Computing with Exact Real Numbers in a Radix-r System[1]

Alexander Kaganovsky

15 October, 1999

[1]This report is a revised version of the paper presented at the Birmingham COMPROX Workshop (1997) which appears in [7].

Abstract

This paper investigates an arithmetic based upon the representation of computable exact real numbers by lazy infinite sequences of signed digits in a positional radix-r system. We discuss advantages and problems associated with this representation, and develop well-behaved algorithms for a comprehensive range of numeric operations, including the four basic operations of arithmetic.

1 Introduction

The standard implementations of real numbers on a computer are approximately held to some fixed number of significant figures. The accumulation of rounding errors leads to well-known difficulties calculating accurate numerical results for scientific and engineering problems. Going to double, quadruple or even multiple precision in no way eliminates these problems, but merely ameliorates them. No matter how much precision is provided, there are always problems for which it is insufficient to produce reliable results. Perhaps one of the worst features of floating point arithmetic is that the computer can give us no indication of how many of the digits printed are actually meaningful, so with a poor choice of algorithm it is quite easy to generate numerical answers that are completely meaningless. An illustrative example of such rounding anomalies is given in [9] (also in [8]), where computation of a simple function on single precision floating point is shown to produce completely wrong results after only 14 divisions and 12 subtractions.

As computing power becomes cheaper, it seems reasonable that we may wish to move to a form of real arithmetic that is perhaps more expensive but which will generate results to numerical calculations that carry with them some easily understood guarantee of accuracy. Modern programming languages provide certain computing abstractions — infinite lists, higher order functions — which make it possible to represent real numbers exactly as they are defined in mathematics, using any of several possible methods.

Mathematically a real number is defined as an infinitary object — for example, a converging sequence of rationals. Since all our computers are finite, it stands to reason that only finitely many entries of an infinite sequence can be instantiated in finite time. It also follows that not all real numbers can be represented on a computer — only those whose defining sequence can be determined by a *finite* amount of information.

The concept of a *computable real number* was first introduced by Alan Turing in his classical paper [16]. He defined a computable real as one whose decimal[1] expansion can be written down by a Turing machine. Roughly speaking, a real number is computable if there exists a finite computer program that can effectively approximate it to any degree of precision. When more precision is desired, the computation may take longer, but the program itself does not change. Herefrom, it follows that not all real numbers are computable; at least, because the set of all finite computer programs is countable, whereas the set of all real numbers is not.

In classical mathematical analysis, real numbers are defined in a variety

[1] Any other radix $r > 1$ could be used in exactly the same way.

of ways, all of which are equivalent to each other, so that the choice of a particular representation is matter of convenience. In constructive mathematics, however, some functions on the computable reals, and even the four basic arithmetic operations, are critically dependent on the representation, and with a poor choice of the latter may become non-computable. For instance, conventional fixed-radix positional weighted number systems, for which the weight of the i-th digit is r^{-i} and the range of each digit is $\{0, 1, \cdots, r-1\}$, appear to be unsuitable for exact computations, because it is sometimes impossible to compute even the first digit of a result without having to inspect an infinite number of the operands' digits. A specific example can be found e.g. in [10].

One of the pioneer investigators of this problem was Wiedmer who suggested the use of *redundant signed-digit systems* to effect computability [20]. Although signed-digit notation was proposed as a means of avoiding carry propagation chains in hardware arithmetic as early as in 1960's, and has been well known among hardware designers, having led to the development of *digit-pipelined* or *on-line* arithmetic [6], it was probably Wiedmer who first suggested its use in the context of exact computations. His PhD thesis [21] contains a detailed investigation of the algorithms necessary for exact real arithmetic on redundant signed-digit sequences.

In 1981-2, Carl Pixley at Burroughs Corporation undertook a study of Wiedmer's work, implementing a complete package of functions for exact real arithmetic in the lazy functional language SASL [18]. Pixley spent some time analyzing the efficiency of the algorithms, in particular for division, which is the most subtle of the four basic operations. Although never formally published, Pixley's work [11] was privately circulated and stimulated interest in the topic.

In 1986, Boehm, Cartwright, et al. [3] reported their two implementations of exact real arithmetic — as lazy infinite sequences of decimal digits, and as functions mapping rational errors to rational approximations. Having carried out a comparative study of the two methods, they contended that the lazy sequence method led to unsatisfactory implementations and performed very poorly, while the functional method performed surprisingly well. Their claim was partially based on what they called "the granularity effect" — computation of arguments to one digit's more accuracy than necessary, which makes the evaluation of expressions such as $x_1 + (x_2 + (x_3 + (\cdots + x_n)))$ highly inefficient. Since then, an extensive literature has arisen devoted to representations of exact reals [4, 5, 8, 13, 19]; yet, no further attempts have been made to find out whether the claimed advantage of functions over lazy lists of digits was simply an artifact of a particular class of implementations of lazy languages, or evidence of something more fundamental.

The purpose of this paper is to investigate the properties of the redundant signed-digit representation of the reals, and find whether it can be rendered free from the objections which have caused its rejection by the majority of the researchers, who have deserted altogether its line of approach. In so doing, we develop algorithms for a wide range of numerical operations, including the four basic operations of arithmetic, discuss the complexity issues, and examine various factors that can affect implementations.

2 Radix-r redundant signed-digit expansions

A number system is said to be *redundant* if there are at least two distinct representations that are mapped onto the same number; otherwise, it is *non-redundant.* A radix r number system requires *at least* r digit symbols; if this number is greater than r, the system becomes redundant.

The following variation of the fixed-radix number system was originally used by Avizienis [1, 2] to eliminate carry propagation chains in addition and subtraction.

Definition 1. *A radix-r redundant signed-digit (SD) number system is one based on a digit set*

$$S_\rho = \left\{\overline{\rho}, \dots, \overline{1}, 0, 1, \dots, \rho\right\},$$

where $\overline{x}$ denotes $-x$, $1 \leq \rho \leq r-1$, and $\rho \geq r/2$.

The last condition allows each digit to assume more than r values and thus gives rise to the redundancy. We can measure the degree of redundancy of a given SD system by calculating the *redundancy coefficient*

$$\mu(S_\rho) = \frac{\rho}{r-1}.$$

A digit set is said to be *maximally* or *minimally redundant* if its redundancy coefficient is maximal or minimal for the associated radix. Thus, for radix-10, the digit set $\left\{\overline{5}, \dots, \overline{1}, 0, 1, \dots, 5\right\}$ is minimally redundant, while $\left\{\overline{9}, \dots, \overline{1}, 0, 1, \dots, 9\right\}$ is maximally redundant.

Throughout this paper, we shall use the symbols $\mathbb{N}$ and $\mathbb{N}_0$ to denote the sets of all *positive* and *non-negative* integers respectively.

If $x \in \mathbb{R}$ is a real number, $r > 1$ an integer, and $(x_i)_{i\in\mathbb{N}_0}$ a sequence of integers with $-\rho \leq x_i \leq \rho$ for all $i \in \mathbb{N}$ such that

$$x = \sum_{i=0}^{\infty} x_i r^{-i},$$

then the symbol on the right side of

$$x = (x_0, x_1, x_2, \dots, x_n, \dots)_r \tag{1}$$

is called an *infinite radix-r redundant signed-digit expansion* for x. If $x_i = 0$ for all $i > p \geq 1$, we also write

$$x = (x_0, x_1, x_2, \dots, x_p)_r$$

This is a *finite* or *terminating radix-r expansion* for x. In case $r = 10$ these are called decimal signed-digit expansions and the subscript 10 is omitted.

If we allow the first digit of signed-digit expansions to be unbounded, $x_0 \in \mathbb{Z}$, then for every real number x there exist an infinite number of different radix-r redundant signed-digit expansions of the form (1). How are all these expansions related to each other? In order to answer this question, we shall introduce a few concepts and definitions.

Definition 2. *Let $(a_n)_{n\in\mathbb{N}_0}$ be a sequence of integers such that the series*

$$\sum_{n=0}^{\infty} a_n r^{-n} \tag{2}$$

is convergent. A sequence of integers $(b_n)_{n\in\mathbb{N}_0}$ is said to be equivalent to $(a_n)_{n\in\mathbb{N}_0}$ if

$$\sum_{n=0}^{\infty} a_n r^{-n} = \sum_{n=0}^{\infty} b_n r^{-n}$$

(and, in particular, the series on the right is also convergent). To indicate the equivalence of two sequences, we shall use the symbol $\sim$.

If we denote by S the set of all integer sequences $(a_n)_{n\in\mathbb{N}_0}$ for which the series (2) converges, then obviously $\sim$ is an equivalence relation on S, and using the fact that for any number $x \in \mathbb{R}$ there exists at least one expansion of the form (2), the equivalence classes are in one-to-one correspondence with the reals: $\mathbb{R} = S/\sim$.

We next define a family of functions $f : S \to S$ such that $f(s) \sim s$ for all $s \in S$.

