

Arbitrary precision real arithmetic: design and algorithms

Valérie Ménissier-Morain^{*†}

May 21, 1995

Abstract

We describe here a representation of computable real numbers and a set of algorithms for the elementary functions associated to this representation. This arithmetic is the first real arithmetic with mathematically proved algorithms.

1 Introduction

1.1 Motivation

We try to determine here what should be an arithmetic for a modern and reliable programming language.

The exactitude of the arithmetic is clearly an essential feature of a reliable programming language, but we show here that even a floating point representation with variable length, as it exists in symbolic computation softwares, is insufficient. Furthermore this arithmetic gives its users some wrong ideas, as disastrous as the round-off errors themselves. The floating point arithmetic is obviously not an exact arithmetic: each partial result is systematically rounded off, and these successive round-off errors may lead to completely erroneous answers for ill-conditioned problems like the computation of $1/(y-x)$ where x is "much greater" than 1 (for example $x = 1.234567890e10$) and $y = x + 1$. For this particular computation, a single precision floating point computation induces an error due to a division by zero, but an exact computation yields a result, 1, that is furthermore exactly represented in a floating point arithmetic. However, in order to handle rational or real numbers, one represents them generally with the floating point arithmetic supplied by the computer with a fixed number of significant digits. The programmer is usually aware of these round-off problems, but nevertheless, he remains confident of the computed results because of "intuitive properties" of the floating point arithmetic. Among these pseudo-properties, one can cite:

1. even if the result is not rigorously exact, it is certainly close to the exact result, in particular the order of magnitude is supposed to be preserved;
2. a few floating point operations can only induce a slight inaccuracy in the computed result;
3. if more accurate results are needed, it is sufficient to increase the size of the number representation before the computation (for instance many programming languages offer double precision floating point numbers).

The following sequence, discovered by Jean-Michel Muller [15], is a counter-example to all these ideas. Let $(a_n)_{n \in \mathbb{N}}$ be the sequence defined recursively by

$$a_0 = \frac{11}{2}, a_1 = \frac{61}{11}, a_{n+1} = 111 - \frac{1130 - 3000/a_{n-1}}{a_n}.$$

^{*}INRIA, B.P. 105, F-78153 Rocquencourt Cedex, France. *E-mail*: Valerie.Menissier@inria.fr

[†]Université d'Évry Val d'Essonne, Département d'Informatique, 4 Boulevard des Coquibus, 91025 Évry Cedex.

With single precision floating point arithmetic, we get the following values:

n	2	5	6	7	8	10	11	12
a_n	5.6	5.6	4.3	-29.0	125.7	100.1	100.0	100.0

Thus the sequence seems to converge very quickly to 100.0. In fact one can prove easily that

$$a_n = \frac{6^{n+1} + 5^{n+1}}{6^n + 5^n},$$

from which we deduce immediately that the limit of this sequence is 6. This result is far from 100.0 and the floating point arithmetic does not preserve the order of magnitude of the limit.

Moreover, the corresponding mathematical values of this sequence (rounded off to 1 decimal place) are as follows:

n	2	5	6	7	8	10	11	12
a_n	5.6	5.7	5.7	5.8	5.8	5.9	5.9	5.9

Thus as soon as n is greater than 5 the floating point results are completely irrelevant, even though the computation of a_n leads to very few operations (precisely $2n$ divisions and $2n-2$ subtractions). The common idea that "a few operations induce a slight inaccuracy" is false: for a_n the computation diverges for a_6 after only 12 divisions and 10 subtractions.

By the way, one can notice that the convergence of the floating point representation of $(a_n)_{n \in \mathbb{N}}$ to 100.0 is fast and stable (for $n > 10$, a_n is always very close to 100.0): these properties ensure neither the accuracy nor the order of magnitude of the computed limit. This phenomenon can be explained by the fact that the function

$$x \mapsto 111 - \frac{3000}{x}$$

has three fix points: 5, 6 and 100, and 100 is an attractive fix point, while 5 and 6 are repulsive one's, particularly 6.

According to the third common idea, the preceding phenomenon may be due to the lack of precision in the floating point representation of numbers. Let's try a more powerful floating point arithmetic that allows the programmer to fix the number of significant digits used during computation (as in the symbolic computation system MAPLE [8]). Here are the values obtained for a_n with various precisions:

precision	a_{10}	a_{20}	a_{30}	a_{40}	a_{50}
10	147.26	100	100	100	100
20	5.86	99.7	100	100	100
30	5.86	5.97	100	100	100
40	5.86	5.97	5.994	100	100
50	5.86	5.97	5.996	5.92	100

The sequence stubbornly converges to 100.0. We conjecture that with p decimal digits of precision during computation, the value a_{2p} is almost equal to 100.0 (and so for a_n if $n \geq 2p$ since the sequence reaches the basin of attraction of 100). Thus, whatever the precision one starts with, the floating point arithmetic fails to perform a computation which respects the mathematical limit of the sequence $(a_n)_{n \in \mathbb{N}}$, in opposition to the third common idea.

One can argue that the computation of the limit of a sequence requires potentially an infinity of operations and that for any finite computation, one can estimate the order of magnitude of all intermediate results and deduce from this a lower bound for the required precision for floating point arithmetic. But even this viewpoint proves that to inspire real confidence in results obtained with floating point arithmetic, one must analyze the computation and the evolution of the precision. This analysis is so tedious that in practice the programmers ignores it. Yet the programmer desires reliable arithmetic results, so the correction must be ensured by the programming language.

