Efficient on-line computation of real functions using exact floating point

Peter John Potts
Department of Computing, Imperial College,
180 Queen's Gate, London SW7 2BZ, UK

17 October 1997

Abstract

In this article, we develop the work of Vuillemin and Nielsen and Kornerup, and show that incrementality and efficiency can be simultaneously achieved in exact real arithmetic. We present a formal account of incremental digit representations born out of domain theory, which includes the redundant binary representation and continued fraction representation. The generalisation of both these representations leads to the notion of a general normal product constructed using Möbius transformations. We then examine a specialisation of general normal products called exact floating point with interesting mathematical properties. Real functions are captured by the composition of 2-dimensional Möbius transformations, leading to the notion of expression trees. Various reduction rules and a lazy form of information flow analysis is used to allow expression trees to be converted efficiently into the exact floating point representation. Algorithms for the basic arithmetic operations and the transcendental functions are presented using "redundant if" statements for range reduction and expression trees derived from the theory of continued fractions.

Index Terms: Arbitrary precision, redundancy, real numbers, real functions, Möbius transformations, continued fractions, information flow analysis.

1 Introduction

Real numbers are usually represented by finite strings of digits belonging to some digit set. The real number representation specifies a function that maps strings to real numbers. However, finite strings of digits can only represent a limited subset of the real numbers exactly. This means that most real numbers are represented by nearby real numbers or enclosing real intervals with distinct end-points giving rise to the notion of round-off errors. This is generally accepted for a wide range of applications. However, it is well-known that the accumulation of round-off errors due to a large number of calculations can produce grossly inaccurate or even incorrect results.

Interval analysis [25] has been used to partially circumvent this problem by maintaining a pair of bounding numbers that is guaranteed to contain the real number or interval in question. However, this interval can get unjustifiably large and thereby convey very little information.

Alternatively, by allowing infinite strings of digits all the real numbers can be represented exactly. There have been a number of theoretical and practical attempts to find a viable framework for exact real arithmetic. Broadly speaking they fall into three categories:

1. Infinite sequences of linear maps proposed by Avizienis [2] and appeared in the work of Watanuki and Ercegovac [37], Boehm and Cartwright [4], Di Gianantonio [13], Escardó [10], Nielsen and Kornerup [28] and Ménissier-Morain [24].

2. Continued fraction expansions proposed by Gosper [14], developed by Peyton-Jones [29] and Vuillemin [35], implemented by Lester [22] and advanced more recently by Kornerup and Matula [17, 18, 19, 20].

3. Infinite composition of Möbius transformations generalizes the other two frameworks as demonstrated by Vuillemin [35]. Nielsen and Kornerup [28] showed that this framework can be used to represent quasi-normalized floating point [37].

In this article, we develop the work of Vuillemin [35] and Nielsen and Kornerup [28] culminating in the presentation of efficient algorithms for exact real arithmetic involving the basic arithmetic operations and the transcendental operations. We start by noting that every real number can be represented by a sequence of nested closed intervals whose lengths converge to zero. The intersection of these intervals is a singleton set whose element is the real number being represented. Using a computability argument, we justify an extension of the real numbers with infinity and bottom.

We then present the incremental digit representation, which brings together the digit serial representation by Nielson and Kornerup [28] and domain theory. We show that the standard real number representations such as decimal, continued fractions and redundant binary fit into this framework. We then review the relevant properties of Möbius transformations (also known as homographies and linear fractional transformations) before using them to construct the incremental digit representations of general normal product and exact floating point by Potts and Edalat [31, 32, 9]. We highlight the crucial features of these representations in an attempt to answer the challenge by Vuillemin [35] to find a rational choice for a normal form.

We then explore the relevant properties of 2-dimensional Möbius transformations, first used by Gosper [14] and Vuillemin [35] to perform the basic arithmetic operations on redundant continued fractions. Motivated by the goal to represent real functions elegantly, we use these 2-dimensional Möbius transformations to construct the notion of an expression tree and provide the link to domain theory. We present a straightforward algorithm for converting an expression tree into the exact floating point representation, involving the essential notion of the "redundant if" statement.

We first show how expression trees are constructed for the basic arithmetic operations and then we outline a general two part procedure for converting any function with a power series representation into an expression tree via an intermediate continued fraction representation. Using this procedure, we derive new algorithms for π, e and the transcendental functions [30].

Finally, we quantize information and use this idea to analyze the flow of information around an expression tree in order to improve temporal efficiency. We also examine the storage requirements of an expression tree in order to improve spatial efficiency. This section culminates in the presentation of an efficient algorithm for converting an expression tree into the exact floating point representation.

2 Representing Real Numbers

A real number x can be represented by any sequence of nested closed intervals

$$[p_0, q_0] \supseteq [p_1, q_1] \supseteq [p_2, q_2] \supseteq \cdots$$

such that $\lim_{n\to\infty} |p_n - q_n| = 0$ and $x \in [p_n, q_n]$ for all $n \geq 0$. All real numbers have reciprocals except 0. However, equality is not decidable on the real numbers and therefore we must include 0^{-1}, namely infinity ∞. This is known as the one-point compactification of the real numbers $\mathbb{R}^\infty$ (or the *extended real numbers)*. It is usually represented by the stereographic projection $\sigma : \mathbb{R}^\infty \longrightarrow \mathbb{C}$

$$\sigma(x) = \frac{2x + (x^2 - 1)i}{x^2 + 1}$$

of the extended real numbers onto the unit complex circle as illustrated in figure 1.

The usual metric $d_{\mathrm{C}} : \mathbb{R}^\infty \times \mathbb{R}^\infty \longrightarrow [0, 2]$ on the extended real numbers is the chordal distance given by

$$d_{\mathrm{C}}(x, y) = |\sigma(x) - \sigma(y)| = \frac{2|x - y|}{\sqrt{x^2 + 1}\sqrt{y^2 + 1}}. \tag{1}$$

This means that an extended real number x can be represented by any sequence of nested closed intervals

$$[p_0, q_0] \supseteq [p_1, q_1] \supseteq [p_2, q_2] \supseteq \cdots$$

such that $\lim_{n\to\infty} d_{\mathrm{C}}(p_n, q_n) = 0$ and $x \in [p_n, q_n]$ for all $n \geq 0$ where $[p_n, q_n]$ denotes the closed interval represented by the arc from $\sigma(p_n)$ to $\sigma(q_n)$ anti-clockwise in figure 1. However, including ∞ leads to further difficulties. What are we to make of $0 \times \infty$? The only sensible answer is to introduce the concept of an "undefined number" or "not a number" denoted by NaN in the floating point community. Domain theoretically, this object is of course bottom.

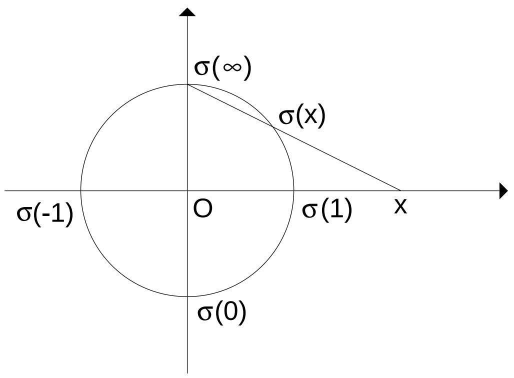


Figure 1: Stereographic projection σ

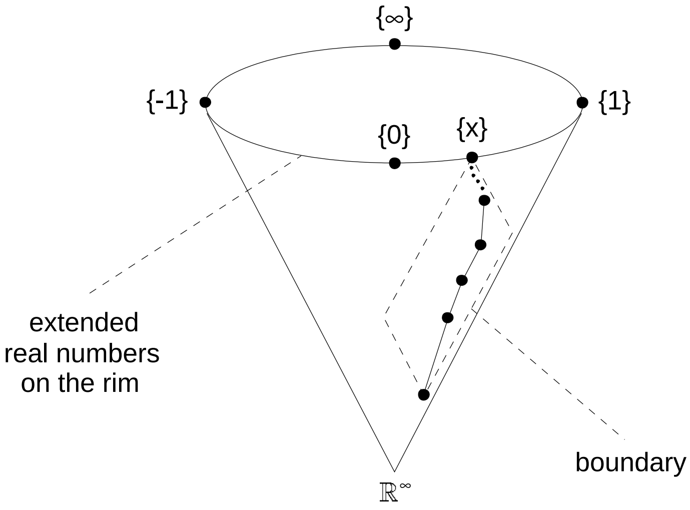


Figure 2: A chain representing the extended real number x on the continuous domain $(\mathrm{I}\mathbb{R}^\infty, \supseteq)$

3 Domains Theory and the Incremental Digit Representation

A domain is a partially ordered set equipped with a notion of completion and approximation for modelling computation in various areas of computer science and mathematics [1, 8, 13]. A sequence of nested closed intervals representing an extended real number can be modelled by a chain in the continuous domain $(\mathrm{I}\mathbb{R}^\infty, \supseteq)$ [33], which is the set of closed intervals in $\mathbb{R}^\infty$ (including $\mathbb{R}^\infty$ itself) ordered by reverse inclusion as illustrated in figure 2. The extended real numbers are represented by singleton sets on the rim of the cone, while the other intervals are represented by points on the surface of the cone. The bottom element is $\mathbb{R}^\infty$.

The incremental digit representation defined below is similar to the digit serial number representation defined by Nielsen and Kornerup [28].