Definition 3. *Let $i \in \mathbb{Z}$ be an integer, $i \neq 0$. We define*

$$f_0\left((a_n)_{n\in\mathbb{N}_0}\right) = (a_n)_{n\in\mathbb{N}_0}$$

$$f_i\left((a_n)_{n\in\mathbb{N}_0}\right) = (b_n)_{n\in\mathbb{N}_0}, \text{ where } b_j = \begin{cases} a_j + \operatorname{sgn}(i), & \text{if } j = |i| - 1 \\ a_j - \operatorname{sgn}(i) \cdot r, & \text{if } j = |i| \\ a_j, & \text{otherwise.} \end{cases}$$

Now let $(i_k)_{k=1}^m$ *be a finite sequence of integers. We define*

$$f_{i_1 i_2 \cdots i_m} \stackrel{def}{=} f_{i_m} \circ \ldots \circ f_{i_2} \circ f_{i_1}$$

For example, if $r = 10$, $(a_n)_{n \in \mathbb{N}_0} = (5, 5, \ldots 5, \ldots)$, we have

$$\begin{aligned} f_1\,(5,5,5,\ldots 5,\ldots) &= (6,-5,5,\ldots,5,\ldots) \\ f_{-1}\,(5,5,5,\ldots 5,\ldots) &= (4,15,5,\ldots,5,\ldots) \\ f_{1,2}\,(5,5,5,\ldots 5,\ldots) &= (6,-4,-5,5,\ldots,5,\ldots) \\ f_{-1,3}\,(5,5,5,\ldots 5,\ldots) &= (4,15,6,-5,\ldots,5,\ldots) \end{aligned}$$

One can see that the n-th element of a sequence can only be changed by $f_{\pm n}$ and $f_{\pm(n+1)}$, so our next step is to carry over the definition of $f_{(i_k)}$ to the case where (i_k) is an infinite sequence.

Definition 4. *Let* $(i_k)_{k \in \mathbb{N}}$, $i_k \neq 0$ *be an unbounded sequence of integers such that the sequence* $(|i_k|)_{k \in \mathbb{N}}$ *is nondecreasing. We then define*

$$f_{(i_k)_{k \in \mathbb{N}}}\left((a_n)_{n \in \mathbb{N}_0}\right) \stackrel{def}{=} (b_n)_{n \in \mathbb{N}_0},$$

where $b_n = \left(f_{i_1 \cdots i_{j_n}}\left((a_n)_{n \in \mathbb{N}_0}\right)\right)_n$ *and* $(j_n)_{n \in \mathbb{N}_0}$ *is any sequence of natural numbers with* $|i_{j_n - 1}| \leq n < |i_{j_n}|$ *(it is easy to verify that the value of* b_n *does not depend on the choice of* $(j_n)_{n \in \mathbb{N}_0}$*).*

Since f_0 is the identity function, we can also allow zeros to appear in the sequence $(i_k)_{k \in \mathbb{N}}$ by agreeing to calculate the value of $f_{(i_k)_{k \in \mathbb{N}}}$ as

$$f_{(i_k)_{k \in \mathbb{N}}} \stackrel{def}{=} f_{(i'_k)_{k \in \mathbb{N}}}$$

where the sequence $(i'_k)_{k \in \mathbb{N}}$ is obtained from the original sequence $(i_k)_{k \in \mathbb{N}}$ by skipping all encountered zeros.

One of the main properties of the functions $f_{(i_k)}$ is that they do not take us out of the equivalence classes with respect to $\sim$, i.e. for any $(a_n)_{n \in \mathbb{N}_0} \in S$

$$\begin{aligned} f_{i_1 i_2 \cdots i_m}\left((a_n)_{n \in \mathbb{N}_0}\right) &\sim (a_n)_{n \in \mathbb{N}_0} \\ f_{(i_k)_{k \in \mathbb{N}}}\left((a_n)_{n \in \mathbb{N}_0}\right) &\sim (a_n)_{n \in \mathbb{N}_0} \end{aligned}$$

Among other properties, we can indicate that

$$f_{m,-m} = f_{-m,m} = f_0 \text{ for all } m \in \mathbb{N}$$

Theorem 1. *Let $r \in \mathbb{N}$, $r > 1$ be a radix value, ρ be an integer with $1 \leq r/2 \leq \rho \leq r-1$, and let $x \in \mathbb{R}$. Then there exists a sequence $(a_n)_{n\in\mathbb{N}_0}$ of integers such that $-\rho \leq a_n \leq \rho$ for all $n \in \mathbb{N}$, and*

$$x = \sum_{n=0}^{\infty} a_n r^{-n}$$

Moreover, if $(b_n)_{n\in\mathbb{N}_0}$ is any other ($a_j \neq b_j$ for some j) sequence of integers such that $-\rho \leq b_n \leq \rho$ for all $n \in \mathbb{N}$; $b_n \neq r-1$ (or $b_n \neq -r+1$) for infinitely many n (if $\rho = r-1$), and

$$x = \sum_{n=0}^{\infty} b_n r^{-n}, \tag{3}$$

then there exists a (possibly finite) integer sequence $(i_k)_{k\in\mathbb{N}}$ such that $(|i_k|)_{k\in\mathbb{N}}$ is nondecreasing and $(b_n)_{n\in\mathbb{N}_0} = f_{(i_k)_{k\in\mathbb{N}}}\left((a_n)_{n\in\mathbb{N}_0}\right)$.

Proof. Let $x_0.x_1x_2\ldots x_n\ldots$ be the conventional radix-r expansion of x, i.e. $x_0 \in \mathbb{Z}$, $0 \leq x_i < r$ for all $i \in \mathbb{N}$. Then we define

$$(a_n)_{n\in\mathbb{N}_0} = f_{(i_n)_{n\in\mathbb{N}}}\left((x_n)_{n\in\mathbb{N}_0}\right)$$

where

$$i_n = \begin{cases} n, & \text{if} \quad \rho \leq x_n \leq r-1 \\ 0, & \text{if} \quad 0 \leq x_n \leq \rho - 1 \end{cases}$$

It is quite easy to see that $|a_n| \leq \rho$ for all $n \in \mathbb{N}$: we know that $0 \leq x_n < r$, and a_n is obtained from x_n through application of $f_{i_1\ldots i_k}$ for some $i_1,\ldots i_k$. Thus, if $x_n \in [0, \rho-1]$, it may only be changed by f_{n+1}, in which case it will be increased by 1; if $x_n \in [\rho, r-1]$, then f_n will reduce its value by r, and the resulting value may, in its turn, be also increased by 1 by f_{n+1}. In either case, we have $-\rho \leq a_n \leq \rho$, and $x = \sum_{n=0}^{\infty} a_n r^{-n}$.

Now suppose that (3) obtains for some sequence $(b_n)_{n\in\mathbb{N}_0}$ of integers where $|b_n| \leq \rho$ for all $n \in \mathbb{N}$, and $b_j \neq a_j$ for some j. Let $k = \inf\{j \in \mathbb{N} : a_j \neq b_j\}$, then we have

$$\sum_{n=k}^{\infty} a_n r^{-n} = \sum_{n=k}^{\infty} b_n r^{-n}$$

or

$$b_k = a_k + \sum_{n=1}^{\infty} (a_{n+k} - b_{n+k})\, r^{-n}$$

Since $|a_{n+k}| \le \rho$, $|b_{n+k}| \le \rho$, we can estimate

$$\left| \sum_{n=1}^{\infty} (a_{n+k} - b_{n+k})\, r^{-n} \right| \le \sum_{n=1}^{\infty} |a_{n+k} - b_{n+k}|\, r^{-n} \le \sum_{n=1}^{\infty} 2\rho r^{-n} = \frac{2\rho}{r-1}$$

Generally, we have $1 < 2\rho/(r-1) \le 2$, but the pathological equality $b_k = a_k \pm 2$ may only hold true in the case where $\rho = r-1$ and $x = (a_0, a_1, \dots, a_k, \pm\rho, \pm\rho, \pm\rho \dots) = (b_0, b_1, \dots, b_k, \mp\rho, \mp\rho, \mp\rho, \dots)$, which we have excluded from consideration. Hence, we deduce that

$$b_k = a_k \pm 1$$

Now we set

$$i_1 = \begin{cases} k, & \text{if} \quad b_k = a_k + 1 \\ -k, & \text{if} \quad b_k = a_k - 1 \end{cases}$$

and if $(a'_n)_{n\in\mathbb{N}_0} = f_{i_1}\left((a_n)_{n\in\mathbb{N}_0}\right)$, then $b_n = a'_n$, $n \in \{1, \dots, k\}$.

Once $i_1, \dots, i_{n-1}$ have been chosen, let $i_n = |i_n| \cdot \operatorname{sgn}(i_n)$, where

$$\begin{aligned} |i_n| &= \inf\left\{ j \in \mathbb{N}_0 : b_j \ne \left(f_{i_1 \dots i_{n-1}}\left((a_n)_{n\in\mathbb{N}_0}\right)\right)_j \right\} \\ \operatorname{sgn}(i_n) &= \begin{cases} 1, & \text{if} \quad b_{|i_n|} = \left(f_{i_1 \dots i_{n-1}}\left((a_n)_{n\in\mathbb{N}_0}\right)\right)_{|i_n|} + 1 \\ -1, & \text{if} \quad b_{|i_n|} = \left(f_{i_1 \dots i_{n-1}}\left((a_n)_{n\in\mathbb{N}_0}\right)\right)_{|i_n|} - 1 \end{cases} \end{aligned}$$

It may happen that $i_n = 0$ for all $n > p$, $p \in \mathbb{N}$. In this case, we shall consider the resulting sequence $(i_1, \dots, i_p)$ to be finite. This concludes the proof. □

3 The representation

We aim to represent real numbers by sequences from the representation set S, as defined in Section 2. For example, one might define a computable exact real number x as a triple (r, E, M), where $E \in \mathbb{Z}$ is an exponent, M is a mantissa which is a sequence of numbers $(a_n)_{n\in\mathbb{N}_0} \in S$, and the value of x is computed as

$$x = r^E \cdot \sum_{n=0}^{\infty} a_n r^{-n}. \tag{4}$$

Such a representation, however, would be too loose a concept to be useful by itself. We must also provide some constructive condition in order to guarantee convergence of the series in (4) and be able to make useful inferences about a number from a finite amount of information about its representation.