The most usual solution to this question is interval analysis [16, 17, 1, 10]. The computation is performed using floating point arithmetic and propagates during all this computation an upper bound of the round-off error for rational operations, according to the IEEE-754 norm, so that one obtains at the end of the computation a floating point result and an upper bound for the round-off error on this result. For the preceding sequence, a computation with such an arithmetic will indicate that the upper bound for the round-off error on this result is very big. However if one wants to compute the exact result or a result as close as one want of the exact value, this solution is not satisfactory.

We give here an answer to the programmer who needs reliable arithmetic results for which the correction is ensured by the programming language and not by an arithmetic that indicates an upper bound for the distance between the exact value and the obtained result. The programmer indicates an upper bound for the final round-off error and this bound is respected all along the computation, even if it requires a very high precision in a particular step of the computation.

The ML language was designed for safe programming and so should ensure safety in numerical programming. So we implement a small prototype of an arbitrary precision library in this language according to the work described here.

1.2 History of the problem

Wiedmer proposed in 1977 a solution for real number computations in [26, 27, 28]. This solution can be considered as unbounded on-line arithmetic. However, Wiedmer proposed only an algorithm to add real numbers. These ideas were studied again and extended by Boehm in [7, 6]. Computable real numbers are represented by an infinite sequence of digits in a given base B . For such a representation the digits of the results are produced "from left to right", beginning with the most significant digits, in opposition to the usual algorithms for addition and multiplication for example. Particularly, Avizienis in [2] and Wiedmer in [28] proved that an addition algorithm "from left to right" implies the redundancy of the representation: for example, digits are in the integer interval $[-B + 1, B - 1]$ rather than the classical interval $[0, B - 1]$. The idea is that one is compelled to anticipate what will be the next digits of the arguments of the addition algorithm and for example to overestimate by one the absolute value of the sum, even if one needs to correct this trend on the next produced digit by a negative sign. We studied this representation, described and proved algorithms for rational operations, but we did not work out so far algorithms for transcendental functions. Perhaps the Cordic algorithms described by Lin and Sips in [11] may be used to compute these functions. The incrementality is a natural good point for this representation: if one need some more digits, one starts from the list of already computed digits rather than from the beginning of the computation. However, apart from the lack of well-integrated algorithms for transcendental functions, the algorithms for rational operations are intricate and rather inefficient.

In [7, 6], Boehm studied a more natural representation. This representation is designed for almost automatic evaluation of round-off errors in programs written in Fortran. In his implementation, the classical operations on floating point numbers are replaced by exact operations on real numbers, then some numerical tests of small size are performed with each arithmetic, so that if a floating point result does not correspond to the expected value, one can attribute this computation error either to a round-off error if the real result is correct or to an error in the implementation of the algorithm by the program if the real result is wrong. Boehm describes algorithms for addition and multiplication on this representation.

Boehm developed an implementation for each of these two representations. The comparison of the running times indicates clearly that the second one is much faster than the first one.

We studied this second representation and now we propose a complete and entirely proved set of algorithms for all elementary functions. This work leads to an implementation in the Caml implementation of the ML language.

Finally, in [23, 24, 25], Vuillemin represents real numbers by continued fractions, with the underlying idea that continued fractions are the "closest" rational numbers to the real numbers. However, apart from the fact that these algorithms are principally not proved, this representation is rather inadequate to current computers so it is inefficient. The author implements a complete prototype for this representation that

exhibits poor running times despite the natural incrementality of the method.

We present these three representations with all details in [14]. We describe in this article the second representation mentioned above.

We first recall the main properties of computable real numbers. We deduce from one definition, among the three definitions of this notion, a representation of these numbers as sequence of B -adic numbers and then we describe algorithms for rational operations and transcendental functions for this representation. Finally we describe briefly the prototype written in Caml.

2 Computable real numbers

Unlike Bishop [5], Martin-Löf [12] and Stolzenberg [19], we use here the classical real analysis as the framework to state properties of computable real numbers*, as in Rice fundamental paper [18].

2.1 Definitions

Let us note $f_{\mathbb{Q} \rightarrow \mathbb{N}}$ a recursive bijection from \mathbb{Q} to \mathbb{N} . We define first the notion of *recursive Cauchy sequence* for rational numbers and for intervals with rational bounds.

Definition (Recursive Cauchy sequence)

1. A sequence of rational numbers $(q_n)_{n \in \mathbb{N}}$ is called recursively enumerable if the function $n \mapsto f_{\mathbb{Q} \rightarrow \mathbb{N}}(q_n)$ is recursive.
2. A sequence of intervals with rational bounds $(I_n = [i_n, s_n])_{n \in \mathbb{N}}$ is called recursively enumerable if the sequences $(i_n)_{n \in \mathbb{N}}$ and $(s_n)_{n \in \mathbb{N}}$ are recursively enumerable sequences of rational numbers.
3. A sequence of rational numbers $(q_n)_{n \in \mathbb{N}}$ is called a recursive Cauchy sequence if it is recursively enumerable and there exists a recursive function g , the function of convergence of the sequence, such that for all strictly positive integer N and all pair of integers n and m with $n \geq m \geq g(N)$, we have:

$$|q_n - q_m| < \frac{1}{N}.$$

We can now give several definitions of the notion of *computable real numbers*.

Definition (Computable real numbers, first definition)

A real number r is a computable real number if and only if there exists a recursively enumerable sequence $(I_n = [i_n, s_n])_{n \in \mathbb{N}}$ of nested intervals with rational bounds, enclosing r and the sequence of the lengths of these intervals $(|s_n - i_n|)_{n \in \mathbb{N}}$ converges to 0.

Definition (Computable real numbers, second definition)

A real number is a computable real number if and only if it is the limit of a recursive Cauchy sequence of rational numbers.