Definition 1 *The tuple $(B, \Delta, \psi, \Omega, \phi)$ is called an **unsigned incremental digit representation** if*

- *the **base interval** B is a member of $\mathbb{IR}^{\infty}$ [9],*
- *the **digit set** Δ is a countable set of symbols,*
- *the **digit map** ψ is a function $\Delta \to B \to B$ and*
- *the **terminator set** Ω is a countable set of symbols and*
- *the **terminator map** ϕ is a function $\Omega \to B$.*

An infinite sequence of digits $d_0 d_1 d_2 \cdots$ represents a real number x in B by inducing an infinite sequence of nested closed intervals

$$[p_n, q_n] = \psi(d_0)(\psi(d_1)(\psi(d_2)(\ldots\psi(d_n)(B)\ldots)))$$

such that the induced sequence $[p_0, q_0] \supseteq [p_1, q_1] \supseteq [p_2, q_2] \supseteq \cdots$ is a representative of x. Similarly, a finite sequence of digits $d_0 d_1 d_2 \cdots d_n$ terminated by $\tau \in \Omega$ represents a real number x in B by inducing a finite sequence of nested closed intervals terminating with the singleton set $\{x\}$.

$$\psi(d_0)(\psi(d_1)(\psi(d_2)(\ldots\psi(d_n)(\phi(\tau))\ldots))) = x$$

Definition 2 *The tuple $(B, \Sigma, \varphi, \Delta, \psi, \Omega, \phi)$ is called a **signed incremental digit representation** if*

- *$(B, \Delta, \psi, \Omega, \phi)$ is an unsigned incremental digit representation,*
- *the **sign set** Σ is a countable set of symbols and*
- *the **sign map** φ is a function $\Sigma \to B \to \mathbb{R}^{\infty}$.*

A real number x in the unsigned incremental digit representation prefixed by a sign $\sigma \in \Sigma$ represents the real number $\varphi(\sigma)(x)$. Here are some examples:

- The *decimal representation* on $\mathbb{R}$

$$\left([0,1], \mathbb{Z}, \sigma \mapsto x \mapsto \sigma + x, \{0, \ldots, 9\}, d \mapsto x \mapsto \tfrac{d+x}{10}, \{1, \ldots, 9\}, \tau \mapsto \tfrac{\tau}{10}\right)$$

- The *simple continued fraction representation* on $\mathbb{R}$

$$\left([1,\infty], \mathbb{Z}, \sigma \mapsto x \mapsto \sigma + \tfrac{1}{x}, \mathbb{N} - \{0\}, d \mapsto x \mapsto d + \tfrac{1}{x}, \mathbb{N} - \{0, 1\}, \tau \mapsto \tau\right)$$

- The *redundant binary representation* on $\mathbb{R}$

$$\left([-1,1], \mathbb{Z}, \sigma \mapsto x \mapsto \sigma + x, \{-1, 0, 1\}, d \mapsto x \mapsto \tfrac{d+x}{2}, \{-1, 0, 1\}, \tau \mapsto \tfrac{\tau}{2}\right)$$

4 Möbius Transformations

A *Möbius transformation* $\Psi(M)$ is a linear fractional transformation on the extended real numbers

$$x \mapsto \frac{ax+c}{bx+d} \tag{2}$$

parameterized by the four integers a, b, c and d arranged conveniently in the 2×2 matrix $M = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$.

The symbol $\bullet$ will be used to indicate the usual dot product between matrices and vectors. However, the same symbol with a numerical subscript $\bullet_n$ will be used to indicate a more general dot product between tensors, yet to be defined. The composition of Möbius transformations $\Psi(M)$ and $\Psi(N)$ is equivalent to the product of matrices M and N.

$$\Psi(M)(\Psi(N)(x)) = \Psi(M \bullet N)(x)$$

Therefore, Ψ is a morphism from the monoid of matrices with integer coefficients $\mathbb{M}$ to Möbius transformations and the kernel of Ψ is the monoid of non-zero integers $\mathbb{Z}^*$. Consequently, the monoid of Möbius transformations is isomorphic to the quotient monoid $\mathbb{M}/\mathbb{Z}^*$. In the light of this, we shall consider matrices to be equivalent up to scaling. Therefore, we shall define the inverse of a non-singular matrix by

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix}^{-1} = \begin{pmatrix} d & -c \\ -b & a \end{pmatrix}. \tag{3}$$

The set of vectors with integer coefficients $\mathbb{V}$ with product $\times$ defined by

$$\begin{pmatrix} a \\ b \end{pmatrix} \times \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ bd \end{pmatrix}$$

is a monoid. Therefore, the kernel of the morphism Φ from the monoid of vectors $\mathbb{V}$ to the extended real numbers given by

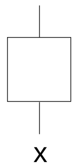
$$\Phi\begin{pmatrix} a \\ b \end{pmatrix} = \frac{a}{b}$$

is also the monoid of non-zero integers $\mathbb{Z}^*$. In the light of this, we shall also consider vectors to be equivalent up to scaling. The function apply given by

$$\begin{aligned} \mathsf{apply} : \Phi(\mathbb{V}) \times \Psi(\mathbb{M}) &\longrightarrow \Phi(\mathbb{V}) \\ (x, f) &\longmapsto f(x) \end{aligned}$$

is an action of the monoid of Möbius transformations on the extended real numbers because

$$\Psi(M)(\Phi(V)) = \Phi(M \bullet V).$$

For convenience, we will drop Ψ and Φ whenever it is clear to do so. Also, Möbius transformations and extended real numbers will be pictured abstractly by 

and

$\square$ respectively.

5 General Normal Products

Two real numbers x and y have the **same sign** if $x \times y \geq 0$. Let $\mathbb{V}^+$ and $\mathbb{M}^+$ denote the set of vectors and matrices with same sign integer coefficients. Let the **unsigned general normal product representation** on $[0, \infty]$ be the unsigned incremental digit representation $([0, \infty], \mathbb{M}^+, \Psi, \mathbb{V}^+, \Phi)$ and let the **signed general normal product representation** on $\mathbb{R}^\infty$ be the signed incremental digit representation $([0, \infty], \mathbb{M}, \Psi, \mathbb{M}^+, \Psi, \mathbb{V}^+, \Phi)$ [30]. The *special base interval* $[0, \infty]$ [9] ensures that evaluating the *information in a matrix* with integer coefficients defined by

$$\mathsf{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}([0, \infty]) \tag{4}$$

is computationally trivial.

$$\mathsf{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{cases} \left[\frac{a}{b}, \frac{c}{d}\right] & \text{if } \det\begin{pmatrix} a & c \\ b & d \end{pmatrix} < 0 \\ \left[\frac{c}{d}, \frac{a}{b}\right] & \text{if } \det\begin{pmatrix} a & c \\ b & d \end{pmatrix} > 0 \end{cases} \tag{5}$$

For the neglected case, $\det\begin{pmatrix} a & c \\ b & d \end{pmatrix} = 0$, it can be shown that $\exists\, p, q, r, s \in \mathbb{Z}$ such that $\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} rp & sp \\ rq & sq \end{pmatrix}$. In which case

$$\mathsf{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{cases} \mathbb{R}^\infty & \text{if } rs \leq 0 \\ \left\{\frac{p}{q}\right\} & \text{if } rs > 0 \end{cases}. \tag{6}$$

For completeness, we define the information in a vector by

$$\mathsf{info}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{cases} \mathbb{R}^\infty & \text{if } a = 0 \text{ and } b = 0 \\ \left\{\frac{a}{b}\right\} & \text{otherwise} \end{cases}. \tag{7}$$

Clearly, by comparing equations (6) and (7), a singular matrix can be reduced to a vector.

The use of same sign coefficients is justified by the following proposition.

Proposition 1 *The information in a non-singular matrix is a subset of the special base interval if and only if the non-singular matrix has same sign coefficients.*

6 Exact Floating Point

General normal products can be used to elegantly represent algorithms derived from the theory of continued fractions [30]. However, as a general tool for representing extend real numbers it is extremely difficult to control the size of integers and the flow of information. Of course, the redundant binary representation does not suffer from these drawbacks. Consider the following commuting diagram [32, 9]:

$$\begin{tikzcd}
{[-1,1]} \arrow[r, "\begin{pmatrix} 1 & d \\ 0 & 2 \end{pmatrix}"] & {[-1,1]} \\
{[0,\infty]} \arrow[u, "\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}"] \arrow[r, "D(d)"'] & {[0,\infty]} \arrow[u, "\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}"']
\end{tikzcd}$$

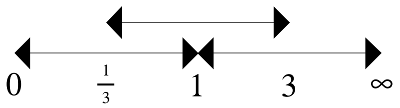
In other words

$$D(d) = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}^{-1} \bullet \begin{pmatrix} 1 & d \\ 0 & 2 \end{pmatrix} \bullet \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 3+d & 1+d \\ 1-d & 3-d \end{pmatrix}.$$

This gives the three *digit matrices* [19, 28, 32, 9]:

$$D_0 = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

$$D_{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}$$



$$D_1 = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$$

It follows from the basic properties of the redundant binary representation that

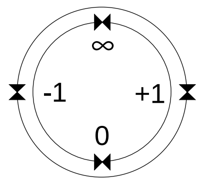
$$D_{d_1} D_{d_2} \dots D_{d_n} = \mathfrak{D}^n_c \stackrel{\text{def}}{=} \begin{pmatrix} 2^n + c + 1 & 2^n + c - 1 \\ 2^n - c - 1 & 2^n - c + 1 \end{pmatrix} \text{ where } c = \sum_{i=1}^{n} d_i 2^{n-i}. \tag{8}$$

The implication of this identity is that any sequence of n digit matrices can be compressed into $n+1$ bits of memory. For example, the two's complement representation of c. Note that $c \in \{1 - 2^n, \dots, 2^n - 1\}$ and $\mathfrak{D}^n_c D_d = \mathfrak{D}^{n+1}_{2c+d}$. Beware however that the original sequence of digits cannot usually be recovered except for $n = 1$. Let the **unsigned exact floating point representation** on $[0, \infty]$ be the unsigned incremental digit representation $\left([0,\infty], D_{\{-1,0,1\}}, \Psi, \mathbb{V}^+, \Phi\right)$. This representation is called floating point because a sequence of digits can be divided into two parts in a fashion reminiscent of an exponent and a mantissa. An infinite sequence of D_1 digits represents ∞. Any other infinite sequence of digits can be identified as a finite sequence of n D_1 digits (exponent) followed by an infinite sequence of digits starting with either a D_{-1} or D_0 digit (mantissa). In particular, the mantissa part represents a real number in $\mathsf{info}(D_{-1}) \cup \mathsf{info}\,(D_0) = [0, 3]$, while the exponent part maps this number into the interval $D_1^n\,([0, 3]) = [2^n - 1, 2^{n+2} - 1]$.

There are no "natural" constants in physics, goes the maxim, except zero, one and infinity. Therefore, the most natural choice for redundantly dividing up the one-point compactification of the real numbers is the four intervals $[0,\infty]$, $[1,-1]$, $[\infty,0]$ and $[-1,1]$. These intervals can be represented by the four *sign matrices*

$$S_\infty = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

$$S_- = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$



$$S_+ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$S_0 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

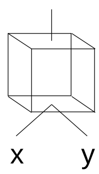
which conveniently form a cyclic group of order 4 [32, 9, 27]. Let the **signed exact floating point representation** on $\mathbb{R}^\infty$ be the signed incremental digit representation $\left([0,\infty], S_{\{+,\infty,-,0\}}, \Psi, D_{\{-1,0,1\}}, \Psi, \mathbb{V}^+, \Phi\right)$.

7 2-dimensional Möbius Transformations

In order to compute the basic arithmetic operations on redundant continued fractions, Gosper [14] and Vuillemin [35] used *2-dimensional Möbius transformations.* A *2-dimensional Möbius transformation* $\Upsilon(T)$ is a linear fractional transformation on the extended real numbers

$$(x,y) \mapsto \frac{axy + cx + ey + g}{bxy + dx + fy + h} \tag{9}$$

parameterized by the eight integers a, b, c, d, e, f, g and h arranged conveniently in the $2 \times 2 \times 2$ tensor $T = \begin{pmatrix} a & & e & \\ & c & & g \\ b & & f & \\ & d & & h \end{pmatrix}$, denoted more compactly by $\begin{pmatrix} a & c & e & g \\ b & d & f & h \end{pmatrix}$ and pictured by



.