In this light, we define a *representation* of an exact real number x to be a quadruple (r, ρ, E, M), where $r \in \mathbb{N}$, $r > 1$ is the *radix* value, the *range parameter* ρ is an integer with $r/2 \leq \rho \leq r-1$, $E \in \mathbb{Z}$ is a signed *exponent*, M is a *mantissa*, which is an *effectively given*[2] sequence of integers $(a_n)_{n\in\mathbb{N}_0}$ such that

$$|a_n| \leq Cn, \; n \in \mathbb{N}, \tag{5}$$

where $C > 0$ is a constant, common to all real numbers in a given system — we therefore do not include it in the representation[3]. The representation $(r, \rho, E, (a_n)_{n\in\mathbb{N}_0})$ is said to be *canonical* or *normalized*, if

$$|a_n| \leq \rho, \; n \in \mathbb{N}_0.$$

The value of $x = (r, \rho, E, (a_n)_{n\in\mathbb{N}_0})$ is taken as in (4). Later on we will centre on the factors that affect the choice of appropriate values for the parameters r and ρ.

For brevity and ease of reading, we shall not always distinguish between a number x and its representation (r, ρ, E, M), and refer to a number as normalized if its representation is normalized, and vice versa. We shall also assume that r and ρ are fixed and sometimes use the notation (E, M) instead of (r, ρ, E, M).

Observe that we can view a *finite* number as being infinite, by attaching an infinite sequence of zeros at the end of its mantissa:

$$r^E \cdot \sum_{i=0}^{N} a_i r^{-i} = r^E \cdot \sum_{i=0}^{\infty} a_i r^{-i},$$

where we have set $a_i = 0$ for $i > N$. We can therefore assume, without any restriction of generality, that the mantissas of all operands are infinite, unless otherwise specified.

4 Normalization

Most algorithms presented in this and subsequent chapters assume that all operands are normalized, and also require normalization of the results, so

[2]The sequence $(a_n)_{n\in\mathbb{N}_0}$ could in principle be given by an oracle — it does not necessarily have to be *computable* in the sense of being the sequence of values $g(0), g(1), g(2), \ldots$ of a general recursive function $g(x)$.

[3]The convergence criterion (5) is somewhat arbitrary and only required to ensure effective convergence of the sequence. If a sequence were found to violate (5), an error message would be produced at run-time. The value of $C = 2\,(r-1)^2$ could be given as a rough estimate that satisfies the algorithmic requirements.

we shall now discuss the algorithms for normalizing real numbers. We recall that normalization refers to the process of restoring the individual digits of a real number's mantissa $(a_i)_{i\in\mathbb{N}_0}$ to the canonical range $[-\rho,\rho]$.

Let $(a_i)_{i\in\mathbb{N}_0}$ be an unnormalized mantissa of a real number $a = r^E \cdot \sum_{i=0}^{\infty} a_i r^{-i}$. We shall first confine our attention to the case where $|a_i| \le r+\rho-1,\ i\in\mathbb{N}_0$, and show how to obtain a new exponent E' and mantissa $(a_i')_{i\in\mathbb{N}_0}$ such that

$$\begin{aligned} &1)\quad a = r^E \cdot \sum_{i=0}^{\infty} a_i r^{-i} = r^{E'} \cdot \sum_{i=0}^{\infty} a_i' r^{-i} \\ &2)\quad |a_i'| \le \rho,\ i\in\mathbb{N}_0 \end{aligned} \tag{6}$$

To this end, we first consider $\sum_{i=0}^{\infty} a_i r^{-i}$ and repeatedly divide a_i by r for all $i\in\mathbb{N}_0$:

$$a_i = d_i r + m_i, \quad |m_i| < r, \quad \operatorname{sgn}(m_i) = \operatorname{sgn}(d_i) \tag{7}$$

We have:

$$\begin{aligned} \sum_{i=0}^{\infty} a_i r^{-i} &= \sum_{i=0}^{\infty} (d_i r + m_i)\, r^{-i} = \sum_{i=0}^{\infty} d_i r^{-i+1} + \sum_{i=0}^{\infty} m_i r^{-i} \\ &= (d_0 r + m_0 + d_1) + \sum_{i=1}^{\infty} (m_i + d_{i+1})\, r^{-i} \end{aligned} \tag{8}$$

Now $|a_i| \le r+\rho-1$ implies $|d_i| \le 1$, $|m_i| \le r-1$ $(i\in\mathbb{N}_0)$ and, thus, $|m_i + d_{i+1}| \le r$. We, however, aim to obtain a value less or equal to ρ (instead of r). Let us introduce the following notation:

$$\begin{aligned} d_i' &= \begin{cases} d_i, & \text{if } |m_i| < \rho \\ d_i + \operatorname{sgn}(m_i), & \text{if } |m_i| \ge \rho \end{cases} \\ m_i' &= \begin{cases} m_i, & \text{if } |m_i| < \rho \\ m_i - \operatorname{sgn}(m_i)\cdot r, & \text{if } |m_i| \ge \rho \end{cases} \\ a_i' &= m_i' + d_{i+1}' \ (i\in\mathbb{N}), \quad a_0' = a_0 + d_1' \end{aligned} \tag{9}$$

From (9) it can be seen that $a_i = d_i' r + m_i'$. and, similarly to (8), we arrive at

$$\sum_{i=0}^{\infty} a_i r^{-i} = (a_0 + d_1') + \sum_{i=1}^{\infty} \left(m_i' + d_{i+1}'\right) r^{-i} = \sum_{i=0}^{\infty} a_i' r^{-i}$$

Let us now verify that $|a_i'| \le \rho$ for all $i \in \mathbb{N}$. For this purpose we note that $|d_i'| \le 1$ for all $i \in \mathbb{N}_0$, so if $|m_i| < \rho$, then $m_i' = m_i$, $|m_i'| < \rho$, and $|a_i'| = \left|m_i' + d_{i+1}'\right| \le \rho$. If $|m_i| \ge \rho$, then $m_i' = m_i - \operatorname{sgn}(m_i) \cdot r$, $1 \le |m_i'| \le r - \rho$, and $|a_i'| = \left|m_i' + d_{i+1}'\right| \le r - \rho + 1$. Now we require $r - \rho + 1 \le \rho$ or, equivalently,

$$\rho \ge \frac{r+1}{2} \tag{10}$$

which in its turn implies that $(r+1)/2 \le r - 1$ or $r \ge 3$. Using (10), we finally obtain $|a_i'| \le \rho$ for all $i \in \mathbb{N}$. For $i = 0$, however, the inequality does not necessarily hold true. On the other hand, from the definition of a_0' we can conclude that $a_0 - 1 \le a_0' \le a_0 + 1$.

In this manner, we have constructed a function $f : \mathbb{N}^2 \times \mathbb{Z}^{\mathbb{N}_0} \to \mathbb{Z}^{\mathbb{N}_0}$ (it will be referred to as `reduce`) which assigns to any triple $\left(r, \rho, (a_i)_{i\in\mathbb{N}_0}\right)$ the sequence $(a_i')_{i\in\mathbb{N}_0}$, calculated according to formulae (7) and (9). Evaluation of this function can be performed totally in parallel (Fig. 1).

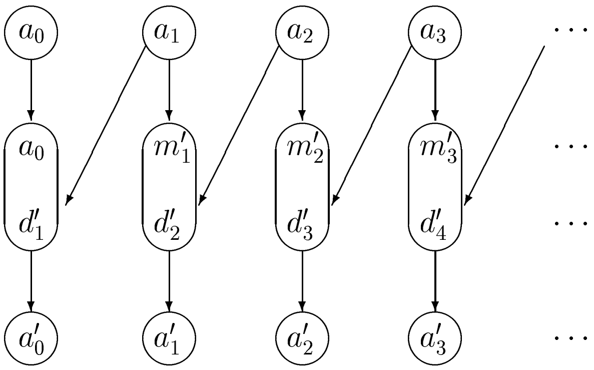


Figure 1: Totally parallel normalization

Returning to formula (6), we now construct the promised number as follows:

$$E' = \begin{cases} E, & \text{if } |a_0| \le \rho - 1 \\ E+1, & \text{if } |a_0| \ge \rho \end{cases}$$

$$(a_i')_{i\in\mathbb{N}_0} = \begin{cases} f\left(r, \rho, (a_i)_{i\in\mathbb{N}_0}\right), & \text{if } |a_0| \le \rho - 1 \\ f\left(r, \rho, (0, a_0, a_1, \dots, a_n, \dots)\right), & \text{if } |a_0| \ge \rho \end{cases}$$

Now let us consider a more general case where $|a_i| \le M$, $i \in \mathbb{N}_0$, where $M > 0$ is an arbitrary positive integer. We can now easily show that it is possible to normalize mantissa $(a_i)_{i\in\mathbb{N}_0}$ in a finite number of steps. Indeed, applying `reduce`, we shall obtain a sequence $(a_i')_{i\in\mathbb{N}_0}$, satisfying the following condition:

$$|a_i'| = \left|m_i' + d_{i+1}'\right| \le |m_i'| + \left|d_{i+1}'\right| \le \rho - 1 + \left\lfloor \frac{M}{r} \right\rfloor + 1,$$

or

$$|a_i'| \le M_1 \stackrel{def}{=} \left\lfloor \frac{M}{r} \right\rfloor + \rho.$$

Applying `reduce` again, we get another sequence $(a_i'')_{i\in\mathbb{N}_0}$, satisfying

$$|a_i''| \le M_2 \stackrel{def}{=} \left\lfloor \frac{M_1}{r} \right\rfloor + \rho,$$

etc. The sequence $M, M_1, M_2, \ldots$ is a sequence of decreasing natural numbers, and if $M = m_n r^n + \ldots + m_1 r + m_0$, the algorithm will terminate in at most $n+1$ steps.