We will now define the related notion of *finite B -adic numbers* for a given base B and deduce the notion of *B -approximable real number*.

Definition (Finite B -adic number)

If B is an integer greater than or equal to 2, a rational number r is called a finite B -adic number if there exists two integers p and q such that $r = p/B^q$ and q is a positive integer.

We define now the notion of *B -approximable real number*.

*We use here "computable real numbers as Turing did [20] rather than the expression "recursive real numbers" employed by Rice, but these two terms are of course equivalent

Definition (*B*-approximable real number)

A real number x is called *B*-approximable if there exists a recursive function g such that, for all integer N , $g(N)$ is a finite *B*-adic number and

$$|x - g(N)| < \frac{1}{B^N}.$$

These different definitions lead naturally to the following property:

Property

The two definitions of the notion of computable real numbers and the definition of *B*-approximable real number are equivalent.

Proof: This property and the following ones are all proved in [14]. We will generally indicate the idea of the proof when the argument is not only an obvious construction or the suitable definition used for the proof. \odot

And now a last definition that will be useful afterwards.

Definition (Recursively enumerable sequence of real numbers)

A sequence $(x_n)_{n \in \mathbb{N}}$ of computable real numbers is called recursively enumerable if there exists two recursive functions g and h such that $(g(n, k))_{k \in \mathbb{N}}$ is a recursive Cauchy sequence of rational numbers that converges to x_n as k tends to infinity, with $k \mapsto h(n, k)$ as convergence function.

We denote up to the end of this section by \mathcal{R} the set of computable real numbers and as usual by \mathbb{R} the set of real numbers.

2.2 Properties of \mathcal{R}

The set \mathcal{R} has the following properties:

Algebraic property of \mathcal{R} : The set \mathcal{R} with the addition and the multiplication of \mathbb{R} is an archimedean commutative field.

Topological property of \mathcal{R} : Each recursive Cauchy sequence of elements of \mathcal{R} has a limit in \mathcal{R} .

Analytical property of \mathcal{R} : \mathcal{R} is closed for elementary functions.

Cardinal of \mathcal{R} : \mathcal{R} is a denumerable subset and is dense in \mathbb{R} .

Proof: We use the first definition of computable real numbers to prove the algebraic and the analytic properties. The proof of the analytic property uses essentially the lemma "If $(a_n)_{n \in \mathbb{N}}$ is a recursively denumerable decreasing sequence of positive rational numbers with a null limit to infinite, the sum of the alternate series of general term $(-1)^n a_n$ is a computable real number". The proof of the topological property and of the cardinality of \mathcal{R} uses naturally the first definition of computable real numbers. Finally \mathcal{R} is dense in \mathbb{R} because \mathbb{Q} is dense in \mathbb{R} and is included in \mathcal{R} . \odot

2.3 Indecidability theorems about \mathcal{R}

Rice proved the following result:

Theorem (Rice)

There exists no general algorithm to determine whether a computable real number is zero or not.

Proof: As a first point, if such an algorithm exists, then we can decide for all recursive function from \mathbb{N} into $\{0, 1\}$ if this function is single-valued or not. As a second point, the question of the stopping of a Turing machine can be describe by the question of the single-valuation of a recursive function from \mathbb{N} into $\{0, 1\}$. Then since we know that the question of the stopping of a Turing machine is undecidable, the result follows. \odot

We deduce from this the following results:

Corollary

1. *There exists no general algorithm to determine the image of a computable real number by a function with a discontinuity at this point.*
2. *There exists no general algorithm to determine if a computable real number is greater than another one.*
3. *There exists no general algorithm to determine the integer part of a computable real number.*
4. *There exists no general algorithm to determine if a computable real number is rational.*

Proof: Obvious. \odot

A consequence of the third proposition of this corollary is that one cannot determine exactly the continued fraction expansion or the development in a given base of any computable real number.

However, one should not attach an excessive importance to these impossibilities because according to the last definition of computable real numbers, any computable real numbers may be known to a precision of B^{-n} in a given base B . As far as the comparison is concerned, the following theorem establishes that if two computable real numbers differ, then there is an algorithm that indicate which one is the greater one and at which rank their definition sequences differ.

Property

Let $A = (a_n)_{n \in \mathbb{N}}$ and $B = (b_n)_{n \in \mathbb{N}}$ be two recursive Cauchy sequences of rational numbers with distinct respective limits a and b , then there exists an algorithm to compare a and b that terminates.

Proof: Obvious argument, technical details. \odot

3 Description of a representation of computable real numbers with particular sequences of B -adic numbers

According to the third definition, computable real numbers are considered here as B -approximable real numbers. Precisely reals numbers will be represented by B -adic numbers and as in Boehm's work, we represent the B -adic numbers by longer and longer integer corresponding to the numerator of B -adic approximations more and more precise.

It is well-known that the limit of a sequence of B -adic numbers $(a_n/B^{k_n})_{n \in \mathbb{N}}$ is also the limit of this other sequence of B -adic numbers $(b_n/B^n)_{n \in \mathbb{N}}$ with $b_n = \lfloor a_n B^{n-k_n} \rfloor$. This second sequence is interesting because the denominator of each B -adic is exactly B raised to its rank in the sequence so we need only the sequence of integers $(b_n)_{n \in \mathbb{N}}$ to represent the limit of this sequence.

Thus we approximate a computable real number r with a sequence of integers $(c_n)_{n \in \mathbb{N}}$ such that $|r - c_n B^{-n}| < B^{-n}$ for all integers n .

We will now describe this representation and a complete set of algorithms to compute elementary functions for this representation, using the corresponding algorithms for rational numbers.