Let $\mathbb{T}$ denote the set of tensors with integer coefficients and let $\mathbb{T}^+$ denote the set of tensors with same sign integer coefficients. Let us refer collectively to the members of $\mathbb{L} = \mathbb{V} \cup \mathbb{M} \cup \mathbb{T}$ as *linear fractional transformations* and let $\mathbb{L}^+ = \mathbb{V}^+ \cup \mathbb{M}^+ \cup \mathbb{T}^+$. For any two vectors $P = \begin{pmatrix} a \\ b \end{pmatrix}$ and $Q = \begin{pmatrix} c \\ d \end{pmatrix}$, let $\begin{pmatrix} P & Q \end{pmatrix}$ and $\begin{pmatrix} P & \\ & Q \end{pmatrix}$ denote $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$. For any two matrices $R = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ and $S = \begin{pmatrix} e & g \\ f & h \end{pmatrix}$, let $\begin{pmatrix} R & S \end{pmatrix}$ denote

$\begin{pmatrix} a & & e & \\ & c & & g \\ b & & f & \\ & d & & h \end{pmatrix}$ and $\begin{pmatrix} R & \\ & S \end{pmatrix}$ denote $\begin{pmatrix} a & & c & \\ & e & & g \\ b & & d & \\ & f & & h \end{pmatrix}$. Let us define the *transpose* of a tensor by

$$\begin{pmatrix} R & S \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} R & \\ & S \end{pmatrix}. \tag{10}$$

Let us define the *left product* $\bullet_1$ of a tensor with a linear fractional transformation $L \in \mathbb{V} \cup \mathbb{M}$ by

$$\begin{pmatrix} R & \\ & S \end{pmatrix} \bullet_1 L = \begin{pmatrix} R \bullet L & \\ & S \bullet L \end{pmatrix} \tag{11}$$

and the *right product* $\bullet_2$ of a tensor with a linear fractional transformation $L \in \mathbb{V} \cup \mathbb{M}$ by

$$\begin{pmatrix} R & S \end{pmatrix} \bullet_2 L = \begin{pmatrix} R \bullet L & S \bullet L \end{pmatrix}. \tag{12}$$

It can be shown that

$$\begin{aligned} \Upsilon(T)(M(x), y) &= \Upsilon(T \bullet_1 M)(x, y) \\ \Upsilon(T)(x, M(y)) &= \Upsilon(T \bullet_2 M)(x, y) \\ \Upsilon(T)(V, y) &= \Psi(T \bullet_1 V)(y) \\ \Upsilon(T)(x, V) &= \Psi(T \bullet_2 V)(x) \end{aligned}$$

and

$$\begin{aligned} M(\Upsilon(T)(x, y)) &= \Upsilon(M \bullet T)(x, y) \\ \text{where } M \bullet \begin{pmatrix} R & S \end{pmatrix} &= \begin{pmatrix} M \bullet R & M \bullet S \end{pmatrix} \end{aligned}$$

for $V \in \mathbb{V}$ and $M \in \mathbb{M}$.

Define the *information in a tensor T* by

$$\mathsf{info}(T) = \Upsilon(T)([0, \infty], [0, \infty]). \tag{13}$$

A 2-dimensional Möbius transformation $\Upsilon(T)$ is monotonic in each argument separately, with respect to the topology of the one-point compactification of the real numbers. Therefore

$$\mathsf{info}\begin{pmatrix} a & & e & \\ & c & & g \\ b & & f & \\ & d & & h \end{pmatrix} = \mathsf{info}\begin{pmatrix} a & c \\ b & d \end{pmatrix} \cup \mathsf{info}\begin{pmatrix} c & g \\ d & h \end{pmatrix} \cup \mathsf{info}\begin{pmatrix} g & e \\ h & f \end{pmatrix} \cup \mathsf{info}\begin{pmatrix} e & a \\ f & b \end{pmatrix}.$$

For convenience, we will drop Υ whenever it is clear to do so.

8 Expression Trees

Definition 3 *An* ***unsigned expression tree*** *is a binary tree. Each node may be either*

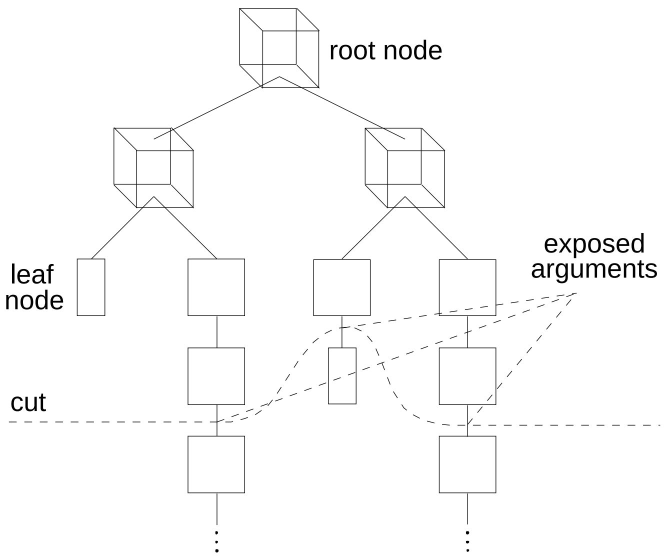


Figure 3: A typical expression tree

- *a tensor $T \in \mathbb{T}^+$ with 2 children or*
- *a matrix $M \in \mathbb{M}^+$ with 1 child or*
- *a vector $V \in \mathbb{V}^+$ with no children.*

Definition 4 *A **signed expression tree** is a tree consisting of either*

- *a tensor $T \in \mathbb{T}$ connected to two unsigned expressions trees or*
- *a matrix $M \in \mathbb{M}$ connected to one unsigned expressions tree or*
- *a vector $V \in \mathbb{V}$ with no children.*

Let $\mathbb{E}$ and $\mathbb{E}^+$ denote the set of signed and unsigned expression trees respectively. An expression tree (e.g. as shown in figure 3) induces a directed set in the continuous domain $(\mathrm{I}\mathbb{R}^\infty, \supseteq)$ (e.g. as illustrated in figure 4). Each element of the directed set corresponds to a particular *cut* in the expression tree. This element is then the interval derived by plugging each *exposed argument* with the special base interval $[0, \infty]$.

An expression tree represents an extended real number x if the least upper bound of the induced directed set is the singleton set $\{x\}$. The expression tree $E \in \mathbb{E}$ consisting of a root node $L \in \mathbb{L}$ connected to expression trees $E_{\{1,\ldots,N\}} \in \mathbb{E}^+$ with $N \in \{0, 1, 2\}$ will be denoted by $L\{E_1, \ldots, E_N\}$. Note that the general normal product representation and exact floating point representation are just special expression trees.

8.1 Matrix Application

The application of a matrix $M \in \mathbb{M}$ to a signed expression tree $L\{E_1, \ldots, E_N\} \in \mathbb{E}$ can be reduced to the signed expression tree with the root node $M \bullet L$ connected to the same

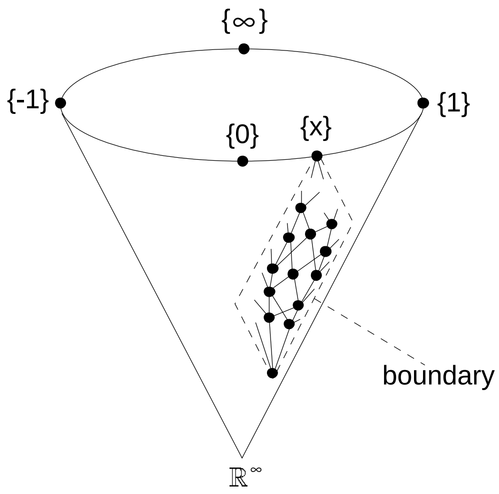


Figure 4: A directed set representing the extended real number x on the continuous domain $(\mathrm{I}\mathbb{R}^\infty, \supseteq)$ induced by an expression tree

unsigned expression trees $E_{\{1,\ldots,N\}}$.

$$M\left[L\left\{E_1, \ldots, E_N\right\}\right] = (M \bullet L)\left\{E_1, \ldots, E_N\right\}$$

8.2 Reciprocal and Negation

Reciprocal and negation are achieved by matrix application. For example, the signed expression tree corresponding to the reciprocal of the signed exact floating point number $S_\sigma\left\{D_{d_1}\left\{D_{d_2}\left\{\cdots\right\}\right\}\right\}$ is

$$\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \bullet S_\sigma\right)\left\{D_{d_1}\left\{D_{d_2}\left\{\cdots\right\}\right\}\right\}.$$

In fact, a purely symbolic algorithm is possible here for converting this expression tree into the signed exact floating point representation.

8.3 Tensor Application

The application of a tensor $T \in \mathbb{T}$ to two signed expression trees $K\left\{E_1, \ldots, E_M\right\}$, $L\left\{F_1, \ldots, F_N\right\} \in \mathbb{E}$ with $K, L \in \mathbb{V} \cup \mathbb{M}$ and $M, N \in \{0, 1\}$ can be reduced to the signed expression tree with the root node $T \bullet_1 K \bullet_2 L$ connected to the same unsigned expression trees $E_{\{1,\ldots,M\}}$ and $F_{\{1,\ldots,N\}}$.

$$T\left[K\left\{E_1, \ldots, E_M\right\}, L\left\{F_1, \ldots, F_N\right\}\right] = (T \bullet_1 K \bullet_2 L)\left\{E_1, \ldots, E_M, F_1, \ldots, F_N\right\}$$

Note that a signed expression tree with root node $T \in \mathbb{T}$ can be converted into a sign matrix $S_{\{+,\infty,-,0\}}$ connected to an unsigned expression tree using the reduction rules in the section 9.

8.4 The Basic Arithmetic Operations

The basic arithmetic operations $+$, $-$,$\times$ and $\div$ are achieved by tensor application as pointed out by Gosper [14]. For example, the signed expression tree corresponding to the subtraction of the two signed exact floating point numbers $S_\sigma \{D_{d_1} \{D_{d_2} \{\cdots\}\}\}$ and $S_\rho \{D_{e_1} \{D_{e_2} \{\cdots\}\}\}$ is

$$\left(\begin{pmatrix} 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet_1 S_\sigma \bullet_2 S_\rho\right) \{D_{d_1} \{D_{d_2} \{\cdots\}\}, D_{e_1} \{D_{e_2} \{\cdots\}\}\}.$$

9 Reduction Rules

In this section, We present the core reduction rules for converting an expression tree into an exact floating point number. The ethos behind the reduction rules is:

Emission Extract information from the root node in order to construct an exact floating point number *if you can.*

Absorption Assimilate information from the depths of the expression tree into the root node *if you have to.*

In section 12, the reduction rules are enhanced for efficiency reasons using information flow analysis.