More specifically, we can prove the following result.

Theorem 2. *Let $(a_i)_{i\in\mathbb{N}_0}$ be a sequence with $|a_i| \le M$, $i \in \mathbb{N}$, where M is an arbitrary positive integral number. In order that the sequence $(a_i)_{i\in\mathbb{N}_0}$ be normalized to an equivalent sequence $(b_i)_{i\in\mathbb{N}_0}$ with $|b_i| \le N$, $i \in \mathbb{N}$ on applying* **`reduce`** *at most n times, it is sufficient that $M \le g^{(n)}(N)$, where $g^{(n)}(x) = r^n x + C_n$, $C_n = (r^n - 1)(1-\rho)$.*

Proof. To prove the sufficiency of the condition imposed upon M, we need but note that the functions $g^{(n)}(x)$ satisfy the following recurrence formulae

$$g^{(n)}(x) = r \cdot g^{(n-1)}(x) + C_1,\ n \in \mathbb{N},$$

where $g^{(0)}(x) \equiv x$. Equivalently,

$$g^{(n)}(x) = \underbrace{g \circ g \circ \cdots \circ g}_{n \text{ times}}(x),$$

where

$$g(x) = g^{(1)}(x) = rx + C_1.$$

Thus, it suffices to show that any sequence $(a_i)_{i\in\mathbb{N}_0}$ with $|a_i| \le g(x)$, $i \in \mathbb{N}$ can be reduced, in a single step, to a sequence $(a_i')_{i\in\mathbb{N}_0}$ with $|a_i'| \le x$, $i \in \mathbb{N}$.

Let $(a_i)_{i\in\mathbb{N}_0}$ be any such sequence, i.e. $|a_i| \le rx + C_1$, $i \in \mathbb{N}$. As indicated above, $|a_i| \le M$ implies $|a_i'| \le M_1 = \left\lfloor \frac{M}{r} \right\rfloor + \rho$, and thus picking $M = rx + C_1$ yields $M_1 = \left\lfloor x + (1-\rho)\frac{r-1}{r} \right\rfloor + \rho < x + (1-\rho) + \rho = x + 1$, i.e.

$$|a_i'| \le x,$$

which is what had to be proved. $\square$

Corollary 1. *If $(a_i)_{i\in\mathbb{N}_0}$ is a sequence satisfying $|a_i| \le g^{(n)}(\rho)$ for some $n \in \mathbb{N}$ and all $i \in \mathbb{N}$, it can be fully normalized by* `reduce` *in at most n steps.*

This follows immediately from the theorem: n normalizations give us a sequence $(a'_i)_{i\in\mathbb{N}_0}$ with $|a'_i| \le \rho$.

The converse statement is not necessarily true: even if $|a_k| > g^{(n)}(N)$ for some $k \in \mathbb{N}$, after n normalizations we may still get a $(b_i)_{i\in\mathbb{N}_0}$ with $|b_i| \le N$ for all $i \in \mathbb{N}$. Suppose, for instance, that $a_k = g(N) + 1 = r(N - \rho + 1) + \rho$ for some $k \in \mathbb{N}$ and $|a_i| \le g(N)$ for $i \ne k$. This implies that $d_k = N - \rho + 1$, $m_k = \rho$ and, therefore, $d'_k = N - \rho + 2$, $m'_k = \rho - r$. Recalling that

$$a'_{k-1} = m'_{k-1} + d'_k, \ k \in \mathbb{N},$$

one can see that the larger-than-usual value of d'_k can only affect the $(k-1)$-st element of the resulting sequence, and, further still, only if $m'_{k-1} = \rho - 1$, in which case $a'_{k-1} = \rho - 1 + N - \rho + 2 = N + 1$. However, the value of m'_{k-1} depends solely on a_{k-1}, and can be anywhere in the range from $-\rho + 1$ to $\rho - 1$, irrespective of the value of the next element, a_k. If it so happens that $m'_{k-1} \ne \rho - 1$, we will have $\left|a'_{k-1}\right| \le N$, and consequently — since the a'_i for $i \ne k - 1$ have remained intact — $|a'_i| \le N$ for all $i \in \mathbb{N}$.

This example shows that the functions $g^{(n)}(x)$ give us, in fact, the best upper bound one could possibly have in order that *any* sequence bounded by it be safely normalized. More precisely, for any integer function $f^{(n)}(x) > g^{(n)}(x)$ there is a sequence $(a_i)_{i\in\mathbb{N}_0}$ with $|a_i| \le f^{(n)}(\rho)$ that cannot be fully normalized in n applications of `reduce`.

By way of illustration, let us give a few examples.

Example 1. *Let $r = 6$, $\rho = 4$, $(a_i)_{i\in\mathbb{N}_0}$ be a sequence with $|a_i| \le 3500$ for all $i \in \mathbb{N}$. How many times does one have to apply* `reduce` *to obtain an equivalent sequence $(a'_i)_{i\in\mathbb{N}_0}$ with $|a'_i| \le 100$, $i \in \mathbb{N}$?*

We have

$$\begin{aligned} g(100) &= 6 \cdot 100 - 15 = 585 \\ g^{(2)}(100) &= g(585) = 6 \cdot 585 - 15 = 3495 \\ g^{(3)}(100) &= g(3495) = 6 \cdot 3495 - 15 = 20955 \end{aligned}$$

Since $g(100) < g^{(2)}(100) < 3500 < g^{(3)}(100)$, 3 normalizations will be sufficient by theorem 2.

Example 2. *Let $r = 10$, $\rho = 6$. Find the bound for the elements of a sequence that can be fully normalized in 3 applications of* `reduce`*.*

According to Corollary 1, we need but calculate

$$g^{(3)}(6) = 1000 \cdot 6 + 999 \cdot (-5) = 1005.$$

Thus, if $|a_i| \le 1005$, $i \in \mathbb{N}$, $(a_i)_{i\in\mathbb{N}_0}$ *can be fully* **`reduced`** *in three passes.*

The functions $g^{(n)}(x)$ have a much simpler form when $x = \rho$: indeed, it is easy to see that

$$g^n(\rho) = r^n + \rho - 1,\ n \in \mathbb{N}. \tag{11}$$

The right-hand side of equality (11) is solvable for n, which enables us to determine the number of times one has to apply `reduce` in order to *fully* normalize a given sequence $(a_i)_{i\in\mathbb{N}_0}$. In more exact terms, let $(a_i)_{i\in\mathbb{N}_0}$ be a sequence with $|a_i| \le M$, $i \in \mathbb{N}$. By theorem 2,

$$n = \min\left\{k \in \mathbb{N} \;\middle|\; M \le g^k(\rho)\right\}.$$

Solving the inequality $M \le g^k(\rho)$ for $k \in \mathbb{N}$, we find that

$$k \ge \log_r\left(M - \rho + 1\right),$$

or,

$$n = \lceil \log_r\left(M - \rho + 1\right) \rceil \tag{12}$$

As a conclusion, let us take note of the fact that, as it follows from (10), in order for our system to allow totally parallel normalization, i.e. absence of carry propagation chains, it must *not* be minimally redundant. For $r = 2$, for instance, there is only one possible digit set, $\left\{\overline{1}, 0, 1\right\}$; thus, in the binary case the condition $\rho \ge (r+1)/2 = 3/2$ cannot be satisfied. Henceforth, only non-minimally redundant systems will be considered.

5 Basic arithmetic operations

5.1 Addition and subtraction

In this section, we shall discuss algorithms for the operations of exact real addition and subtraction. The emphasis will mainly be on the former, since subtraction is usually carried out as the addition of a negated number. We shall first discuss addition of two numbers and then look at multiple number addition.

5.1.1 Addition of two numbers

Let $a = r^{E_a} \cdot \sum\limits_{i=0}^{\infty} a_i r^{-i}$ and $b = r^{E_b} \cdot \sum\limits_{j=0}^{\infty} b_j r^{-j}$ be the two normalized radix-r numbers to be added. Since the addition operation is commutative, we can assume $e = E_a - E_b \geq 0$ without loss of generality. The procedure for addition or subtraction is as follows:

$$
\begin{aligned}
a + b &= r^{E_a} \cdot \sum_{i=0}^{\infty} a_i r^{-i} + r^{E_b} \cdot \sum_{i=0}^{\infty} b_i r^{-i} = r^{E_a} \left(\sum_{i=0}^{\infty} a_i r^{-i} + r^{-e} \cdot \sum_{i=0}^{\infty} b_i r^{-i} \right) \\
&= r^{E_a} \cdot \sum_{i=0}^{\infty} \left(a_i + b_i'\right) r^{-i},
\end{aligned}
$$

where

$$
(b_0', b_1', b_2', \cdots, b_n', \cdots)_r = \left(\underbrace{0, 0, \cdots, 0}_{e \text{ zeros}}, b_0, b_1, b_2, \cdots \right)_r .
$$

Thus, in order to perform addition, we must first adjust the mantissa of one of the operands to make the two exponents equal (align the radix points), and then add the two sequences digit by digit. The resulting sequence

$$
(a_0 + b_0',\ a_1 + b_1', \cdots,\ a_n + b_n',\ \cdots)_r
$$

can then be normalized in a single pass, since

$$
|a_n + b_n'| \leq |a_n| + |b_n'| \leq 2\rho \leq r + \rho - 1.
$$

5.1.2 Subtraction

Subtraction is carried out in the usual way by negating the minuend and adding the result to the subtrahend. Negation is performed as follows:

$$
-\sum_{i=0}^{\infty} a_i r^{-i} = \sum_{i=0}^{\infty} (-a_i) \cdot r^{-i}
$$

5.1.3 Addition of several numbers

The above addition algorithm can be readily modified to operate with n numbers, where $n > 2$. The procedure is essentially the same — the mantissas of all n numbers are first aligned to match the one with the largest exponent, and then added digit-by-digit. As it follows from (12), the resulting sequence can be normalized by applying `reduce` $\lceil \log_r (n\rho - \rho + 1) \rceil$ times.