3.1 Definitions and generalities

Let B be a given base, i.e. an integer greater than or equal to 2. A computable real number is represented by a sequence of integers that satisfy the following property:

Definition (Bounds property)

Let x be a computable real number, for all integer p , the bounds property of x by p for order n is characterized by the following inequality $|x - pB^{-n}| < B^{-n}$ i.e. $(p - 1)B^{-n} < x < (p + 1)B^{-n}$.

We authorize negative indices for practical reasons because sometimes we need only to know the order of magnitude of a real number rather than its integer part. The bounds property apply easily to negative

indices and it saves some time during the computation. We will now express some properties of the integers that satisfy the bounds property for a given real number and a given order.

Property

Let x be a computable real number, n an integer and p be an integer. Suppose that the bounds property of x by p for order n is satisfied. Then $p = \lfloor B^n x \rfloor$ or $p = -\lfloor B^n(-x) \rfloor$.

Proof: Technical verification using intensively the definition of the function $x \mapsto \lfloor x \rfloor$. \odot

The real numbers that we will consider in this section are computable real numbers x represented by sequences of integers $(x_n)_{n \in \mathbb{N}}$ such that the bounds property for x by x_n for order n is satisfied.

We will now define the *sign* function before we express the following property that describes the relations between the integers that satisfy the bounds property for x and $|x|$ for the same order.

Definition (sign)

The function *sign* is defined from \mathbb{R} to $\{-1, 0, 1\}$ with the usual following equality:

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$$

Practically, for each non-zero real number x and for each integer n , the sign of x is the sign of each non-zero value x_n and its computation terminates.

Property

Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$ and n be an integer, $|x_n|$ satisfies the bounds property of $|x|$ for order n and if p is a positive integer that satisfies the bounds property of $|x|$ for order n , then the integer q defined by $q = \text{sign}(x) \times p$ satisfies the bounds property of x for order n .

Proof: We study the three following cases for x_n or q : $0, \geq 1$ and ≤ -1 . In the first case, all implied integers are zero and the property follows. In the two other cases we know the sign of x and the property follows too. \odot

We have also the following technical properties:

Properties

1. Let x, α be two real numbers and n be an integer. If $\alpha B^{-n} < x < (\alpha + 1)B^{-n}$, then $\lfloor \alpha \rfloor + 1$ and $\lceil \alpha \rceil$ satisfy the bounds property of x for order n .
2. Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, n and m be integers such that $n \leq m$, then the integer $\lfloor \frac{x_m}{B^{m-n}} \rfloor$ satisfies the bounds property of x for order n .

Proof: Technical manipulation of inequalities \odot

We will now define the *msd* function that indicates the order of magnitude of a real number.

Definition (msd)

The function *msd* ("most significant digit") is defined from \mathbb{R} to \mathbb{Z} for all real number x represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, by the equality $\text{msd}(x) = \min_{n \in \mathbb{Z}} (|x_n| > 1)$.

Practically, the function *msd* is recursively computed and does not terminate for zero. This function satisfies the following properties:

Properties

1. For all non-zero real number x , $\text{msd}(x)$ exists and is unique (at the exact time of its computation, see the remark below), with $2 \leq |x_{\text{msd}(x)}| \leq 2B$ and $\text{msd}(x) = -\lfloor \log_B |x| \rfloor$ or $\text{msd}(x) = -\lfloor \log_B |x| \rfloor + 1$.

2. For all non-zero real number x and all integer $n < \text{msd}(x)$, we have $|x_n| \leq 1$.
3. For all non-zero real number x and all integer $n \geq \text{msd}(x)$, then

$$B^{n-\text{msd}(x)} \leq |x_n| \leq B^{n-\text{msd}(x)}(2B + 1)$$

and

$$1 \leq \left| \left\lfloor \frac{x_n}{B^{n-\text{msd}(x)}} \right\rfloor \right| \leq 2B + 1.$$

Proof: Technical and boring proof. \odot

4 Algorithms for the usual elementary functions

4.1 Introduction to algorithms for the computation of elementary functions on \mathcal{R}

We will now describe algorithms for computing elementary functions on \mathcal{R} . For each of the following algorithms, we have established in [14] a theorem of correction as the following one:

Theorem (Correction of the algorithm for computing a function f on computable real arguments $args$)

Let $(\overline{f(args)})_{n \in \mathbb{N}}$ be the sequence computed by the algorithm for $f(args)$, then, for all integer $p \in \mathbb{N}$, the bounds property of $f(args)$ by $\overline{f(args)}_p$ for order p is satisfied.

4.2 Algorithms for rational operations

We will now describe the representation of the image of any computable real number by an elementary function and we begin with the heart of these algorithms: the representation of rational numbers.

Representation of rational numbers

Each rational number q is represented by the sequence of integers $(q_n)_{n \in \mathbb{N}}$ such that q_n is defined, for all integer n by the equality:

$$q_n = \lfloor B^n q \rfloor - \lfloor B^n (-q) \rfloor.$$

Proof: Immediate. \odot

Addition of real numbers

Let x and y be two real numbers represented by the sequences $(x_n)_{n \in \mathbb{Z}}$ and $(y_n)_{n \in \mathbb{Z}}$ respectively, the sum of these two numbers $x + y$ is represented by the sequence $(\overline{x+y}_n)_{n \in \mathbb{Z}}$ such that:

$$\overline{x+y}_n = \left\lfloor \frac{x_{n+1} + y_{n+1} + \frac{B}{2}}{B} \right\rfloor.$$

Theorem ([])

Correction of the addition algorithm] For all integer n , $\overline{x+y}_n$ satisfies the bounds property of x for order n .