9.1 Emission

Generally speaking a matrix E can be *emitted* from the root node $L \in \mathbb{L}$ provided

$$\mathsf{info}(E) \supseteq \mathsf{info}(L) \Leftrightarrow E^{-1} \bullet L \in \mathbb{L}^+. \tag{14}$$

For a signed expression tree, only a sign matrix $S_{\{+,\infty,-,0\}}$ may be emitted leaving an unsigned expression tree

$$E \rightarrow S_\sigma \left\{S_\sigma^{-1}[E]\right\}$$

provided $S_\sigma^{-1}[E] \in \mathbb{E}^+$. For an unsigned expression tree, only a digit matrix $D_{\{-1,0,1\}}$ may be emitted leaving another unsigned expression tree

$$E \rightarrow D_d \left\{D_d^{-1}[E]\right\}$$

provided $D_d^{-1}[E] \in \mathbb{E}^+$. The digit matrix D_0 should be avoided, if given a choice, because it involves slightly more computation.

9.2 Absorption

It has been shown earlier that

- matrices can always be assimilated with tensors, matrices and vectors using dot product and

- tensors can always be assimilated with matrices and vectors using left and right products.

However, tensors cannot be assimilated with tensors without introducing rank 4 tensors. The next best thing is an *information exchange*. Only the digit matrices $D_{\{-1,0,1\}}$ should be exchanged. This policy ensures a linear integer bit size growth for the coefficients in the two tensors.

9.3 Strategy

Consider an arbitrary tensor $T = \begin{pmatrix} P & Q \end{pmatrix} = \begin{pmatrix} R \\ & S \end{pmatrix} \in \mathbb{T}$ in which no more emissions are possible. We need to decide whether to absorb from the left or from the right. A left absorption increases the information in R and S, while a right absorption increases the information in P and Q. So, always left absorbing whenever the information in R and S are overlapping is a workable strategy.

$$\mathsf{strategy}\,(T) = \mathsf{if\ info}\,(R) \cap \mathsf{info}\,(S)\ \mathsf{is\ not\ empty\ then}\ 1\ \mathsf{else}\ 2 \tag{15}$$

Others strategies have been investigated, but this one has proved the most efficient in practice.

9.4 Redundancy

It is well known that the basic arithmetic operations are computable only for *redundant* representations of the real numbers [26, 23, 7, 34]. In effect, non-determinism is being traded for computability. It is also well known that basic predicates, such as equality and comparison, are not computable. Consequently, the standard "if" statement should be replaced by the *"redundant if"* statement with the reduction rule

$$\mathsf{rif}\ x < [p,q]\ \mathsf{then}\ P\ \mathsf{else}\ Q \to \begin{cases} P & \text{if } x < p \\ \{P,Q\} & \text{if } x \in [p,q] \\ Q & \text{if } x > q \end{cases} \tag{16}$$

where $\{P,Q\}$ denotes a non-deterministic choice between P and Q. This apparently problematic non-determinism can and **must** be annulled by the side condition

$$P = Q \qquad \text{if} \qquad x \in [p,q]\,.$$

The "redundant if" statement is similar to the *quasi-relational comparison* operator $<_\epsilon$ introduced by Boehm and Cartwright [5, 4]. The "redundant if" statement will not be used explicitly in this article. However, it will be used implicitly in statements like "if $f\,(d)$ for some $d \in D$". This notion is also used in the algorithms of section (11) for the transcendental functions by utilizing the idea of analytic continuation in complexity analysis.

9.5 Straightforward Reduction Rules

All the ideas above can be brought together into the following compact algorithm:

$$
\begin{aligned}
\mathsf{sem} &: \mathbb{E} \times \mathbb{N} \to \mathbb{E} \\
\mathsf{sem}\,(E,i) &= \begin{cases} S_\sigma\left\{\mathsf{dem}\left(\mathfrak{D}_0^0, S_\sigma^{-1}[E], i\right)\right\} & \text{if } S_\sigma^{-1} \bullet L \in \mathbb{L}^+ \\ & \text{for some } \sigma \in \{+,\infty,-,0\} \\ \mathsf{sem}\,(L[F_1,\ldots,F_N], i) & \text{otherwise} \end{cases} \\
& \quad \text{where } F_n = \mathsf{ab}\,(L, E_n, \Delta_n(L)) \text{ and } L\{E_1,\ldots,E_N\} = E \\
\mathsf{dem} &: \mathbb{M}^+ \times \mathbb{E}^+ \times \mathbb{N} \to \mathbb{E}^+ \\
\mathsf{dem}\left(\mathfrak{D}_c^i, E, j\right) &= \begin{cases} \mathfrak{D}_c^i\{E\} & \text{if } j = 0 \text{ or } L \in \mathbb{V} \\ \mathsf{dem}\left(\mathfrak{D}_{2c+d}^{i+1}, D_d^{-1}[E], j-1\right) & \text{if } D_d^{-1} \bullet L \in \mathbb{L}^+ \text{ for} \\ & \text{some } d \in \{-1,0,1\} \\ \mathsf{dem}\left(\mathfrak{D}_c^i, L[F_1,\ldots,F_N], j\right) & \text{otherwise} \end{cases} \\
& \quad \text{where } F_n = \mathsf{ab}\,(L, E_n, \Delta_n(L)) \text{ and } L\{E_1,\ldots,E_N\} = E \\
\Delta_n &: \mathbb{L} \to \mathsf{boolean} \\
\Delta_n(L) &= L \notin \mathbb{T} \text{ or } \mathsf{strategy}\,(L) = n \\
\mathsf{ab} &: \mathbb{L} \times \mathbb{E}^+ \times \mathsf{boolean} \to \mathbb{E}^+ \\
\mathsf{ab}\,(K,E,b) &= \begin{cases} \mathfrak{D}_0^0\{E\} & \text{if } b = \text{false} \\ \mathsf{dem}\left(\mathfrak{D}_0^0, E, 1\right) & \text{if } K \in \mathbb{T} \text{ and } L \in \mathbb{T} \\ E & \text{otherwise} \end{cases}
\end{aligned}
$$

The "sign emit" term $\mathsf{sem}\,(E,i)$ partially converts the signed expression tree E into the signed exact floating point representation. In particular, it returns a signed expression tree of the form $S_\sigma\{\mathfrak{D}_c^i\{E'\}\}$ where S_σ is the required sign, $\mathfrak{D}_c^i$ is the i required digits compressed according to equation (8) and E' is an unsigned expression tree for the remaining digits (i.e. a continuation). The "digit emit" term $\mathsf{dem}\,(\mathfrak{D}_c^i, E, j)$ partially converts the unsigned expression tree $\mathfrak{D}_c^i\{E\}$ into the unsigned exact floating point representation. In particular, it returns an unsigned expression tree of the form $\mathfrak{D}_{c'}^{i+j}\{E'\}$ where $\mathfrak{D}_{c'}^{i+j}$ is the $(i+j)$ required digits compressed according to equation (8) and E' is an unsigned expression tree for the remaining digits (i.e. a continuation). The "absorb" term $\mathsf{ab}\,(K,E,b)$ converts the unsigned expression tree E into another unsigned expression tree with a root node ready for absorption (by square bracket application) into its parent node K. The boolean b determines whether any information is actually required.

10 Continued Fractions

The development

$$x = a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{\ddots}}} \tag{17}$$

is called a *continued fraction* [6] and it represents the Cauchy sequence $\langle x_n \rangle_{n\geq 0}$ with limit x where

$$x_n = a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{\ddots \, a_n}}}.$$

A *simple continued fraction* has $b_n = 1$ for all $n \geq 0$. For convenience, we shall also denote the continued fraction in equation (17) by

$$[a_0, b_0; a_1, b_1; a_2, b_2; \ldots].$$

The continued fraction in equation (17) can be converted directly into the general normal product

$$x = \begin{pmatrix} a_0 & b_0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & b_1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \\ 1 & 0 \end{pmatrix} \cdots = \prod_{n=0}^{\infty} \begin{pmatrix} a_n & b_n \\ 1 & 0 \end{pmatrix}$$

provided $a_0, b_0 \in \mathbb{Z}$ and $a_n, b_n \in \mathbb{N}$ for all $n \geq 1$. Otherwise, a sequence of matrices $M_n \in \mathbb{M}$ for all $n \geq 0$ must be found satisfying

$$M_{n-1}^{-1} \bullet \begin{pmatrix} a_n & b_n \\ 1 & 0 \end{pmatrix} \bullet M_n \in \mathbb{M}^+$$

for all $n \geq 1$. In which case

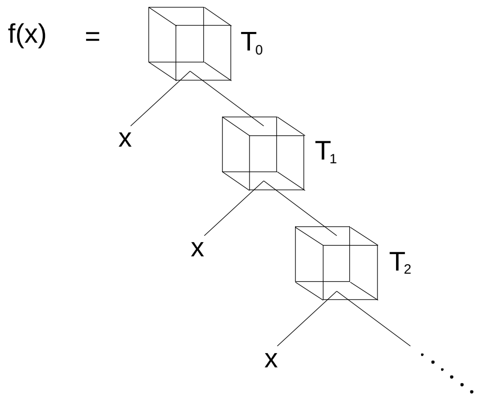
$$x = \left(\begin{pmatrix} a_0 & b_0 \\ 1 & 0 \end{pmatrix} \bullet M_0 \right) \prod_{n=1}^{\infty} \left(M_{n-1}^{-1} \bullet \begin{pmatrix} a_n & b_n \\ 1 & 0 \end{pmatrix} \bullet M_n \right).$$

11 Real Functions

The Taylor series of a function $f(x)$ can often be used to derive a number of continued fractions with the general form

$$f(x) = \alpha_0(x) + \cfrac{\beta_0(x)}{\alpha_1(x) + \cfrac{\beta_1(x)}{\alpha_2(x) + \cfrac{\beta_2(x)}{\ddots}}} \tag{18}$$

including the Stieltjes, Jacobi and Euler types [36, 3, 30]. Sometimes these continued fractions can be converted into an expression tree



(19)

for some $T_0 \in \mathbb{T}$ and $T_n \in \mathbb{T}^+$ for all $n \geq 1$ [30]. Note that the orientation has been carefully chosen to ensure that the strategy in equation (15) works. A general procedure for this is:

Part (i) Using the simple matrix identities

$$\begin{pmatrix} a & c\mu \\ b & 0 \end{pmatrix} \begin{pmatrix} d & f \\ e & 0 \end{pmatrix} = \begin{pmatrix} a & c \\ b & 0 \end{pmatrix} \begin{pmatrix} d & f \\ e\mu & 0 \end{pmatrix} \tag{20}$$

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix} = \begin{pmatrix} c & a \\ d & b \end{pmatrix} \begin{pmatrix} f & h \\ e & g \end{pmatrix} \tag{21}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \prod_{n=0}^{\infty} \begin{pmatrix} a_n & c_n \\ b_n & d_n \end{pmatrix} = \prod_{n=0}^{\infty} \begin{pmatrix} d_n & b_n \\ c_n & a_n \end{pmatrix} \tag{22}$$

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \equiv \begin{pmatrix} a\lambda & c\lambda \\ b\lambda & d\lambda \end{pmatrix} \text{ for } \lambda \neq 0 \tag{23}$$

find

$$T_n = \begin{pmatrix} a_n & c_n & e_n & g_n \\ b_n & d_n & f_n & h_n \end{pmatrix} \in \mathbb{T}$$

such that

$$\prod_{n=0}^{\infty} \begin{pmatrix} \alpha_n(x) & \beta_n(x) \\ 1 & 0 \end{pmatrix} \equiv \prod_{n=0}^{\infty} \begin{pmatrix} a_n x + e_n & c_n x + g_n \\ b_n x + f_n & d_n x + h_n \end{pmatrix}.$$

Part (ii) Find a sequence of matrices $M_n \in \mathbb{M}$ for $n \geq 0$ and a matrix N such that

$$M_{n-1}^{-1} \bullet T_n \bullet_1 N \bullet_2 M_n \in \mathbb{T}^+$$

for all $n \geq 1$. In which case

$$\begin{aligned} f(N(y)) &= (T_0 \bullet_1 N \bullet_2 M_0)\{y, E_1(y)\} \text{ where} \\ E_n(y) &= \left(M_{n-1}^{-1} \bullet T_n \bullet_1 N \bullet_2 M_n\right)\{y, E_{n+1}(y)\} \end{aligned}$$

for all $y \in [0, \infty]$. Note that the matrix N effectively limits the domain of the function to $\mathsf{info}(N)$.

