Speculatively Redundant Continued Logarithm Representation

Tomáš Brabec

Abstract—Continued logarithms, as originally introduced by Gosper, represent a means for exact rational arithmetic, but their application to exact real arithmetic is limited by the uniqueness of their representation. This is quite unfortunate, as this representation seems promising for efficient hardware implementation. We propose an idea of making the representation redundant using speculative recognition of noncomputable cases. This approach solves the problem of real number computability, preserves most of the beneficial properties of continued logarithms, and only moderately affects complexity of arithmetic algorithms, thus, keeping the prospect of efficient implementation.

Index Terms—Computer arithmetic, representation of numbers, continued fraction, redundancy, computable real numbers, exact arithmetic.

1 Introduction

Continued fractions represent an excellent theoretical paradigm [1], [2] for arbitrarily precise numerical calculations. Their software implementation is unfortunately very inefficient, being the major reason for their rejection [3], [4], [5]. The character of their arithmetic suggests, however, that specific hardware support could significantly improve this efficiency [6], [7]. The key is to combine the principles of continued fractions with the binary nature of digital circuits.

There exist but few such binary representations: continued logarithms (CL) [2], lexicographical continued fractions (LCF) [8], and Stern-Brocot tree (SB) representation [9]. All of them are basically nonredundant and, as such, of limited use for exact real arithmetic. The issue with nonredundancy is that different representations denote different numbers. We may, thus, experience problems such as with infinite decimal expansion $x = 0.33\ldots$ multiplied by three; uncertain about x being eventually below or above $\frac{1}{3}$, one cannot say if the result is $0.9\ldots$ or $1.0\ldots$ To establish the real number computability, one has to make these representations redundant [10], such as by using signed digits. Both LCF and SB were shown to be extensible in this sense [11], [9].

The principal account of this paper is to show that even CL representation can be made redundant and, thus, useful for exact real arithmetic. Our redundant extension is intentionally designed to be compatible with that of [7] and to affect algorithms developed therein as little as possible. In this way, the rather theoretical results arrived at in this paper could be easily embedded in a practical hardware implementation. Whether or not it is worth of the effort, such a specialized hardware may improve the performance by at least an order [7].

• *The author is with the Faculty of Electrical Engineering, Department of Computer Science and Engineering, Czech Technical University, Karlovo nam. 13, 12135 Prague 2, Czech Republic. E-mail: brabect1@fel.cvut.cz.*

Manuscript received 22 July 2007; revised 2 May 2008; accepted 2 Dec. 2009; published online 14 May 2010.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2007-07-0351.
Digital Object Identifier no. 10.1109/TC.2010.110.

We introduce the paper by covering related work and reporting how our contribution fits within. Next, we provide a brief overview of "standard" continued logarithm arithmetic (Section 3). Starting with Gosper's original representation [2], we extend it in the sense of [7] to support full-rational arithmetic and revise its basic properties and principles. We close the section discussing the computability issues.

The redundant extension, inspired by Gosper [2], is introduced in Section 4. It uses speculative approach as a "technical" solution to handle noncomputable cases. After analyzing these cases, we formalize the new representation and revise the arithmetic algorithms. It is shown that a naive use of speculation may exhibit some undesired properties. However, if the outcome of speculation is interpreted properly, the arithmetic will provide the expected results. Proof of this core statement is left to an online appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2010.110 [12].

Section 5 demonstrates some practical effects of the proposed speculative concept. Samples of computing several functions are used therein to verify computability and some other properties of the new representation. We also discuss the increased complexity of redundant algorithms. However, as the complexity comes at the expense of redundancy, our proposal allows to minimize its overhead by using speculation only in situations that seem to be otherwise uncomputable.

Conclusions and implications of the proposed redundancy on hardware architecture of nonredundant continued logarithms are finally discussed in Section 6. Therein, we also open a number of questions relating to the use of such architecture to build a highly parallel system for exact real arithmetic.

2 Related Work

This paper basically follows the well-known work of Gosper [2]. Therefrom, we took over the proposal of continued

logarithms and extended it for rational [7] and here also for computable real arithmetic. The idea of speculation, though not called so, comes from [2] too. Herein, we elaborate more on this concept, formalize it, and verify its validity. Regarding the implementation, Gosper proposed to make arithmetic operations input and output elements of a continued logarithm in every cycle, taking the risk of retraction (i.e., of a temporarily inaccurate result). We believe that incurred loss of incremental character is not of that price and our algorithm carefully avoids any retraction.

In Section 1, we recognized some alternative approaches. The most related seems the work of Kornerup and Matula on application of continued fractions in hardware. In [11], they introduced the redundant admissible representation (RACF) combining principles of redundancy and bit-level encoding. RACF directly extends simple continued fractions; the bit-level encoding is established by prefixing binary representation of each partial quotient with a self-delimiting unary part. Redundancy is in RACF employed on two levels: on the bit level by using signed-digit binary representation, and on the partial quotient level. Likewise continued logarithms, RACF uses only add/sub and shift operations for arithmetic operations (see [13]), thus, being equally useful for hardware implementation. Logarithmic nature of the two representations brings them even closer. However, built atop simple continued fractions, RACF has better approximation capabilities resulting in shorter expansions and smaller bit length of internal variables [13]. Kornerup's and Matula's representation, thus, seems to be more advantageous. A thorough comparison of our proposed representation with RACF and other alternatives is a subject of our present research, though.