Proof:(Complete proof)

Let n be an integer. By definition of $x \mapsto \lfloor x \rfloor$, we have

$$(\overline{x+y}_n - 1)B^{-n} \leq \left(\frac{x_{n+1} + y_{n+1} + \frac{B}{2}}{B} - 1 \right) B^{-n}$$

and

$$(\overline{x+y}_n + 1)B^{-n} > \left(\frac{x_{n+1} + y_{n+1} + \frac{B}{2}}{B} \right) B^{-n}.$$

Thus

$$(\overline{x+y}_n - 1)B^{-n} < \frac{x_{n+1} + y_{n-1} - \frac{B}{2}}{B^{n+1}}$$

and

$$(\overline{x+y}_n + 1)B^{-n} > \frac{x_{n+1} + y_{n-1} + \frac{B}{2}}{B^{n+1}}.$$

But $B \geq 4$, so $B/2 \geq 2$ and we have

$$(\overline{x+y}_n - 1)B^{-n} < (x_{n+1} - 1)B^{-(n+1)} + (y_{n+1} - 1)B^{-(n+1)}$$

and

$$(\overline{x+y}_n + 1)B^{-n} > (x_{n+1} + 1)B^{-(n+1)} + (y_{n+1} + 1)B^{-(n+1)}.$$

According to the bounds property of x and y for order $n + 1$ respectively by x_{n+1} and y_{n+1} , we have

$$(x_{n+1} - 1)B^{-(n+1)} < x < (x_{n+1} + 1)B^{-(n+1)}$$

and

$$(y_{n+1} - 1)B^{-(n+1)} < y < (y_{n+1} + 1)B^{-(n+1)},$$

and we deduce from this that

$$(\overline{x+y}_n - 1)B^{-n} < x + y < (\overline{x+y}_n + 1)B^{-n}.$$

◊

Opposite of a real number

Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, the opposite of this number $-x$ is represented by the sequence $(\overline{-x}_n)_{n \in \mathbb{Z}}$ such that:

$$\overline{-x}_n = -x_n.$$

Proof: The correction is obvious. ◊

Multiplication of two real numbers

Let x et y be two real numbers represented by the sequences $(x_n)_{n \in \mathbb{Z}}$ and $(y_n)_{n \in \mathbb{Z}}$ respectively, the product of these two numbers $x + y$ is represented by the sequence $(\overline{x \times y}_n)_{n \in \mathbb{Z}}$ such that:

$$\overline{x \times y}_n = \left\lfloor \frac{1 + x_{p_x} \times y_{p_y}}{B^{p_x + p_y - n}} \right\rfloor$$

with $p_x = \max(n - \text{msd}(y) + 3, \lfloor n/2 \rfloor + 1)$
and $p_y = \max(n - \text{msd}(x) + 3, \lfloor n/2 \rfloor + 1)$.

Proof: The proof of correction of this algorithm is more technical than those of addition buut with the same spirit. ◊

Inverse of a real number

Let x be a real number respectively represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, the inverse of this number $1/x$ is represented by the sequence $(\overline{1/x_n})_{n \in \mathbb{Z}}$ such that:

$$\begin{aligned} &\text{if } n \leq -\text{msd}(x) \\ &\quad \text{then } \overline{1/x_n} = 0 \\ &\quad \text{else if } x_k > 1, \\ &\quad \quad \text{then } \overline{1/x_n} = \left[\frac{B^{k+n}}{x_k} \right] \\ &\quad \quad \text{else } \overline{1/x_n} = \left[\frac{B^{k+n}}{x_k} \right] \\ &\quad \text{with } k = n + 2\text{msd}(x) + 1. \end{aligned}$$

Proof: Same remark as for the correction of the multiplication. \odot

4.3 Algorithms for algebraic or transcendental functions

General idea of these algorithms

We introduce here for each function f the function \underline{f} that associates to a rational number r and an integer n a finite B -adic approximation $\underline{f}(r, n)/B^n$ such that

$$\frac{\underline{f}(r, n)}{B^n} < f(r) < \frac{\underline{f}(r, n) + 1}{B^n}.$$

We deduce from this function \underline{f} and for each real number r , a representation of $f(r)$ with $\overline{f(r)_n}$ close to $\underline{f}(x_k, n)$. The work consists in determining such a k that ensures the correction of the algorithm and we try to choose this index as small as possible. As an example we will give a proof of the algorithm for computing the exponential function.

We present now such algorithms for algebraic and transcendental usual functions.

k -th root

Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$ and k be an integer greater than or equal to 2. The k -th root of this number $\sqrt[k]{x}$ is represented by the sequence $(\overline{\sqrt[k]{x_n}})_{n \in \mathbb{Z}}$ such that:

$$\begin{aligned} &\text{if } x_{kn} \geq 0 \\ &\quad \text{then } \left[\overline{\sqrt[k]{x_{kn}}} \right] \\ &\quad \text{else fails.} \end{aligned}$$

Remark

We choose here to always give a value to $\sqrt[k]{x}$ when it make sense, even if it means that it does not fail for some slightly negatives values of x . We can of course choose to fail for all negative values by replacing the condition $x_{kn} \geq 0$ by $x_{kn} \geq 1$, but in this case for some slightly positive values it will fail also. \diamond

Exponential function

Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, $\exp(x)$ is represented by the sequence $(\overline{\exp(x)_n})_{n \in \mathbb{Z}}$ such that:

$$\overline{\exp(x)_n} = \underline{\exp} \left(\frac{x_k - 1}{B^k} \right) + 1$$

with $k = \max(0, n + 1, n + \log_B(2) + \log_B(e) \times \max(2, \frac{x_m + 2}{B^m}))$ and $m = \text{msd}(x)$.