The above technique will be used in the following subsections to derive algorithms for the basic set of transcendental functions; namely square root, natural logarithm, exponential, tangent and inverse tangent. The other transcendental functions can be derived from this basic set.

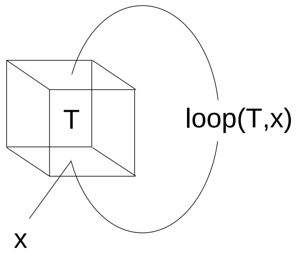
$$
\begin{aligned}
x^y &= \mathsf{exp}\left(y \,\mathsf{log}\left(x\right)\right)\\
\mathsf{sin}\left(x\right) &= \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \left[\mathsf{tan}\left(\frac{x}{2}\right), \mathsf{tan}\left(\frac{x}{2}\right)\right]\\
\mathsf{cos}\left(x\right) &= \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \left[\mathsf{tan}\left(\frac{x}{2}\right), \mathsf{tan}\left(\frac{x}{2}\right)\right]\\
\mathsf{arcsin}\left(x\right) &= \mathsf{arctan}\left(\sqrt{\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} [x, x]}\right)\\
\mathsf{arccos}\left(x\right) &= \mathsf{arctan}\left(\sqrt{\begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} [x, x]}\right)\\
\mathsf{sinh}\left(x\right) &= \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \left[\mathsf{exp}\left(x\right), \mathsf{exp}\left(x\right)\right]\\
\mathsf{cosh}\left(x\right) &= \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \left[\mathsf{exp}\left(x\right), \mathsf{exp}\left(x\right)\right]\\
\mathsf{tanh}\left(x\right) &= \begin{pmatrix} 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \left[\mathsf{exp}\left(x\right), \mathsf{exp}\left(x\right)\right]\\
\mathsf{arcsinh}\left(x\right) &= \mathsf{log}\left(x + \sqrt{x^2+1}\right)\\
\mathsf{arccosh}\left(x\right) &= \mathsf{log}\left(x + \sqrt{x^2-1}\right)\\
\mathsf{arctanh}\left(x\right) &= \frac{1}{2}\mathsf{log}\left(S_\infty\left[x\right]\right)
\end{aligned}
$$

11.1 Square Root

Let $\mathsf{loop}\left(T, x\right)$ be the function represented by the expression tree in equation (19) with $T_n = T$ for all $n \geq 0$. It can be shown, for example, that

$$
\begin{aligned}
\sqrt{x} &= \mathsf{loop}\left(\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}, x\right)\\
x + \sqrt{x^2+1} &= \mathsf{loop}\left(\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, x\right).
\end{aligned}
$$

The usual algorithm for converting an expression tree into an exact floating point number can be improved dramatically for $\mathsf{loop}\,(T, x)$ by employing a feedback mechanism.



For $\mathsf{loop}\,(T, x)$ with $T \in \mathbb{T}^+$ and $x \in [0, \infty]$:

1. Ensure x is an unsigned exact floating point number.

2. While $D_d^{-1} \bullet T \bullet_2 D_d \in \mathbb{T}^+$ for some $d \in \{-1, 0, 1\}$, emit digit D_d and set T to $D_d^{-1} \bullet T \bullet_2 D_d$.

3. Absorb a digit D_d from x, set T to $T \bullet_1 D_d$ and repeat 2.

This algorithm for $\mathsf{loop}\,(T, x)$ can be improved even further if x is a rational number. For $\mathsf{loop}\,(T, V)$ with $T \in \mathbb{T}^+$ and $V \in \mathbb{V}^+$:

1. Set M to $T \bullet_1 \begin{pmatrix} p \\ q \end{pmatrix}$.

2. If $D_{-1}^{-1} \bullet M \bullet D_{-1} \in \mathbb{M}^+$ then set d to -1 else set d to 1.

3. Emit digit D_d, set M to $D_d^{-1} \bullet M \bullet D_d$ and repeat 2.

11.2 Logarithm

The Stieltjes type continued fraction for logarithm [3] is

$$\mathsf{log}\,(1+x) = \left[0, x; 1, \frac{x}{2}; \left\langle 1, \frac{nx}{4n+2}; 1, \frac{(n+1)\,x}{4n+2}; \right\rangle_{n \geq 1}\right]. \tag{24}$$

Note that

$$\begin{aligned}
& \begin{pmatrix} 0 & x \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \frac{x}{2} \\ 1 & 0 \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} 1 & \frac{nx}{4n+2} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \frac{(n+1)x}{4n+2} \\ 1 & 0 \end{pmatrix} \\
= & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & x \\ x & 0 \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} 1 & \frac{n}{2} \\ \frac{n}{2} & 0 \end{pmatrix} \begin{pmatrix} 1 & \frac{x}{2n+1} \\ \frac{x}{2n+1} & 0 \end{pmatrix} \text{ by (20)} \\
\equiv & \prod_{n=0}^{\infty} \begin{pmatrix} 0 & x \\ x & 2n+1 \end{pmatrix} \begin{pmatrix} 0 & n+1 \\ n+1 & 2 \end{pmatrix} \text{ by (23) and (22).}
\end{aligned}$$

Therefore, according to part (i), the corresponding sequence of tensors is

$$T_n = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 2n+1 \end{pmatrix} \bullet_2 \begin{pmatrix} 0 & n+1 \\ n+1 & 2 \end{pmatrix}.$$

Since $T_n \in \mathbb{T}^+$ for all $n \geq 0$, this immediately gives an expression tree for $\mathsf{log}\,(x)$ valid for all $x \in [1, \infty]$. However, we can do better. Note that for $M_n = \begin{pmatrix} 1 & -1 \\ 0 & n+1 \end{pmatrix}$ and $N = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$, $M_{n-1}^{-1} \bullet T_n \bullet_1 N \bullet_2 M_n \in \mathbb{T}^+$ for all $n \geq 1$. Therefore, according to part (ii), we have

$$\begin{aligned} \mathsf{log}\,(x) &= \mathsf{let}\, E_n\,(x) = \begin{pmatrix} n & 2n+1 & n+1 & 0 \\ 0 & n+1 & 2n+1 & n \end{pmatrix} \{x, E_{n+1}\,(x)\}\ \mathsf{in} \\ &\quad \begin{pmatrix} 1 & 1 & -1 & -1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \{x, E_1\,(x)\} \end{aligned}$$

for all $x \in [0, \infty]$. This expression tree is only efficient for x close to 1. However, the domain can easily be reduced to $\left[\frac{1}{2}, 2\right]$ by repeated application of the identity

$$\mathsf{log}\,(x) = \mathsf{log}\left(\frac{x}{2}\right) + \mathsf{log}\,(2)\,.$$

11.3 Exponential

The Jacobi type continued fraction for exponential [11, 12] is

$$\mathsf{exp}\,(x) = \left[1, x; 1 - \frac{x}{2}, \frac{x^2}{12}; \left\langle 1, \frac{x^2}{16n^2 - 4}; \right\rangle_{n \geq 2}\right]. \tag{25}$$

Note that

$$\begin{aligned} &\begin{pmatrix} 1 & x \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 - \frac{x}{2} & \frac{x^2}{12} \\ 1 & 0 \end{pmatrix} \prod_{n=2}^{\infty} \begin{pmatrix} 1 & \frac{x^2}{16n^2-4} \\ 1 & 0 \end{pmatrix} \\ =\ &\begin{pmatrix} 1 & x \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 - \frac{x}{2} & \frac{x}{2} \\ 1 & 0 \end{pmatrix} \prod_{n=2}^{\infty} \begin{pmatrix} 1 & \frac{x}{(4n-2)} \\ \frac{x}{(4n-2)} & 0 \end{pmatrix} \text{ by (20)} \\ \equiv\ &\begin{pmatrix} 2+x & x \\ 2-x & x \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} 4n+2 & x \\ x & 0 \end{pmatrix} \text{ by (23).} \end{aligned}$$

Therefore, according to part (i), the corresponding sequence of tensors is

$$T_n = \begin{cases} \begin{pmatrix} 1 & 1 & 2 & 0 \\ -1 & 1 & 2 & 0 \end{pmatrix} & \text{if } n = 0 \\ \begin{pmatrix} 0 & 1 & 4n+2 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} & \text{if } n \geq 1 \end{cases}.$$

Note that $S_\infty^{-1} \bullet T_n \bullet_1 S_0 \bullet_2 S_\infty \in \mathbb{T}^+$ for all $n \geq 1$. Therefore, according to part (ii), we have

$$\mathsf{exp}\left(S_0\left\{y\right\}\right) = \mathsf{let}\, E_n\left(y\right) = \begin{pmatrix} 2n+2 & 2n+1 & 2n & 2n+1 \\ 2n+1 & 2n & 2n+1 & 2n+2 \end{pmatrix} \left\{y, E_{n+1}\left(y\right)\right\} \;\mathsf{in}\; E_0\left(y\right)$$

for all $y \in [0, \infty]$. This deals with exponential for $x \in [-1, 1]$. For x outside this range, repeatedly apply the identity

$$\mathsf{exp}\left(x\right) = \left(\mathsf{exp}\left(\frac{x}{2}\right)\right)^2$$

until $x \in [-1, 1]$. A general normal product for e can be derived

$$e = \prod_{n=0}^{\infty} \begin{pmatrix} 2n+2 & 2n+1 & 2n & 2n+1 \\ 2n+1 & 2n & 2n+1 & 2n+2 \end{pmatrix} \bullet_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \prod_{n=0}^{\infty} \begin{pmatrix} 2n+2 & 2n+1 \\ 2n+1 & 2n \end{pmatrix}$$

with a good convergence rate because the determinant of each matrix has an absolute value of 1.