Note that this is considerably more efficient than adding the n numbers pairwise using $(n-1)$ nested additions, as we discuss later (Section 6.1).

5.2 Multiplication

Let the multiplier and multiplicand be denoted by $a, b \in \mathbb{R}$ respectively, with the following normalized sequences of signed digits:

$$(a_0, a_1, a_2, \cdots, a_n, \cdots), \quad (b_0, b_1, b_2, \cdots, b_n, \cdots),$$

i.e.

$$a = r^{E_a} \cdot \sum_{i=0}^{\infty} a_i r^{-i}, \quad b = r^{E_b} \cdot \sum_{j=0}^{\infty} b_j r^{-j}$$

Then

$$ab = r^{E_a + E_b} \cdot \left(\sum_{i=0}^{\infty} a_i r^{-i} \right) \cdot \left(\sum_{j=0}^{\infty} b_j r^{-j} \right)$$

The Cauchy product of the two series $\sum\limits_{i=0}^{\infty} a_i r^{-i}$ and $\sum\limits_{j=0}^{\infty} b_j r^{-j}$ is the series

$$\sum_{m=0}^{\infty} c_m r^{-m} = \sum_{m=0}^{\infty} \left(\sum_{i=0}^{m} a_i b_{m-i} \right) r^{-m},$$

where $c_m = \left(\sum\limits_{i=0}^{m} a_i b_{m-i} \right)$. Since both series $a = \sum\limits_{n=0}^{\infty} a_n r^{-n}$ and $b = \sum\limits_{n=0}^{\infty} b_n r^{-n}$ are absolutely convergent, by Mertens' theorem (see e.g. [15]) their Cauchy product $\sum\limits_{n=0}^{\infty} c_n r^{-n}$ converges to ab.

Since $(a_i)_{i \in \mathbb{N}_0}$ and $(b_i)_{i \in \mathbb{N}_0}$ are canonical representations of a and b, we have

$$|c_m| \leq \rho^2 \cdot (m+1), \; m \in \mathbb{N}_0.$$

Now we want to find the result in the form

$$ab = c = r^{E_c} \cdot \sum_{m=0}^{\infty} c'_m r^{-m},$$

where $-\rho \leq c'_m \leq \rho$ for all $m \in \mathbb{N}_0$. However, the sequence $(c_m)_{m \in \mathbb{N}_0}$ cannot be normalized directly, because generally it is not bounded by any positive integer. Instead, we can recursively apply `reduce` to small bounded portions of $(c_m)_{m \in \mathbb{N}_0}$, as shown in Figures 2 and 3.

$$
\begin{array}{cccc|ccccc|c}
a_0b_0 & a_0b_1 & \cdots & a_0b_{n-1} & a_0b_n & a_0b_{n+1} & a_0b_{n+2} & \cdots & a_0b_{2n-1} & a_0b_{2n} \\
 & a_1b_0 & \cdots & a_1b_{n-2} & a_1b_{n-1} & a_1b_n & a_1b_{n+1} & \cdots & a_1b_{2n-2} & a_1b_{2n-1} \\
 & & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & & & a_{n-1}b_0 & a_{n-1}b_1 & a_{n-1}b_2 & a_{n-1}b_3 & \cdots & a_{n-1}b_n & a_{n-1}b_{n+1} \\
 & & & & a_nb_0 & a_nb_1 & a_nb_2 & \cdots & a_nb_{n-1} & a_nb_n \\
\hline
 & & & & & a_{n+1}b_0 & a_{n+1}b_1 & \cdots & a_{n+1}b_{n-2} & a_{n+1}b_{n-1} \\
 & & & & & & a_{n+2}b_0 & \cdots & a_{n+2}b_{n-3} & a_{n+2}b_{n-2} \\
 & & & & & & & \ddots & \vdots & \vdots \\
 & & & & & & & & a_{2n-1}b_0 & a_{2n-1}b_1 \\
 & & & & & & & & & a_{2n}b_0
\end{array}
$$

Figure 2: Multiplication — before normalizing

$$
\begin{array}{cccc|ccccc|c}
c_{00} & c_{01} & \cdots & c_{0,n-1} & c_{0n} & c_{0,n+1} & c_{0,n+2} & \cdots & c_{0,2n-1} & c_{0,2n} \\
\hline
 & & & & & a_{n+1}b_0 & a_{n+1}b_1 & \cdots & a_{n+1}b_{n-2} & a_{n+1}b_{n-1} \\
 & & & & & & a_{n+2}b_0 & \cdots & a_{n+2}b_{n-3} & a_{n+2}b_{n-2} \\
 & & & & & & & \ddots & \vdots & \vdots \\
 & & & & & & & & a_{2n-1}b_0 & a_{2n-1}b_1 \\
 & & & & & & & & & a_{2n}b_0
\end{array}
$$

Figure 3: Multiplication — after normalizing first $(n+1)$ lines

Namely, let us choose some $n \in \mathbb{N}$, then for all $m > n$ we write:

$$\sum_{i=0}^{m} a_i b_{m-i} = \sum_{i=0}^{n} a_i b_{m-i} + \sum_{i=n+1}^{m} a_i b_{m-i}$$

We have:

$$
\begin{aligned}
\sum_{m=0}^{\infty} c_m r^{-m} &= \sum_{m=0}^{\infty} \left(\sum_{i=0}^{m} a_i b_{m-i} \right) r^{-m} \\
&= \sum_{m=0}^{n} \left(\sum_{i=0}^{m} a_i b_{m-i} \right) r^{-m} + \sum_{m=n+1}^{\infty} \left(\sum_{i=0}^{n} a_i b_{m-i} + \sum_{i=n+1}^{m} a_i b_{m-i} \right) r^{-m} \\
&= \sum_{m=0}^{\infty} \left(\sum_{i=0}^{\min(m,n)} a_i b_{m-i} \right) r^{-m} + \sum_{m=n+1}^{\infty} \left(\sum_{i=n+1}^{m} a_i b_{m-i} \right) r^{-m} \qquad (13)
\end{aligned}
$$

Now the sums $\sum\limits_{i=0}^{\min(m,n)} a_i b_{m-i}$ are bounded for all $m \in \mathbb{N}_0$

$$\left| \sum_{i=0}^{\min(m,n)} a_i b_{m-i} \right| \leq (n+1)\, \rho^2, \tag{14}$$

so we can apply `reduce` to the sequence $\left(\sum\limits_{i=0}^{\min(m,n)} a_i b_{m-i} \right)_{m \in \mathbb{N}_0}$. Having done so $m(n)$ times, where

$$m\,(n) = \left\lceil \log_r \left((n+1)\, \rho^2 - \rho + 1 \right) \right\rceil ,$$

we shall obtain an equivalent sequence $(c_{0m})_{m \in \mathbb{N}_0}$ satisfying $|c_{0m}| \leq \rho$ for all $m \in \mathbb{N}_0$, i.e.

$$\sum_{m=0}^{\infty} \left(\sum_{i=0}^{\min(m,n)} a_i b_{m-i} \right) r^{-m} = \sum_{m=0}^{\infty} c_{0m} r^{-m}, \text{ where } |c_{0m}| \leq \rho. \tag{15}$$

Returning to (13), we rewrite it in the form

$$\sum_{m=0}^{\infty} c_m r^{-m} = \sum_{m=0}^{n-1} c_{0m} r^{-m} + r^{-n} \sum_{m=0}^{\infty} c_m^{(1)} r^{-m},$$

where

$$\begin{aligned} c_0^{(1)} &= c_{0n} \\ c_m^{(1)} &= c_{0,n+m} + \sum_{i=n+1}^{n+m} a_i b_{n+m-i},\ m \in \mathbb{N} \end{aligned} \tag{16}$$

Proceeding recursively with the series

$$\sum_{m=0}^{\infty} c_m^{(j)} r^{-m} = \sum_{m=0}^{n-1} c_{jm} r^{-m} + r^{-n} \sum_{m=0}^{\infty} c_m^{(j+1)} r^{-m},\ j \in \mathbb{N},$$

we obtain an equivalent sequence $(c'_m)_{m \in \mathbb{N}_0}$:

$$\begin{aligned} \sum_{m=0}^{\infty} c_m r^{-m} &= \sum_{m=0}^{n-1} c_{0m} r^{-m} + r^{-n} \left(\sum_{m=0}^{n-1} c_{1m} r^{-m} + r^{-n} \left(\sum_{m=0}^{n-1} c_{2m} r^{-m} + \ldots \right.\right. \\ &= \sum_{m=0}^{n-1} c_{0m} r^{-m} + \sum_{m=n}^{2n-1} c_{1,m-n} r^{-m} + \sum_{m=2n}^{3n-1} c_{2,m-2n} r^{-m} + \ldots \\ &= \sum_{k=0}^{\infty} \left(\sum_{m=kn}^{(k+1)n-1} c_{k,m-kn} r^{-m} \right) = \sum_{m=0}^{\infty} c'_m r^{-m}, \end{aligned}$$

where

$$\begin{aligned}
c'_m &= c_{m \operatorname{div} n,\, m \bmod n}, \ |c'_m| \le \rho, \ m \in \mathbb{N}_0 \\
(c_{km})_{m\in\mathbb{N}_0} &\sim \left(\left(\sum_{i=0}^{\min(m,n)} d^{(k)}_{i,m-i} \right)_{m\in\mathbb{N}_0} \right), \ |c_{km}| \le \rho \\
d^{(k+1)}_{ij} &= \begin{cases} c_{k,j+n}, & i = 0 \\ d^{(k)}_{i+n,j}, & i \in \mathbb{N} \end{cases} \\
d^{(0)}_{ij} &= a_i b_j \\
c^{(k+1)}_m &= \sum_{i=0}^{m} d^{(k+1)}_{i,m-i} \\
c^{(0)}_m &= c_m
\end{aligned}$$

Thus, $(c'_m)_{m\in\mathbb{N}_0}$ is the required result of multiplication.