One can precompute the value $\log_B e$ by implementing first the log function on real numbers or one can substitute a simple rational upper bound in the preceding formula. This second choice is practically simpler.

Proof: We have

$$\frac{x_k - 1}{B^k} < x < \frac{x_k + 1}{B^k}$$

and $x \mapsto \exp(x)$ is an increasing function so

$$B^n \exp\left(\frac{x_k - 1}{B^k}\right) < B^n \exp(x) < B^n \exp\left(\frac{x_k + 1}{B^k}\right).$$

Let α, α', β and β' be defined by the following equalities:

$$\begin{aligned} \alpha &= \exp\left(\frac{x_k - 1}{B^k}\right) \\ \alpha' &= B^n \alpha \\ \beta &= \exp\left(\frac{x_k + 1}{B^k}\right) \\ \beta' &= B^n \beta. \end{aligned}$$

For all positive real number z , we have $\exp(z) < 1 + z \exp(z)$, so

$$\beta - \alpha = \alpha \left(\exp\left(\frac{2}{B^k}\right) - 1 \right) < \frac{2}{B^k} \beta.$$

Therefore we have $0 < \beta' - \alpha' < 2 B^{n-k} \beta$. Let us prove that $2 B^{n-k} \beta < 1$, that is to say that $\beta < B^{k-n}/2$.

If $k \geq \text{msd}(x)$, then, according to the property 3.1, we have

$$x_{\text{msd}(x)} = \left\lfloor \frac{x_k}{B^{k-\text{msd}(x)}} \right\rfloor$$

and consequently

$$\frac{x_k}{B^{k-\text{msd}(x)}} < x_{\text{msd}(x)} + 1,$$

thus

$$\frac{x_k}{B^k} < \frac{x_{\text{msd}(x)} + 1}{B^{\text{msd}(x)}},$$

and

$$\frac{x_k + 1}{B^k} < \frac{x_{\text{msd}(x)} + 1}{B^{\text{msd}(x)}} + \frac{1}{B^k} < \frac{x_{\text{msd}(x)} + 2}{B^{\text{msd}(x)}}$$

since $k \geq \text{msd}(x)$.

We deduce from this inequality that

$$\beta = \exp\left(\frac{x_k + 1}{B^k}\right) < \exp\left(\frac{x_{\text{msd}(x)} + 2}{B^{\text{msd}(x)}}\right) = B^{\log_B(e) \times \frac{x_{\text{msd}(x)} + 2}{B^{\text{msd}(x)}}} \leq B^{k-n-\log_B(2)} = \frac{B^{k-n}}{2}.$$

If $0 \leq k < \text{msd}(x)$, then

$$\beta \leq \exp\left(\frac{x_k + 1}{B^k}\right) < \exp(2/B^k) = B^{\log_B(e) \times 2/B^k} \leq B^{2 \log_B(e)} \leq B^{k-n-\log_B(2)} = \frac{B^{k-n}}{2}.$$

Therefore $0 < \beta' - \alpha' < 1$ and we conclude by application of the property 3.1. \odot

Logarithm to base B

Let x be a strictly positive real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, $\log_B(x)$ is represented by the sequence $(\overline{\log_B x}_n)_{n \in \mathbb{Z}}$ such that:

$$\overline{\log_B x}_n = \underline{\log_B}(x_k - 1)_n - kB^n \text{ with } k = \max(n, 0) + \text{msd}(x) + 1.$$

Arctangent

Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, $\arctan(x)$ is represented by the sequence $(\overline{\arctan(x)}_n)_{n \in \mathbb{Z}}$ such that:

$$\overline{\arctan(x)}_n = \underline{\arctan}\left(\frac{x_k - 1}{B^k}\right)_n.$$

We can deduce π for example by the following formula:

$$\frac{\pi}{4} = 12 \arctan\left(\frac{1}{18}\right) + 8 \arctan\left(\frac{1}{57}\right) - 5 \arctan\left(\frac{1}{239}\right).$$

Sinus

Let us notice that there is an additional difficulty for this algorithm since the trigonometric functions are not uniformly monotonic, that is to say that they are not decreasing or increasing functions on the entire interval $[0, 2\pi]$.

Let x be a real number represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, $\sin(x)$ is represented by the sequence $(\overline{\sin(x)}_n)_{n \in \mathbb{Z}}$ such that:

$$\begin{aligned} &\text{if } y_k \leq z_k - 2 \text{ ou } y_k \geq 3z_k + 4, \\ &\quad \text{then } \overline{\sin(x)}_n = (-1)^p \underline{\sin}\left(\frac{y_k - 1}{B^k}\right)_n \\ &\quad \text{else if } y_k \leq z_k + 1, \\ &\quad \quad \text{then } \overline{\sin(x)}_n = (-1)^p B^n \\ &\quad \quad \text{else if } y_k \leq 3z_k - 4, \\ &\quad \quad \quad \text{then } \overline{\sin(x)}_n = (-1)^p \underline{\sin}\left(\frac{y_k + 1}{B^k}\right)_n \\ &\quad \quad \quad \text{else if } y_k \leq 3z_k + 3, \\ &\quad \quad \quad \quad \text{then } \overline{\sin(x)}_n = (-1)^{p+1} B^n \\ &\text{with } p = \left(\frac{x}{\pi}\right)_0 - 1, y = x - p\pi, z = \frac{\pi}{2} \text{ and } k = \max(1, n + 2). \end{aligned}$$