11.4 Pi

A very fast formula for π can be derived using Ramanujan's formula [15, 30]

$$\frac{1}{\pi} = \sum_{n=0}^{\infty} (-1)^n \frac{12(6n)!}{(n!)^3(3n)!} \frac{545140134n + 13591409}{(640320^3)^{n+\frac{1}{2}}}. \tag{26}$$

Ramanujan's formula can be viewed as the instantiation of the Taylor expansion of a function $f\left(x\right) = \sum_{n=0}^{\infty} a_n x^n$

$$\begin{aligned} \frac{\sqrt{10005}}{\pi} &= f\left(\frac{1}{640320^3}\right) \\ a_n &= (-1)^n \frac{(6n)!}{(n!)^3(3n)!} \frac{545140134n + 13591409}{426880} \end{aligned}$$

The general formula for the Euler continued fraction [36] of a Taylor expansion $f\left(x\right) = \sum_{n=0}^{\infty} a_n x^n$ is

$$f\left(x\right) = \left[0, a_0; 1, -\frac{a_1}{a_0}x; \left\langle 1 + \frac{a_{n+1}}{a_n}x, -\frac{a_{n+2}}{a_{n+1}}; \right\rangle_{n \geq 0}\right].$$

Note that

$$\begin{aligned} & \begin{pmatrix} 0 & a_0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & -\frac{a_1}{a_0}x \\ 1 & 0 \end{pmatrix} \prod_{n=0}^{\infty} \begin{pmatrix} 1 + \frac{a_{n+1}}{a_n}x & -\frac{a_{n+2}}{a_{n+1}}x \\ 1 & 0 \end{pmatrix} \\ = & \begin{pmatrix} 0 & a_0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \prod_{n=0}^{\infty} \begin{pmatrix} 1 + \frac{a_{n+1}}{a_n}x & 1 \\ -\frac{a_{n+1}}{a_n}x & 0 \end{pmatrix} \quad \text{by (20)} \\ \equiv & \begin{pmatrix} a_0 & 0 \\ 1 & 1 \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} 1 + \frac{a_n}{a_{n-1}}x & 1 \\ -\frac{a_n}{a_{n-1}}x & 0 \end{pmatrix} \end{aligned}$$

and

$$-\frac{a_n}{a_{n-1}}x = \frac{(2n-1)(6n-5)(6n-1)(545140134n+13591409)}{10939058860032000n^3(545140134n-531548725)}.$$

Therefore

$$\begin{aligned}
\frac{\sqrt{10005}}{\pi} &= \begin{pmatrix} 13591409 & 0 \\ 426880 & 426880 \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} b_n - c_n & b_n \\ c_n & 0 \end{pmatrix} \text{ where} \\
b_n &= 10939058860032000n^3(545140134n - 531548725) \\
c_n &= (2n-1)(6n-5)(6n-1)(545140134n+13591409).
\end{aligned}$$

We can do even better than this. For

$$M_n = \begin{pmatrix} 545140135n + 13591410 & 545140133n + 13591408 \\ -n-1 & n+1 \end{pmatrix},$$

$M_{n-1}^{-1} \bullet \begin{pmatrix} b_n - c_n & b_n \\ c_n & 0 \end{pmatrix} \bullet M_n \in \mathbb{T}^+$ for all $n \geq 1$. Therefore it can be shown, with some rescaling, that

$$\begin{aligned}
\frac{\sqrt{10005}}{\pi} &= \begin{pmatrix} 184726412196690 & 184726398605281 \\ 5801900673920 & 5801901100800 \end{pmatrix} \prod_{n=1}^{\infty} Q_n \text{ where} \\
Q_n &= \begin{pmatrix} e_n - d_n - c_n & e_n + d_n - c_n \\ e_n + d_n + c_n & e_n - d_n + c_n \end{pmatrix} \\
d_n &= (2n-1)(6n-5)(6n-1)(n+1) \\
e_n &= 10939058860032000n^4.
\end{aligned}$$

The convergence rate for this general normal product is extremely impressive. Each matrix corresponds to approximately 14 decimal places of information. Note that the entries of the matrix $Q_n \bullet Q_{n+1}$ are divisible by $2(n+1)$. Therefore the entries of the matrix $\prod_{n=1}^{N} Q_n$ are divisible by at least $N!$. However, we conjecture that the entries of the matrix $\prod_{n=1}^{N} Q_n$ are divisible by $\frac{5(N!)^4}{12^{N-1}}$.

11.5 Tangent

A continued fraction by Lambert [21] for tangent is

$$\mathsf{tan}(x) = \left[0, 1; \left\langle \frac{4n+1}{x}, 1; -\frac{4n+3}{x}, 1; \right\rangle_{n \geq 0}\right]. \tag{27}$$

According to part (i),

$$\begin{aligned}
&\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \prod_{n=0}^{\infty} \begin{pmatrix} \frac{4n+1}{x} & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -\frac{4n+3}{x} & 1 \\ 1 & 0 \end{pmatrix} \\
\equiv\ & \prod_{n=0}^{\infty} \begin{pmatrix} 0 & x \\ x & 4n+1 \end{pmatrix} \begin{pmatrix} 0 & x \\ x & -4n-3 \end{pmatrix} \text{ by (23) and (22)}
\end{aligned}$$

and the corresponding sequence of tensors is

$$T_n = \begin{cases} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 2n+1 \end{pmatrix} & \text{if } n \text{ even} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -2n-1 \end{pmatrix} & \text{if } n \text{ odd} \end{cases}$$

for all $n \geq 0$. Note that $S_0^{-1} \bullet T_n \bullet_1 S_0 \bullet_2 S_0 \in \mathbb{T}^+$ for all $n \geq 1$. Therefore, according to part (ii) and the application of equation (21), we have

$$\begin{aligned} \mathsf{tan}\left(S_0\{y\}\right) &= \mathsf{let}\, E_n(y) = \begin{pmatrix} 2n+1 & 2n-1 & 2n+1 & 2n+3 \\ 2n+3 & 2n+1 & 2n-1 & 2n+1 \end{pmatrix} \{y, E_{n+1}(y)\} \text{ in} \\ & \begin{pmatrix} 1 & 1 & -1 & -1 \\ 2 & 0 & 0 & 2 \end{pmatrix} \{y, E_1(y)\} \end{aligned}$$

for all $y \in [0, \infty]$. This deals with tangent for $x \in [-1, 1]$. For x outside this range, repeatedly apply the trigonometric identity

$$\mathsf{tan}(x) = \frac{2\,\mathsf{tan}\left(\frac{x}{2}\right)}{1 - \mathsf{tan}\left(\frac{x}{2}\right)^2} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \left[\mathsf{tan}\left(\frac{x}{2}\right), \mathsf{tan}\left(\frac{x}{2}\right)\right]$$

until $x \in [-1, 1]$. Beware that $\begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \notin \mathbb{T}^+$. Hence, the subtle presence of square brackets instead of round brackets (see section 8.3).

11.6 Inverse Tangent

A continued fraction for inverse tangent [36] is

$$\mathsf{arctan}(x) = \left[0, x; \langle 2n-1, n^2x^2; \rangle_{n\geq 1}\right]. \tag{28}$$

The formula stated by Vuillemin [35] can be derived using equation (20)

$$\begin{pmatrix} 0 & x \\ 1 & 0 \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} 2n-1 & n^2x^2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \prod_{n=1}^{\infty} \begin{pmatrix} 2n-1 & n^2x \\ x & 0 \end{pmatrix}$$

and simplified even further using equation (22) to

$$\prod_{n=1}^{\infty} \begin{pmatrix} 0 & x \\ n^2x & 2n-1 \end{pmatrix}.$$

Therefore, the corresponding sequence of tensors according to part (i) is

$$T_n = \begin{pmatrix} 0 & 1 & 0 & 0 \\ (n+1)^2 & 0 & 0 & 2n+1 \end{pmatrix}.$$

Since, $T_n \in \mathbb{T}^+$ for all $n \geq 0$, this immediately gives an expression tree for $\mathsf{arctan}(x)$ valid for all $x \in [0, \infty]$. However, the efficiency of this expression tree decreases with increasing

x. Note that for $M_n = \begin{pmatrix} 1 & -1 \\ n+1 & n+1 \end{pmatrix}$ and $N = S_0$, $M_{n-1}^{-1} \bullet T_n \bullet_1 N \bullet_2 M_n \in \mathbb{T}^+$ for all $n \geq 1$. Therefore, according to part (ii), we have

$$\begin{aligned} \mathsf{arctan}\left(S_0\left\{y\right\}\right) &= \mathsf{let}\, E_n\left(x\right) = \begin{pmatrix} 2n+1 & n & 0 & n+1 \\ n+1 & 0 & n & 2n+1 \end{pmatrix} \left\{y, E_{n+1}\left(y\right)\right\} \text{ in} \\ & \begin{pmatrix} 1 & 1 & -1 & -1 \\ 2 & 0 & 0 & 2 \end{pmatrix} \left\{y, E_1\left(y\right)\right\}. \end{aligned}$$

This equation for inverse tangent can be used to capture all the extended real numbers when used in conjunction with the following trigonometric identities:

$$\begin{aligned} \mathsf{arctan}\left(S_+\left\{y\right\}\right) &= \mathsf{arctan}\left(S_0\left\{y\right\}\right) + \frac{\pi}{4} \\ \mathsf{arctan}\left(S_\infty\left\{y\right\}\right) &= \mathsf{arctan}\left(S_0\left\{y\right\}\right) + \frac{\pi}{2} \\ \mathsf{arctan}\left(S_-\left\{y\right\}\right) &= \mathsf{arctan}\left(S_0\left\{y\right\}\right) + \frac{3\pi}{4} \end{aligned}$$

12 Information Flow Analysis

One of the advantages of the exact floating point representation over the general normal product representation is that it gives a natural unit of information. Consider the *metric* $d_{\mathrm{J}} : [0, \infty] \times [0, \infty] \longrightarrow [0, 2]$ defined by

$$d_{\mathrm{J}}\left(x, y\right) = \left|S_0(x) - S_0(y)\right|,$$

which is topologically equivalent to the chordal metric in equation (1) restricted to the base interval $[0, \infty]$ [9]. Note that

$$\begin{aligned} d_{\mathrm{J}}\left(D_d\left(x\right), D_d\left(y\right)\right) &= \left|\left(S_0 \bullet D_d\right)\left(x\right) - \left(S_0 \bullet D_d\right)\left(y\right)\right| \\ &= \left|\left(\begin{pmatrix} 1 & d \\ 0 & 2 \end{pmatrix} \bullet S_0\right)\left(x\right) - \left(\begin{pmatrix} 1 & d \\ 0 & 2 \end{pmatrix} \bullet S_0\right)\left(y\right)\right| = \frac{1}{2} d_{\mathrm{J}}\left(x, y\right). \end{aligned}$$

