinatorics, Complexity, and Logic*, Discrete Mathematics and Theoretical Computer Science, pages 66–89, Singapore, 1997. Springer. Proceedings of DMTCS'96.

[28] K. Weihrauch and C. Kreitz. Representations of the real numbers and the open subsets of the set of real numbers. *Annals of Pure and Applied Logic*, 35:247–260, 1987.

[29] E. Wiedmer. Computing with infinite objects. *Theoretical Computer Science*, 10:133–155, 1980.