Other elementary functions

We deduce the other usual elementary functions from the preceding algorithms using the following formulas:

$$\begin{aligned} \sinh(x) &= \frac{\exp(x) - \exp(-x)}{2}, \\ \cosh(x) &= \frac{\exp(x) + \exp(-x)}{2}, \\ \tanh(x) &= \frac{\sinh(x)}{\cosh(x)}, \\ x^y &= \exp\left(\frac{y \log_B(x)}{\log_B(\exp(1))}\right), \end{aligned}$$

$$\begin{aligned}
\log_x y &= \frac{\log_B y}{\log_B x}, \\
\operatorname{arcsinh}(x) &= \log(x + \sqrt{x^2 + 1}), \\
\operatorname{arccosh}(x) &= \log(x + \sqrt{x^2 - 1}), \\
\operatorname{arctanh}(x) &= \frac{1}{2} \log\left(\frac{1+x}{1-x}\right), \\
\operatorname{arcsin}(x) &= \arctan\left(\frac{x}{\sqrt{1-x^2}}\right), \\
\operatorname{arccos}(x) &= \arctan\left(\frac{\sqrt{1-x^2}}{x}\right), \\
\cos(x) &= \sin\left(\frac{\pi}{2} - x\right) \\
\tan(x) &= \frac{\sin(x)}{\cos(x)}.
\end{aligned}$$

5 In practice: a prototype in Caml

5.1 The choice of the Caml language

The use of this language was of course a natural choice insofar as we began the study on the subject of arithmetic for a modern and reliable programming language about this language. Furthermore the first step of this study leads us to implement a very efficient exact rational arithmetic for this language. It is obvious that (almost) infinite integers are absolutely necessary, but an exact rational arithmetic (see [13]) is also necessary to compute the transcendental functions on rational parameters underlying the transcendental functions on real arguments.

Moreover functions in this language are easy to use as arguments or results of functions and since real numbers (and more generally infinite objects) are naturally represented by functions, it is easier to deal with real numbers in this language.

5.2 Choices of implementation

We choose as Boehm to represent real numbers as finite B -adic numbers and furthermore these particular finite B -adic numbers, instead as general rational numbers. This choice leads us to a rougher granularity and a slightly lesser flexibility for our representation. For instance, if an accuracy under $1/B^n$ is required, this choice of implementation leads to a computation with an accuracy of $1/B^{n+1}$ and induce a greater running time than a computation to the real precision of $1/B^n - \varepsilon$ where ε is a rational number as small as possible.

Boehm's implementation used rational numbers at the beginning and it turned out that with the library of rational arithmetic used by Boehm, the computations with rational numbers were much slower than those performed with finite B -adic numbers. Through lack of details on the rational arithmetic that Boehm used, we don't know if this observation is valid for our implementation too. Moreover this choice deprive us of the natural incrementality of the representation and of a slightly simpler expression of our algorithms.

The implementation includes the storage of the most precise approximation already computed for each real number and we choose to work with the base $B = 4$. We will now justify our choice.

For efficiency, the representation contents for each real number x represented by the sequence $(x_n)_{n \in \mathbb{Z}}$, the most precise approximation already computed $x_{\operatorname{mpa}(x)}$ to the order $\operatorname{mpa}(x)$. Thus any less precise approximation for x needs only a simple shift operation on $x_{\operatorname{mpa}(x)}$ rather than an possibly complex computation

already performed before:

$$\text{if } n \leq \text{mpa}(x), \text{ then we take } x_n = \left\lfloor \frac{x_{\text{mpa}(x)}}{B^{\text{mpa}(x)-n}} \right\rfloor.$$

Let us notice that the value of x_n and of $\text{msd}(x)$ may vary of one unity according to the value of $\text{mpa}(x)$. However the rather strict definition that we give of this function ensures, independently of the value of $\text{mpa}(x)$, the fundamental property we wanted to ensure, i.e. $1 < |x_{\text{msd}(x)}| \leq B$ whatever the value of $\text{mpa}(x)$ is when $\text{msd}(x)$ is computed.

Concerning the choice of the base, some algorithms are valid only for a base greater than or equal to 4. It is preferable to choose 2 to a power greater than or equal to 4, since

$$\left\lfloor \frac{x_{\text{mpa}(x)}}{B^{\text{mpa}(x)-n}} \right\rfloor$$

can be computed by a simple shift of $x_{\text{mpa}(x)}$ of $\text{mpa}(x) - n$ digits to the right in this case, that is to say a basic operation of rational arithmetic and then for the underlying hardware arithmetic. It may seem worthwhile that a digit for the base B corresponds exactly to a computer word. However we choose, as Boehm did before, to work with $B = 4$ because the smaller the base is and the less we pay to compute an additional digit.

5.3 Realisations

Boehm implemented a similar arithmetic, so one can read his commentaries in [7, 6].

Moreover we have currently an almost complete prototype for this representation. The rational operations are already available. Concerning the transcendental functions, we have still to implement the arctangent and sinus functions.

As we see it at the beginning of the subsection 4.3, we need a set of elementary functions from \mathbb{Q} to \mathbb{R} , that we will not detail here, because even if on the one hand this software layer requires some time to implement, on the other hand the techniques we used are rather classical. These functions are defined from their Taylor expansion. Then we have to determine a condition to stop the computation, the number of terms for instance, for an accuracy given as a parameter and then evaluate this polynomial and possibly to apply some optimisations to speed up the computation.

The chosen representation has the advantage of using algorithms on integers that are well understood and very efficient.

Conclusion

We have a description of a representation of \mathcal{R} and proved algorithms for this representation for all elementary functions.

We have implemented an almost complete prototype of this description and its completion will give us a complete chain for reliable arithmetic in the Caml language.