5.3 Division

The intention here is to develop algorithms for division of exact real numbers. Let $N \in \mathbb{R}$ be the dividend, $D \in \mathbb{R}$, $D \ne 0$ — the divisor, their redundant signed-digit radix-r representations given by

$$N = r^{E_N} \cdot \sum_{i=0}^{\infty} n_i r^{-i}, \quad D = r^{E_D} \cdot \sum_{i=0}^{\infty} d_i r^{-i},$$

where $|n_i| \le \rho$, $|d_i| \le \rho$ for $i \in \mathbb{N}$. The task is to compute a real quotient

$$Q = r^{E_Q} \cdot \sum_{i=0}^{\infty} q_i r^{-i}$$

such that $N = Q \cdot D$ and $|q_i| \le \rho$, $i \in \mathbb{N}$.

A considerable body of work exists in the literature on the methods of signed-digit division, most of which in one way or another owe their origin to an algorithm due to Robertson [12]. The substance of the algorithm lies with an iterative process that produces one digit of the quotient per cycle according to the following recurrence equation

$$P_{n+1} = r\,(P_n - q_n D)\,,\ n \in \mathbb{N}_0, \tag{17}$$

where $P_0 = N$, P_n is the current partial remainder, P_{n+1} is the next partial remainder, and q_n is the quotient digit inferred from P_n and D. It is easy to see that

$$P_n = r^n \left[N - \left(q_0 + q_1 r^{-1} + \ldots + q_{n-1} r^{-n+1}\right) D\right],\ n \in \mathbb{N},$$

and so imposing an upper bound on the value of $|P_n|$ will ensure convergence of the algorithm, provided that selection of the quotient digit q_n results in the next partial remainder P_{n+1} adhering to the same allowed range as P_n.

The existing signed-digit division algorithms primarily differ in their selection of quotient digits, restriction of the range of the possible values of the divisor, dividend and partial remainders and, finally, normalization techniques.

The conventional, non-redundant algorithms also use relation (17) but always produce *correct* quotient digits — the multiplications of the divisor by the digits of the quotient are done by repeated subtraction, and a guessed digit is known to be incorrect if it is either too large and the subsequent subtraction leaves a negative result, or it is too small and the subtraction leaves a result that exceeds a multiple of the divisor in that digit position.

In redundant signed-digit representations, however, the sign of an intermediate result may not be readily available for inspection, because a number of its most significant digits, generally unknown in advance, may happen to be all zero. The usual way to get around this problem is to make a *guess* about q_n based on the inspection of several most significant digits of P_n and D. Even though this could result in some quotient digits q_n selected in this way being incorrect, the redundancy allows recovery from wrong guesses by taking an appropriate correction step in the next quotient digit. As long as the next q can correct an error in the previous step, convergence of the algorithm is guaranteed.

The method of division put forward here is a modification of the original Robertson's signed-digit division algorithm and is similar to that recently reported by David Smith [14].

The algorithm uses the above recurrence relation (17) and the following digit selection function:

$$q_n = \left\lfloor \left| \frac{p_{n0}}{d_0} \right| \right\rfloor \cdot \operatorname{sgn}\left(\frac{p_{n0}}{d_0}\right), \tag{18}$$

where p_{n0} is the first digit of the n-th partial remainder P_n, and d_0 is the first digit of the divisor D which, being non-zero[4], is so scaled that $|d_0| \geq r^2$. Beginning with $P_0 = N$, we have the following sequences of digits representing P_{n+1}, $n \in \mathbb{N}_0$:

$$\begin{aligned} P_{n+1} &= r \cdot (p_{n0} - q_n d_0,\, p_{n1} - q_n d_1,\, \cdots,\, p_{nk} - q_n d_k,\, \cdots)_r \\ &= (r(p_{n0} - q_n d_0) + (p_{n1} - q_n d_1),\, p_{n2} - q_n d_2,\, \cdots,\, p_{nk} - q_n d_k,\, \cdots)_r \end{aligned}$$

[4]Note that since D is represented by an infinite sequence of digits, one cannot effectively check whether or not it is non-zero.

The early algorithms fully normalized P_n, $n \in \mathbb{N}$ at each step in order to keep the entries of the sequence bounded. However, as recently shown in [14] (and also suggested by Carl Pixley in the early 1980's), it is possible to skip the full normalization of the partial remainders, and instead normalize only a few leading digits. The details of the algorithm analysis are given in [14], and although considering the operands to be finite and given in non-redundant form, readily lend themselves to the elaboration necessary to extend the method to operate with infinite sequences of signed digits.

The elimination of most intermediate digit normalizations makes the division algorithm run in double-quick time, and at high precision nearly as fast as multiplication.

6 Complexity analysis

The chief and computationally most significant part of the algorithms presented in this paper is the normalization function, which is for the greater part responsible for the complexity of the four arithmetic operations.

The normalization procedure relies upon unbounded integer arithmetic for its operation, and hence the speed of normalization is crucially dependent on the speed of same. As seen in Fig. 1, normalization always requires one-digit carry-look-ahead — to produce N radix-r digits of a normalized result, it is necessary to compute $N+1$ digits of the number being normalized, whereafter N out of the $N+1$ digits (excluding the first one) are divided by r, and the results of the divisions — added, possibly in parallel, resulting in a total of N integer divisions by r, and N integer additions. If r is a power of 2, the divisions by r can be done by simple shifts.

Similarly, if normalization is to be performed m times, in order to obtain N digits of the result, we need $N+m$ digits of the original number, as well as $N+(N+1)+\ldots+(N+m-1)=m(N+(m-1)/2)$ divisions by r and additions.

6.1 Addition and subtraction

The computation of N digits of the sum of two numbers requires $N+1$ digits of the operands, N integer divisions by r and $2N+1$ integer additions. Addition of n numbers, where $n>2$, requires $N+m$ digits of the operands, $m(N+(m-1)/2)$ divisions by r and $m(N+(m-1)/2)+(n-1)(N+m)$ additions, where $m=\lceil\log_r(n\rho-\rho+1)\rceil$. This is, of course, much better than the repeated binary addition $x_1+(x_2+(x_3+(\cdots+x_n)))$, which results in the evaluation of $N+n$ digits of the operands, $(n-1)(N+(n-2)/2)$ divisions by r and $2(n-1)N+(n-1)^2$ additions.

Subtraction is only different from addition in that negation is performed beforehand. Negation, of course, does not require any look-ahead, and its complexity is simply that of changing the sign of a number's digits.

6.2 Multiplication

The complexity of the multiplication algorithm depends on the value of the parameter $n \in \mathbb{N}$ that appears in (14). Let us address ourselves to the question of choosing an appropriate value for n. In principle, the algorithm will work correctly with any $n \in \mathbb{N}$, so our main concern here is to minimize the number of operations needed to compute N digits of the result. Note that when $n = 1$, the algorithm is the same as that adopted by Avizienis [2].

Now let

$$N = pn + q,\ 0 \leq q < n.$$

We have $p+1$ partial normalization groups (the first two groups are shown in Fig. 2), each of which requires at most $m(n) = \lceil \log_r \left((n+1)\,\rho^2 - \rho + 1 \right) \rceil$ applications of `reduce` (see (14)). On account of the granularity effect, to compute N digits of the product, the normalization procedure requires m extra digits from the last partial normalization group, $2m$ extra digits from the second-last one, and so on; the first group requiring as many as pm extra digits, thus making the total number of integer divisions and additions

$$\begin{aligned}
N_{div} = \quad & m(N + pm + (m-1)/2) + \\
& m((N-n) + (p-1)m + (m-1)/2) + \ldots + \\
& m((N-pn) + (m-1)/2) \\
= \quad & m\left(N + (N-n) + (N-2n) + \ldots + (N-pn)\right) + \\
& m^2(p + (p-1) + \ldots + 1) + m(m-1)(p+1)/2 \\
= \quad & \frac{1}{2} m\,(p+1)\,(N + q + mp + m - 1)\,.
\end{aligned}$$

The corresponding formula for the number of integer multiplications of the operands' digits is

$$\begin{aligned}
N_{mult} &= \frac{N(N+1)}{2} + (q-1)m + n \cdot (2m + \ldots + (p+1)m) \\
&= \frac{N(N+1)}{2} + m\left(\frac{p^2+3p}{2} n + q - 1\right)
\end{aligned}$$