Consequently, the units of information stored in a linear fractional transformation can be estimated by the function $\mathsf{units} : \mathbb{L}^+ \rightarrow \mathbb{N} \cup \{\infty\}$ given by

$$\mathsf{units}\left(L\right) = \left\lfloor 1 - \log_2\left(\mathsf{width}\left(\mathsf{info}\left(S_0 \bullet L\right)\right)\right)\right\rfloor \text{ where } \mathsf{width}\left(\left[x, y\right]\right) = \left|x - y\right| \tag{29}$$

where $\lfloor x \rfloor$ is the greatest integer less than or equal to x. This way, it can be shown that at most $\mathsf{units}\left(L\right)$ digits can be emitted from an arbitrary linear fractional transformation $L \in \mathbb{L}^+$ and at least $\mathsf{units}\left(L\right) - 1$. In particular, $\mathsf{units}\left(V\right) = \infty$ for $V \in \mathbb{V}^+$and for $M = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \in \mathbb{M}^+$

$$\mathsf{units}\left(M\right) = \left\lfloor \log_2\left(\frac{\left|a+b\right|\left|c+d\right|}{\left|\det\left(M\right)\right|}\right)\right\rfloor.$$

12.1 Matrix Lazy Flow Analysis

For an arbitrary matrix $M \in \mathbb{M}^{+}$ and natural number ϵ, we need to find the maximum units of information $\delta(M,\epsilon)$ that can be absorbed into M such that it contains at most ϵ units of information. In other words, such that at most ϵ digits can be emitted. We wish to apply the function $\delta(M,\epsilon)$ dynamically rather than statically in a way analogous to damping in a mechanical system. In this way, we can efficiently get the information we want without demanding even a single unit of information too much from the deeper layers of an expression tree. The problem can be expressed mathematically as:

$$\text{Find maximum } \delta \text{ such that } d_{\mathrm{J}}(M(x), M(y)) \geq \frac{1}{2^{\epsilon}} \text{ whenever } d_{\mathrm{J}}(x,y) \geq \frac{1}{2^{\delta}}.$$

Note that

$$\begin{aligned} d_{\mathrm{J}}(M(x), M(y)) &= |(S_0 \bullet M)(x) - (S_0 \bullet M)(y)| \\ &= \left|\left(S_0 \bullet M \bullet S_0^{-1}\right)(S_0(x)) - \left(S_0 \bullet M \bullet S_0^{-1}\right)(S_0(y))\right| \\ &= \left|\left(S_0 \bullet M \bullet S_0^{-1}\right)'(z)\right| d_{\mathrm{J}}(x,y) \text{ for some } z \in (-1,1) \end{aligned}$$

by the mean value theorem provided that the function $\left(S_0 \bullet M \bullet S_0^{-1}\right)(x)$ is continuous over $[-1,1]$ and differentiable over $(-1,1)$. This condition is certainly satisfied for all $M \in \mathbb{M}^{+}$. Given that $M = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ it can be shown that

$$\left(S_0 \bullet M \bullet S_0^{-1}\right)'(x) = \frac{4\det(M)}{((a+b-c-d)x + (a+b+c+d))^2}.$$

Therefore, if we define the function $\Xi : \mathbb{M}^{+} \to \mathbb{Q}$ by

$$\Xi(M) = \inf\left(\left|\left(S_0 \bullet M \bullet S_0^{-1}\right)'([-1,1])\right|\right) = \frac{|\det(M)|}{(\max(|a+b|, |c+d|))^2}. \tag{30}$$

Therefore, the maximum δ is given by

$$\delta(M,\epsilon) = \epsilon + \lfloor \log_2(\Xi(M)) \rfloor. \tag{31}$$

Finally, we need to convert this formula into an efficient algorithm. Let $\#(a)$ denote the number of bits required to represent the absolute value of the integer a. Using the identities

$$\begin{aligned} \lceil a \rceil &= -\lfloor -a \rfloor \\ \lfloor \log_2 a \rfloor &= \#(a) - 1 \\ \lceil \log_2 a \rceil &= \#(a) - \textsf{if } a \textsf{ is power of } 2 \textsf{ then } 1 \textsf{ else } 0 \\ \lfloor a \rfloor + \lfloor b \rfloor &\leq \lfloor a+b \rfloor \leq \lfloor a \rfloor + \lfloor b \rfloor + 1 \end{aligned}$$

it can be shown that the algorithm

$$\begin{aligned} \Delta_1^{-}(M,\epsilon) = {} & \textsf{let } e = |\det(M)| \textsf{ and } \alpha = \max(|a+b|, |c+d|) \textsf{ in} \\ & \epsilon + \#(e) - \#\left(\alpha^2\right) - \textsf{if } \alpha \textsf{ is power of } 2 \textsf{ then } 0 \textsf{ else } 1 \end{aligned}$$

provides a value equal to or one less than δ given in equation (31) (i.e. it is a conservative approximation). In some cases, due to its conservative nature, even though $\epsilon \geq 1$ and no digits can be emitted, the flow analysis given by Δ_1^{-} indicates that no information needs to be absorbed. So, we really need the function $\Delta_1^{+}(M,\epsilon) = \max\left(1, \Delta_1^{-}(M,\epsilon)\right)$ as well.

12.2 Tensor Lazy Flow Analysis

For an arbitrary tensor $T = \begin{pmatrix} P & Q \end{pmatrix} = \begin{pmatrix} R & \\ & S \end{pmatrix} \in \mathbb{T}^{+}$ and natural number ϵ, we need to find the maximum units of information $\delta_1(T,\epsilon)$ and $\delta_2(T,\epsilon)$ that can be left and right absorbed respectively into T such that it contains at most ϵ units of information. The problem can be expressed mathematically as:

$$\text{Find maximum } \delta_1(T,\epsilon) \text{ such that for all } w \in [0,\infty]$$

$$d_{\mathrm{J}}(T(x,w), T(y,w)) \geq \frac{1}{2^{\epsilon}} \text{ whenever } d_{\mathrm{J}}(x,y) \geq \frac{1}{2^{\delta_1}} \text{ and}$$

$$\text{find maximum } \delta_2(T,\epsilon) \text{ such that for all } w \in [0,\infty]$$

$$d_{\mathrm{J}}(T(w,x), T(w,y)) \geq \frac{1}{2^{\epsilon}} \text{ whenever } d_{\mathrm{J}}(x,y) \geq \frac{1}{2^{\delta_2}}.$$

By symmetry, it is clear that $\delta_1(T,\epsilon) = \delta_2\left(T^{\mathsf{T}},\epsilon\right)$. Note that for $U = S_0 \bullet T \bullet_1 S_0^{-1} \bullet_2 S_0^{-1}$

$$\begin{aligned}
d_{\mathrm{J}}(T(w,x), T(w,y)) &= |(S_0 \bullet T)(w,x) - (S_0 \bullet T)(w,y)| \\
&= |U(S_0(w), S_0(x)) - U(S_0(w), S_0(y))| \\
&= |U_2(S_0(w), z)| \; d_{\mathrm{J}}(x,y) \text{ for some } z \in (-1,1)
\end{aligned}$$

where $U_2(x,y)$ is the partial derivative of U with respect to y. It can be shown that

$$\inf(|U_2([-1,1],[-1,1])|) = \min(\Xi(P), \Xi(Q))$$

using the definition of the function $\Xi : \mathbb{M}^{+} \to \mathbb{Q}$ in equation (30). This means that the algorithms

$$\begin{aligned}
\Delta_1^{-}(T,\epsilon) &= \left(\min\left(\Delta_1^{-}(R,\epsilon), \Delta_1^{-}(S,\epsilon)\right)\right) \\
\Delta_2^{-}(T,\epsilon) &= \left(\min\left(\Delta_1^{-}(P,\epsilon), \Delta_1^{-}(Q,\epsilon)\right)\right)
\end{aligned}$$

provide conservative approximations for δ_1 and δ_2 respectively. Note that the overloading of Δ_1^{-} on matrices and tensors is intentional. In some cases, due to its conservative nature, even though $\epsilon \geq 1$ and no digits can be emitted, the flow analysis given by Δ_1^{-} and Δ_2^{-} indicate that no information needs to be absorbed from the left or from the right. So in these cases, we need to decide whether to absorb just one from the left or just one from the right. So, let us define $\Delta_{\{1,2\}}^{+}(T,\epsilon)$ by

$$\Delta_n^{+}(T,\epsilon) = \mathsf{if}\ \Delta_1^{-}(T,\epsilon) \leq 0\ \mathsf{and}\ \Delta_2^{-}(T,\epsilon) \leq 0\ \mathsf{then}\ \Delta_n(T)\ \mathsf{else}\ \Delta_n^{-}(T,\epsilon).$$

12.3 Efficient Linear Fractional Transformations

It should be noted that it is not always desirable to assimilate a vector or a matrix into its parent node. It depends on whether the vector or the matrix is spatially efficient relative to the information that it contains. The idea is that if a vector or a matrix is efficient then it should be assimilated directly otherwise only the digits $D_{\{-1,0,1\}}$ should be exchanged. Actually, it should also depend on the amount of information ϵ required as well. A particularly useful definition is

$$\begin{aligned} \mathsf{efficient} &: \mathbb{L}^+ \times \mathbb{N} \to \mathsf{boolean} \\ \mathsf{efficient}(L, \epsilon) &= \frac{\#(L)}{n} + \alpha \min(\epsilon, \mathsf{units}(L)) \leq \beta \end{aligned}$$

where α and β are adjustable parameters (e.g. $\alpha = 2$ and $\beta = 32$), $\#(L)$ denotes the total number of bits required to represent the coefficients in L and n is the number of coefficients in L.