In the future, we will try to use rational numbers rather than finite B -adics to analyze the influence of the efficiency of the underlying rational arithmetic and of the natural incrementality of the algorithms for rational numbers. As a second point we hope to be able to take a less restrictive definition of \underline{f} for any transcendental functions f corresponding to the bounds property rather than to a half interval. Finally we hope to improve this efficiency of this prototype by an efficient computation of the \underline{f} functions by using as those developed in [3], by a balancement of the abstract syntax tree during the compilation of expression such as $x_1 + \dots + x_n$ to compute each x_i with a well-balanced precision and finally by combination with floating point analysis method such as interval analysis [16, 17, 1, 10] or the CESTAC method [22, 9, 21] in the spirit of the lazy rational arithmetic by Michelucci [4].

References

- [1] O. ABERTH. *Precise numerical analysis*. Wm. C. Brown Publishers, 1988.
- [2] A. AVIZIENIS. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on electronic computers*, 10 (1961), pp. 389–400.
- [3] C. BATUT. *Aspects algorithmiques du système de calcul arithmétique en multiprécision PARI*. Thèse de doctorat, Université de Bordeaux I, February 1989.
- [4] M. BENOAMER, P. JAILLON, D. MICHELUCCI, AND J.-M. MOREAU. A Lazy Solution to Imprecision in Computational Geometry. In *Proceedings of the 5th Canadian Conference on Computational Geometry* (Waterloo, Canada, August 1993), pp. 73–78.
- [5] E. BISHOP AND D. BRIDGES. *Constructive Analysis*, vol. 279 of *Grundlehren der mathematischen Wissenschaften, A series of Comprehensive Studies in Mathematics*. Springer Verlag, 1985.
- [6] H. J. BOEHM. Constructive Real Interpretation of numerical Programs. In *Proceedings of the 1987 ACM conference on Interpreters and Interpretives Techniques* (1987), ACM.
- [7] H. J. BOEHM, R. CARTWRIGHT, M. J. O'DONNELL, AND M. RIGGLE. Exact real arithmetic: a case study in higher order programming. In *Proceedings of the 1986 ACM conference on Lisp and Functional Programming* (1986), ACM.
- [8] B. CHAR, K. GEDDES, G. GONNET, M. MONAGAN, AND S. WATT. *MAPLE: Reference Manual*. Symbolic Computation Group, Department of Computer Science, University of Waterloo, 1988. 5th Edition.
- [9] J.-M. CHESNEAUX. Study of the computing accuracy by using probabilistic approach. In *Contribution to Computer arithmetic and Self-Validating Numerical Methods* (Basel, Switzerland, October 1990), C. Ullrich, Ed., Academic Press, pp. 19–30.
- [10] M. DAUMAS, C. MAZENC, AND J.-M. MULLER. User transparent interval arithmetic. Rapport de recherche 94-02, École Normale Supérieure de Lyon, January 1994.
- [11] H. X. LIN AND H. J. SIPS. On-Line Cordic Algorithms. *IEEE Transactions on Computers* 39, 8 (August 1990), pp. 1038–1052.
- [12] P. MARTIN-LÖF. *Notes on Constructive Mathematics*. Almqvist et Wiksell, Stockholm, 1970.
- [13] V. MÉNISSIER-MORAIN. The CAML Numbers Reference Manual. Tech. Rep. 141, INRIA, July 1992.
- [14] V. MÉNISSIER-MORAIN. *Arithmétique exacte, conception, algorithmique et performances d'une implémentation informatique en précision arbitraire*. Thèse, Université Paris 7, December 1994.
- [15] J.-M. MULLER. *Arithmétique des ordinateurs*. Etudes et recherches en informatique. Masson, 1989.
- [16] K. NICKEL, Ed. *Interval mathematics* (Karlsruhe, Germany, May 1975). LNCS 29.
- [17] K. NICKEL, Ed. *Proceedings of the international symposium on interval mathematics* (September 1985), Springer-Verlag. LNCS 212.
- [18] H. G. RICE. Recursive real numbers. *Proceedings of the American Mathematical Society* 5, 5 (1954).
- [19] G. STOLZENBERG. A new Course in Analysis. Note of a course given between 1972 and 1987 at Northeastern University, Boston MA 02115 USA, 1972-1987.

- [20] A. M. TURING. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* 2, 42 (1936).
- [21] J. VIGNES. A stochastic arithmetic for reliable scientific computation. *Mathematics and computers in simulation* 35 (1993), pp. 233–261.
- [22] J. VIGNES AND M. LA PORTE. Error analysis in computing. In *Information Processing 74* (August 1974), North-Holland.
- [23] J. VUILLEMIN. Exact real computer arithmetic with continued fractions. Research report 760, INRIA, 1987.
- [24] J. VUILLEMIN. Exact real computer arithmetic with continued fractions. In *Proceedings ACM conference on Lisp and Functional Programming* (1988), ACM. Extended version as INRIA research report 760, 1987.
- [25] J. VUILLEMIN. Exact real computer arithmetic with continued fractions. *IEEE Transactions on computers* 39, 8 (August 1990), pp. 1087–1105.
- [26] E. WIEDMER. *Exaktes Rechnen mit reellen Zahlen und anderen unendlichen Objekten*. PhD thesis, ETH, Zurich, 1977. Diss. ETH 5975.
- [27] E. WIEDMER. Calculs avec des fractions décimales et d’autres objets infinis. Compte-rendu d’un exposé à l’Institut de Programmation de l’Université Paris VII, January 1979.
- [28] E. WIEDMER. Computing with infinite objects. *Theoretical Computer Science* 10 (1980).