Thus, we have to choose n such as to minimize the two functions

$$\begin{aligned} N_{div}(n,N) &= \frac{1}{2}m(n)\cdot(p+1)\left(N+q+m(n)\cdot p+m(n)-1\right), \\ N_{mult}(n,N) &= \frac{1}{2}N(N+1)+m(n)\left(\frac{p^2+3p}{2}n+q-1\right), \end{aligned}$$

where

$$p=\left\lfloor\frac{N}{n}\right\rfloor, \quad q=N \mod n,$$
$$m(n)=\left\lceil\log_r\left((n+1)\,\rho^2-\rho+1\right)\right\rceil.$$

For instance, let $r=10$, $\rho=6$, then

$$m(n)=\begin{cases} 2, & \text{if} \quad n=1 \\ 3, & \text{if} \quad 2\le n\le 26 \\ 4, & \text{if} \quad 27\le n\le 276 \\ 5, & \text{if} \quad 277\le n\le 2776 \\ \cdots & \cdots \quad \cdots \end{cases}$$

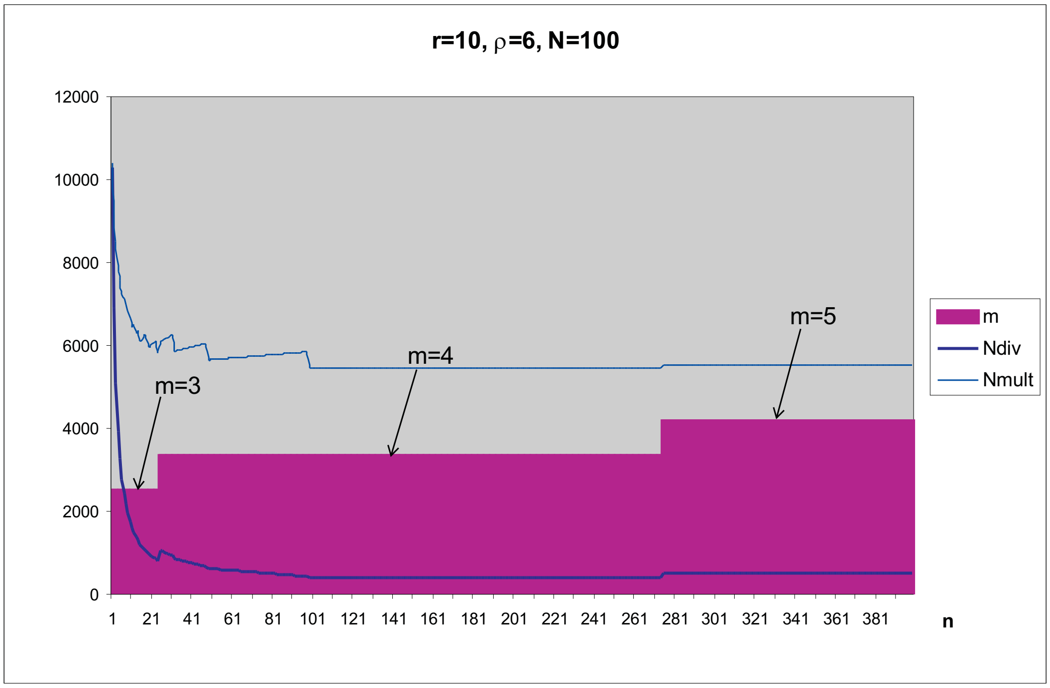
and the corresponding minimal values of N_{div} for $N=100$ are

$$N_{div}(1,100)=\frac{1}{2}\cdot 2\cdot 101\cdot 301=30401$$
$$N_{div}(26,100)=\frac{1}{2}\cdot 3\cdot 4\cdot 133=798$$
$$N_{div}(276,100)=\frac{1}{2}\cdot 4\cdot 1\cdot 203=406$$
$$N_{div}(2776,100)=\frac{1}{2}\cdot 5\cdot 1\cdot 204=510$$

These data have been summarized in graphical form in Fig. 4 as plots of N_{div} and N_{mult} versus n for $N=100$. It is apparent that $N_{div}(n)$ and $N_{mult}(n)$ behave similarly for other values of N, r and ρ. One can see that the optimal value of N_{div} is attained when $n=N+1$, in which case $p=0$, $q=N$, and the total number of divisions is $N_{div}(N+1,N)=mN+m(m-1)/2$. Since the number N of required precision digits is generally unknown in advance, it is reasonable to choose some *fixed* value of n that would ensure reasonable performance of the algorithm for all N. It is also clear that we may only choose n out of

$$n_k=\max\left\{n\in\mathbb{N} \mid m(n)\le k\right\},\ k\in\mathbb{N},$$

Figure 4: Choice of n for multiplication — the number of integer multiplications N_{mult} and divisions N_{div} vs. n calculated for $N = 100$ precision digits



because if $n' > n''$ and $m(n') = m(n'')$, we have $N_{div}(n') \leq N_{div}(n'')$.

In principle, the larger the value of n, the better; except when n is vastly larger than N, the number of operations N_{op} will continue to grow with N (due to the increasing of m). On the other hand, choosing a large value of n would imply large values of the sequence entries (up to $(n+1)\,\rho^2$ — see (14)) which, if exceeded the threshold for representing integers (usually the size of the machine word), would result in slower integer operations. The values of n corresponding to $m = 2$ are obviously inadequate, resulting in an unnecessarily large number of operations (e.g. Avizienis's algorithm), but any of the numbers $n_3, n_4, \ldots$ are equally suitable for the value of n (in terms of operation counts). Of course, the larger the m in n_m, the more normalizations (and therefore more space to hold the intermediate results) will be required. In our implementation, we have used $n = n_3$ (e.g. for $r = 10$ and $\rho = 6$, $n = n_3 = 26$).

6.3 Division

Division can be analyzed in much the same way as multiplication, and is also quadratic. For simplicity's sake, we shall assume that the partial remainders are normalized *fully* — as remarked above, the actual time estimates will only be better.

By (18), to determine the N-th digit q_N of the quotient, the division algorithm must compute the first digit of the N-th partial remainder P_N, which according to (17), involves evaluation of P_{N-1} and D to 3 digits (an extra digit is required because of the multiplication by r), that in turn demands P_{N-2} and D to 5 digits, and the domino effect applies to the rest of the partial remainders, so that P_0 will be evaluated to $2N+1$ digits. In summary, we will have $N^2 = 1 + 3 + \ldots + 2N - 1$ integer divisions and additions from the normalization of $P_N, P_{N-1}, \cdots, P_1$, plus N more divisions from the digit selection guesswork in (18), as well as $N(N+2) = 3 + 5 + \ldots + (2N+1)$ additions and multiplications of the quotient digits q_n by the digits of the divisor D in (17).

7 Elementary functions

In this section, we shall discuss the evaluation of elementary functions on exact real numbers. Functions of real variables that can be defined for normalized signed-digit radix-r representations are precisely those for which there exist left-to-right algorithms defined on representations. These algorithms must work in an on-line fashion: digit-by-digit, most significant digit first, inputting digits of the argument(s) and outputting digits of the result with bounded delay. The question one should ask himself when defining a function on representations is whether, given more digits of the argument, one can produce more digits of the result. In particular, only continuous functions on exact reals are computable.

7.1 Absolute value

The absolute value is probably one of the simplest functions definable on the real numbers. In floating-point systems, all that is required for its computation is changing a number's sign bit, if need be — an operation so trivial that it is never even considered as such.

In exact real arithmetic systems, however, there is no algorithm for deciding whether or not two infinite sequences represent the same number. In particular, the predicates $=$, $<$ and $>$ are non-computable, and in general one cannot even check a number to see whether it is positive, negative, or zero.

Nonetheless, the absolute value function is *definable* on exact reals. Let us show that if the signed-digit radix-r system used is *not* maximally redundant, i.e. $\rho < r - 1$, the sign of a number is determined by the sign of the first non-zero entry of its mantissa. Indeed, if a_k is the first non-zero element of $(a_n)_{n\in\mathbb{N}_0}$, then

$$r^{-k}\left(a_k - \frac{\rho}{r-1}\right) \leq \sum_{n=0}^{\infty} a_n r^{-n} \leq r^{-k}\left(a_k + \frac{\rho}{r-1}\right),$$

and if the system is not maximally redundant, all of these numbers have the same sign as a_k (provided $a_k \neq 0$). From this also results the conclusion that in non-maximally-redundant systems zero is represented *uniquely* (up to differences in exponents).

The algorithm for evaluation of the absolute value is now obvious:

$$\mathrm{abs}\,(a_0,\, a_1,\, \cdots,\, a_n, \cdots) = \begin{cases} 0 : \mathrm{abs}\,(a_1,\, a_2,\, \cdots,\, a_n, \cdots), & \text{if } a_0 = 0 \\ (a_0,\, a_1,\, \cdots,\, a_n, \cdots), & \text{if } a_0 > 0 \\ (-a_0,\, -a_1,\, \cdots,\, -a_n, \cdots), & \text{otherwise} \end{cases}$$

and its complexity is that of negation.

7.2 Minimum and maximum

It may come as a surprise to some to learn that while the comparison operators $<$ and $>$ are clearly non-computable on exact reals, the functions minimum and maximum are. This is most readily seen from the relations

$$\begin{aligned} \min\,(a,b) &= \frac{a+b-|a-b|}{2}, \\ \max\,(a,b) &= \frac{a+b+|a-b|}{2}, \end{aligned}$$

which involve only computable functions: addition, subtraction, absolute value, and division by 2.

The implications of computability of min and max are non-trivial: for example, we can sort lists of exact real numbers using sorting algorithms based upon max and min, rather than upon $<$ and $>$ (such as Batcher's merge sort).