12.4 Efficient Reduction Rules

All the efficiency related ideas above can be brought together into the following compact algorithm:

$$\begin{aligned} \mathsf{edem} &: \mathbb{M} \times \mathbb{E}^+ \times \mathbb{N} \to \mathbb{E}^+ \\ \mathsf{edem}\left(\mathfrak{D}^i_c, E, j\right) &= \begin{cases} \mathfrak{D}^i_c\{E\} & \text{if } j = 0 \text{ or } L \in \mathbb{V} \\ \mathsf{edem}\left(\mathfrak{D}^{i+1}_{2c+d}, D_d^{-1}[E], j-1\right) & \text{if } D_d^{-1} \bullet L \in \mathbb{L}^+ \text{ for} \\ & \text{some } d \in \{-1, 0, 1\} \\ \mathsf{edem}\left(\mathfrak{D}^i_c, L\left[F_1^+, \ldots, F_N^+\right], j\right) & \text{otherwise} \end{cases} \\ & \text{where } F_n^+ = \mathsf{eab}\left(L, E_n, \Delta_n^+(L, j)\right) \\ & \text{and } L\{E_1, \ldots, E_N\} = E \\ \mathsf{eab} &: \mathbb{L} \times \mathbb{E}^+ \times \mathbb{Z} \to \mathbb{E}^+ \\ \mathsf{eab}(K, E, \epsilon) &= \begin{cases} \mathfrak{D}^0_0\{E\} & \text{if } \epsilon \leq 0 \\ L\left[F_1^-, \ldots, F_N^-\right] & \text{if } \mathsf{efficient}(L, \epsilon) \text{ and} \\ & (K \notin \mathbb{T} \text{ or } L \notin \mathbb{T}) \\ \mathsf{edem}\left(\mathfrak{D}^0_0, E, \epsilon\right) & \text{otherwise} \end{cases} \\ & \text{where } F_n^- = \mathsf{eab}\left(L, E_n, \Delta_n^-(L, \epsilon)\right) \\ & \text{and } L\{E_1, \ldots, E_N\} = E \end{aligned}$$

The "efficient digit emit" term $\mathsf{edem}(\mathfrak{D}^i_c, E, j)$ partially converts the unsigned expression tree $\mathfrak{D}^i_c\{E\}$ into the unsigned exact floating point representation. In particular, it returns an unsigned expression tree of the form $\mathfrak{D}^{i+j}_{c'}\{E'\}$ where $\mathfrak{D}^{i+j}_{c'}$ is the $(i+j)$ required digits compressed according to equation (8) and E' is an unsigned expression tree for the remaining digits (i.e. a continuation). The "efficient absorb" term $\mathsf{eab}(K, E, \epsilon)$ converts the unsigned expression tree E into another unsigned expression tree with a root node ready for absorption (by square bracket application) into its parent node K. The integer ϵ indicates the maximum required units of information in the root node. Note that Δ_n^+ is used when a specified number of digits $D_{\{-1,0,1\}}$ is required from an expression tree, whereas Δ_n^- is used when a conservative number of units of information is required from an expression tree.

Any efficient implementation must avoid re-evaluating the same expression. This means that the language used must support references, pointers or destructive data types of some sort. For instance, there need only be one instantiation of the argument x in equation (19). Finally, it should be pointed out that rescaling the coefficients of a linear fractional transformation down by their greatest common divisor is inefficient and generally unnecessary. However, rescaling down by 2 is efficient and necessary.

13 Conclusion

We started this article by noting that any real number can be represented by a sequence of nested closed intervals. We then put this idea into the formal context of an incremental digit representation. In particular, we presented the decimal, continued fraction, redundant binary, general normal product and exact floating point representations in this framework. We pointed out that any such sequence can be seen as a chain in the continuous domain of extended real numbers.

We then introduced the notion of an expression tree and linked it to the concept of a directed set in the continuous domain of extended real numbers. We then outlined a general two part procedure for converting any function with a power series representation into an expression tree. This general procedure was applied to the transcendental functions culminating in the introduction of new algorithms for π, e and the transcendental functions.

We presented the "redundant if" statement, which provides a simple and efficient means to overcome various computability problems during the construction of various expression trees. However, the domain theoretic meaning of "redundant if" is not fully understood at this time. Straightforward reduction rules were presented to allow any valid expression tree to be converted incrementally into the exact floating point representation.

The digit set in the exact floating point representation has properties that enable the size of integers to be controlled and the flow of information to be analyzed in an expression tree. As a result, we introduced an efficient algorithm for converting an expression tree into the exact floating point representation. The quantization of information means that the complexity of these reduction rules is amenable to analysis, although this has not been done yet. Although, both the spacial and temporal aspects of exact real arithmetic have been tackled in this article, for many applications the spacial overhead is still unavoidably prohibitive. This is essentially because, in general, the entire history of every variable must be remembered, not just the last value as in a conventional floating point application. Nevertheless, exact real arithmetic is still useful for many small applications where the user wants guaranteed correct answers, such as verification of algorithms. The efficiency of the exact real arithmetic presented in this article can undoubtedly be improved even further by hardware assisted software and parallel processes.

14 Acknowledgements

I would particularly like to thank my PhD supervisor Abbas Edalat for many interesting discussions. I would also like to acknowledge Martín Escardó, Reinhold Heckmann and Philipp Sünderhauf for their comments and suggestions. Lastly, many of the algorithms were derived using the invaluable tool Mathematica® [39] and implemented in CAML [38] with BigNum [16].

References

[1] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, chapter 1,

pages 1–168. Clarendon Press, 1994.

[2] A. Avizienis. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on electronic computers*, (10):389–400, 1961.

[3] G. A. Baker. *Essentials of Padé approximants.* Academic Press, 1975.

[4] H. J. Boehm and R. Cartwright. Exact real arithmetic: formulating real numbers as functions. In D. A. Turner, editor, *Research topics in functional programming.* Almqvist and Wiksell, 1990. The University of Texas Year of Programming Series.

[5] H. J. Boehm, R. Cartwright, M. J. O'Donnel, 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.* ACM, 1986.

[6] C. Brezinski. *History of continued fractions and Padé approximants*, volume 12 of *Springer series in Computational mathematics.* Springer Verlag, 1991.

[7] A. Church. An unsolvable problem in elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[8] A. Edalat. Domains for computation in mathemetics, physics and exact real arithmetic. *Bulletin of Symbolic Logic*, 1997. To appear.

[9] A. Edalat and P. J. Potts. A new representation for exact real numbers. In *Proceedings of Mathematical Foundations of Programming Semantics 13*, volume 6 of *Electronic Notes in Theoretical Computer Science.* Elsevier Science B. V., 1997. Available from http://www.elsevier.nl/locate/entcs/volume6.html.

[10] M. H. Escardó. PCF extended with real numbers. *Theoretical Computer Science*, 162(1):79–115, August 1996.

[11] L. Euler. Commentatio in fractionem continuam qua illustris La Grange potestates binomiales expressit. *Mémoires Acad. impér. Sci. Petersb.*, 6:3–11, 1813-1814.

[12] L. Euler. An essay on continued fractions. *Mathematical Systems Theory*, 18:295–328, 1985.

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

[14] R. W. Gosper. Continued fraction arithmetic. Technical Report HAKMEM Item 101B, MIT AI MEMO 239, MIT, February 1972. Available from `ftp://ftp.netcom.com/pub/hb/hbaker/hakmem`.

[15] G. Hardy. *Ramanujan's collected papers.* Chelsea Publishing Company, 1962.

[16] J.-C. Hervé, F. Morain, D. Salesin, B. Serpette, J. Vuillemin, and P. Zimmermann. BigNum: a portable and efficient package for arbitrary-precision arithmetic. Technical report.

[17] P. Kornerup and D. W. Matula. An on-line arithmetic unit for bit-pipelined rational arithmetic. *Journal of Parallel and Distributed Computing*, 5(3):310–330, 1988.

[18] P. Kornerup and D. W. Matula. Exploiting redundancy in bit-pipelined rational arithmetic. In *Proceedings of the* 9^{th} *IEEE Symposium on Computer Arithmetic*, pages 119–126, Santa Monica, 1989. IEEE Computer Society Press.

[19] P. Kornerup and D. W. Matula. An algorithm for redundant binary bit-pipelined rational arithmetic. *IEEE Transactions on Computers*, C-39(8):1106–1115, 1990.

[20] P. Kornerup and D. W. Matula. Finite precision lexicographic continued fraction number systems. In *Proceedings of the* 7^{th} *IEEE Symposium on Computer Arithmetic*, pages 207–214, Urbana, 1995. IEEE Computer Society Press.

[21] J. H. Lambert. *Beiträge zum gebrauch der mathematik und deren anwendung*, volume 1 of *r. II.* Zweiten Theil, Berlin, 1770.

[22] D. R. Lester. Vuillemin's exact real arithmetic. In R. Heldal, C. K. Holst, and P. L. Wadler, editors, *Functional Programming, Glasgow 1991: Proceedings of the 1991 Workshop, Portree, UK*, pages 225–238, Berlin, 1992. Springer Verlag.

[23] C. Mazenc. On the redundancy of real number representation systems. Technical Report Research Report 93-16, LIP, Ecole Normale Supérieure de Lyon, 1993. Available from `ftp://ftp.lip.ens-Lyon.fr/ pub/LIP/Rapports/RR/RR93/RR93-16.ps.Z`.

[24] V. Ménissier-Morain. Arbitrary precision real arithmetic: design and algorithms. Submitted to the Journal of Symbolic Computation, September 1996. Available from `ftp://ftp.inria.fr/INRIA/Projects/cristal/Valerie.Menissier/submission_JSC.ps.gz`.

[25] R. E. Moore. *Interval Analysis.* Prentice Hall, Englewood Cliffs, 1966.

[26] J. Myhill. What is a real number? *American Mathematical Monthly*, 79(7):748–754, 1972.

[27] P. M. Neumann, G. A. Stoy, and E. C. Thompson. *Groups and geometry.* Oxford Science Publications, 1995.

[28] A. M. Nielsen and P. Kornerup. MSB-first digit serial arithmetic. *Journal of Universal Computer Science*, 1(7):527–547, July 1995.

[29] S. L. Peyton-Jones. Arbitrary precision arithmetic using continued fractions, 1984. INDRA Note 1530, University College London.

[30] P. J. Potts. Computable real arithmetic using linear fractional transformations, June 1996. Early draft PhD Thesis, Imperial College, available from `http://www-tfm.doc.ic.ac.uk/~pjp`.

[31] P. J. Potts and A. Edalat. Exact real arithmetic based on linear fractional transformations, December 1996. Imperial College, available from `http://www-tfm.doc.ic.ac.uk/~pjp`.

[32] P. J. Potts and A. Edalat. Exact real computer arithmetic. Technical Report DOC 97/9, Imperial College, March 1997. `http://www-tfm.doc.ic.ac.uk/~pjp`.

[33] P. J. Potts, A. Edalat, and M. H. Escardó. Semantics of exact computer arithmetic. In *Twelfth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1997.

[34] A. M. Turing. On computable numbers with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society*, (42):230–265, 1936.

[35] J. Vuillemin. Exact real computer arithmetic with continued fractions. *IEEE Transactions on computers*, 39(8):1087–1105, August 1990.

[36] H. S. Wall. *Analytic theory of continued fractions.* Chelsea Publishing Company, 1973.

[37] O. Watanuki and M. D. Ercegovac. Error analysis of certain floating-point on-line algorithms. *IEEE Transactions on Computers*, C-32(4):352–358, April 1983.

[38] P. Weis, M. Aponte, A. Laville, M. Mauny, and A. Suárez. The CAML reference manual. Technical Report 121, INRIA, Domaine de Voluceau, 78153 Rocquencourt, FRANCE, September 1990.

[39] S. Wolfram. *Mathematica: a system for doing mathematics by computer.* Almqvist and Wiksell, 2 edition, 1991.