7.3 Square root

The square root function is singled out because of its simplicity and amenability to implementation with little additional overhead beyond that of the basic arithmetic operations. It is also almost the only commonly used function that is evaluated iteratively. The algorithm that we will describe is the direct analogue of that for division and produces n digits of the result in n cycles, at a rate of one digit per cycle. Such pseudo-division methods can also be extensible to higher degrees, although roots of order greater than three are usually evaluated by the same methods as x^y for arbitrary y, using exponents and logarithms, and even cube-root functions are somewhat uncommon in function libraries. Our primary emphasis will therefore be on evaluation of $\sqrt{x}$.

Suppose that we want to evaluate $y = \sqrt{x}$ in radix r. Let x be given by a normalized signed-digit sequence $X = (x_0, x_1, \ldots, x_n, \ldots)_r$ with $x_0 > 0$, $|x_n| \leq \rho$, $n \in \mathbb{N}$, and exponent e, so that $x = r^e \cdot \sum\limits_{n=0}^{\infty} x_n r^{-n}$, $x > 0$. Then

$$y = \begin{cases} r^{e/2} \cdot \sqrt{X}, & \text{if } e \text{ is even} \\ r^{(e-1)/2} \cdot \sqrt{rX}, & \text{if } e \text{ is odd} \end{cases}$$

Let $Y = (y_0, y_1, \ldots, y_n, \ldots)$ be a mantissa of y such that $|y_n| \leq \rho$, $n \in \mathbb{N}$. Denote

$$Y_n = y_0 + y_1 r^{-1} + \ldots + y_n r^{-n}$$

so that $Y = \lim\limits_{n\to\infty} Y_n$. Consider the scaled partial remainders

$$P_0 = X,\ P_n = r^n \left(X - Y_{n-1}^2\right),\ n \in \mathbb{N}.$$

Observing that $Y_n = Y_{n-1} + y_n r^{-n}$, we can rewrite the next partial remainder as

$$P_{n+1} = r^{n+1}\left(X - Y_n^2\right) = r\left(P_n - 2y_n Y_{n-1} - y_n^2 r^{-n}\right)$$

The square root algorithm is based on the above recurrence relation, each iteration of which consists of two subcomputations:

1) Determination of the result digit y_n using a digit selection function s, which has P_n and Y_{n-1} as arguments:

$$y_n = s(P_n, Y_{n-1})$$

2) Formation of P_{n+1} from P_n, Y_{n-1} and y_n:

$$P_{n+1} = r\left(P_n - 2y_n Y_{n-1} - y_n^2 r^{-n}\right)$$

If care is exercised in choosing y_0 and the selection function s is such that

$$\ldots \leq |P_{n+1}| \leq |P_n| \leq \ldots \leq |P_1| \leq |X|\,,$$

the algorithm will converge as

$$\left|X - Y_n^2\right| = \frac{1}{r^{n+1}}\,|P_{n+1}| \leq \frac{1}{r^{n+1}}\,|X|\,.$$

As in the case of division, we make guesses about the digits y_n based on the most significant digits of the current remainder P_n and the current approximation Y_{n-1}. Although the guessed digits may be incorrect in some cases, no correction steps would be needed if a redundant signed-digit representation of P_n was used. In particular, it can be shown that the following digit selection function

$$y_n = \left\lfloor \frac{|p_{n0}|}{2y_0} \right\rfloor \cdot \operatorname{sgn} p_{n0},$$

where p_{n0} is the integer part of P_n and $y_0 = \left\lfloor \sqrt{x_0} \right\rfloor$ the first digit of Y, is reliable enough as long as $y_0 \geq \lfloor \sqrt{r} \rfloor$, i.e. $x_0 \geq r$.

8 Summary

The foregoing analysis suggests that, notwithstanding the claim made by Boehm and Cartwright, the representation of exact real numbers by lazy infinite sequences of signed digits in a radix-r system can lead to reasonably efficient implementations of constructive real arithmetic. In particular, the algorithms presented here largely overcome what they called the granularity effect. For the sake of simplicity, our implementation used $r = 10$ and $\rho = 6$, and was written in the functional programming language Miranda[5] [17]. Choosing the radix to be a large number, and using a compiled language such as C/C++ or Java, would yield a large dividend in efficiency, which could be improved even further on a multi-processor system if normalization and other functions were multi-threaded.

What are the advantages and shortcomings of positional arithmetic systems as opposed to others, e.g. functional ones? Laziness is certainly an advantage — a demand-driven system only computes those numbers that are needed, and only to the precision required. It also avoids recomputing the elements calculated earlier, so that each digit gives a better approximation to the number being computed. Conversion of numbers into redundant

[5] Miranda is a product and trademark of Research Software Limited.

form and decoding them back into conventional form is also very simple, as the "conventional" systems are also positional radix-r systems.

Among the shortcomings we can name the problem of choosing the subset of *finitely representable* numbers. The availability of a subset in which numbers are represented finitely is important for many reasons, not least of which is the need to compute equality tests. For example, it is clear that all integers must be finitely represented. After the integers, the rationals seem to be the best candidate for such subset, but any efficient implementation based on the representation of the rationals as repeating radix-r numbers must overcome nontrivial technical challenges, such as being able to recognize a state of computation that has occurred before, or having to deal with finite representations of very great lengths. Evaluation of transcendental functions in radix-r systems is also problematic, as there are no obvious digit-by-digit algorithms that are both simple and efficient.

All these factors must be considered prior to choosing a representation of the exact reals most suitable for a given problem. Situations where one might want to use infinite precision arithmetic include e.g. testing an algorithm to determine whether it suffers from a numerical instability, or computing some numbers to high precision to serve as reference values for conventional methods.

Acknowledgements

Support for this research was provided by EPSRC Grant Ref. GR/L03279. The author wishes to thank Professor David Turner for his numerous suggestions and guidance in this project all the way from inception to completion. Credit is also due to Carl Pixley, whose early unpublished work at Burroughs Corporation's Austin Research Center was a great source of inspiration for this research.

References

[1] Avizienis, A., "Binary-compatible signed-digit arithmetic", *Proc. AFIPS Fall Joint Comp. Conf.*, 1964, pp. 663-672

[2] Avizienis, A., "Signed-digit number representations for fast parallel arithmetic", *IRE Trans. El. Comp.*, Vol. EC-10, No. 3, Sept. 1961, pp. 389-400

[3] Boehm, H.-J., Cartwright R., et al., "Exact Real Arithmetic: A Case Study in Higher Order Programming", *Proceedings 1986 ACM Conference on LISP and Functional Programming*, ACM Press (August 1986), pp. 162-163

[4] Boehm, H. and Cartwright, R., "Exact Real Arithmetic: Formulating Real Numbers as Functions", in *Research Topics in Functional Programming*, (ed) D. A. Turner, Addison-Wesley, 1990

[5] Edalat, A. and Potts, P. J., "A New Representation for Exact Real Numbers", Electronic Notes in Theoretical Computer Science, 6 (1997), URL: `http://www.elsevier.nl/locate/entcs/volume6.html`

[6] Ercegovac M. D., "On-line Arithmetic: an Overview", SPIE Vol. 495, *Real Time Signal Processing VII*, 1984, pp. 86-93

[7] Kaganovsky, A., "Computing with Exact Real Numbers in a Radix-r System", Electronic Notes in Theoretical Computer Science, Vol. 13 (1998)

[8] Ménissier-Morain, V., "Arbitrary Precision Real Arithmetic: Design and Algorithms", *Unpublished manuscript*

[9] Muller, J.-M. "Arithmétique des ordinateurs", *Etudes et recherches en informatique*, Masson, 1989

[10] Myhill, J., "What is a real number?", *American Mathematical Monthly*, Sept. 1972, pp. 748-754

[11] Pixley, C. P., "Demand-Driven Arithmetic", Burroughs Corporation Austin Research Center, Internal Report ARC 82-18, Nov. 1982

[12] Robertson, J. E., "A New Class of Digital Division Methods", *IRE Trans. El. Comp.*, Vol. EC-7, Sept. 1958, pp. 218-222

[13] Schwarz, J., "Implementing Infinite Precision Arithmetic", *Proc. 9th Symposium on Computer Arithmetic*, September 6-8, 1989, pp. 10-17

[14] Smith, D. M., "A Multiple-Precision Division Algorithm", *Mathematics of Computation*, Vol. 65, No. 213, Jan. 1996, pp. 157-163

[15] Stromberg, K. R., *Introduction to Classical Real Analysis*, Wadsworth, Inc., 1981

[16] Turing, A. M., "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proc. London Math. Soc.*, Ser. 2, Vol. 42, Nov. 12, 1936, pp. 230-265. A correction, *ibid.*, Vol. 43, pp. 544-546

[17] Turner, D.A., "An Overview of Miranda", SIGPLAN Notices, Vol. 21, No. 12, pp. 158-166 (December 1986)

[18] Turner, D.A., "SASL Language Manual", St. Andrews University, Department of Computational Science Technical Report, Dec 1976

[19] Vuillemin, J. E., "Exact Real Computer Arithmetic with Continued Fractions", *IEEE Transactions on Computers*, Vol. 39, No. 8, August 1990, pp. 1087-1105

[20] Wiedmer, E., "Computing with Infinite Objects", *Theoretical Computer Science*, Vol. 10 (1980), pp. 133-155

[21] Wiedmer, E., "Exaktes Rechnen mit reellen Zahlen und anderen unendlichen Objekten", Diss. ETH 5975, Zurich (1977)