More recently, another binary encoding of simple continued fractions appeared in work of Niqui [9], and is in principle equivalent to earlier results of Raney [14]. The binary character is obtained via Stern-Brocot tree encoding [9], which shares some ideas of continued logarithms [13]. Nonetheless, while continued logarithms "approach" the represented value in logarithmic scale, Stern-Brocot encoding does the same with a linear granularity [14]. Stern-Brocot expansions are, thus, extremely long, making the representation practically infeasible and uncompetitive with continued logarithms.

Closely related is also work of exact arithmetic group at Imperial College (Edalat, Potts, Heckmann, and others). Even though not primarily dealing with continued fractions, they developed a framework for general arithmetic based on linear fractional transformations (LFTs). Representations they used [15], [16], [17] were derived from or equal to a signed-digit radix representation; with a radix $r = 2$, one gets another binary representation suitable for hardware implementation.

Especially remarkable are their results [16] on complexity of arithmetic operations. Unfortunately, being derived for that particular representation, these results do not provide much useful information when applied to continued logarithms. We, thus, have no results that would quantify the theoretical complexity of our proposed representation. In fact, it seems to be a nature of continued fractions, in general, to be difficult for complexity analysis [3], [18].

3 Continued Logarithms

3.1 Representation

Continued logarithms were introduced by Gosper [2] as a means for high-speed serial, rational arithmetic, which could be easily implemented in hardware. This comes from its purely binary character, expressed in Definition 1. There is also an alternative form, which we call *canonical* (see Definition 2). While the former is central to this text, we use the latter to demonstrate some of the core principles.

In the following, we use $\mathbb{R}^*$ to denote an extended real line $\mathbb{R} \cup \{\infty\}$ as a one-point compactification [17] of a real line $\mathbb{R}$. This includes conventional extension of arithmetic operations such as $-\infty = \infty$, $\frac{x}{0} = \infty$ if $x \neq 0$, etc. For completeness, one should also consider undefined outcome of operations such as $\frac{\infty}{\infty}$ or $\infty - \infty$, but we omit them here for simplicity.

Definition 1. *Binary CL representation of a real number $x \geq 1$ is a sequence of* bits $\{b_i\}$, $b_i \in \{0, 1\}$, $i = 0, 1, \ldots$, *such that there is a corresponding sequence $\{x_i\}$, $x_i \in \mathbb{R}^*$, and $x_0 = x$, satisfying*

$$1 \leq x_{i+1} = \begin{cases} x_i/2, & \text{for } x_i \geq 2, \\ 1/(x_i - 1), & \text{otherwise}, \end{cases} \tag{1}$$

and

$$b_i = \begin{cases} 1, & \text{for } x_i \geq 2, \\ 0, & \text{otherwise}. \end{cases}$$

If there is n such that $x_n = \infty$, the sequence $\{b_i\}$ may be terminated, even with omission of b_n.

Definition 2. Canonical continued logarithm *(CCL) representation of a real number $x \geq 1$ is a sequence $\{k_i\}$, $k_i \in \mathbb{N} \cup \{0\}$, $i = 0, 1, \ldots$, such that there is a corresponding sequence $\{x_i\}$, $x_i \in \mathbb{R}^*$, and $x_0 = x$, satisfying*

$$2^{k_i} \leq x_i < 2^{k_i+1}, \tag{2}$$

and

$$x_{i+1} = 2^{k_i}/(x_i - 2^{k_i}). \tag{3}$$

The sequence $\{k_i\}$ is terminated at $i = n - 1$ for an n such that $x_n = \infty$, considering k_n to be infinite.

This latter form establishes a link between continued logarithms and continued fractions, which is seen by expanding the inverted recurrent relation of (3), $x_i = 2^{k_i}(1 + 1/x_{i+1})$:

$$x = x_0 = 2^{k_0} + \cfrac{2^{k_0}}{2^{k_1} + \cfrac{2^{k_1}}{\ddots + \cfrac{2^{k_{i-1}}}{x_i}}} = \ldots \tag{4}$$

The canonical form has also proved to be more convenient for theoretical analysis [19], while the binary form is more useful for practical implementation [7]. But otherwise, the two forms are equivalent as every k_i could be factored into a sequence of 1 bits of length k_i, followed by a 0 bit.

Regarding the terminology, we denote b_i and k_i as *elements* and call their sequences *expansions*. Real numbers x_i are due to their purpose called *continuations*.

Example 1. Find representation of $x = 26/7$ in both continued logarithm forms. Following directly the

TABLE 1
Continued Logarithm Representation of $x = 26/7$

	i	0	1	2	3	4	5	6	7	8
CL	x_i	$\frac{26}{7}$	$\frac{13}{7}$	$\frac{7}{6}$	6	3	$\frac{3}{2}$	2	1	∞
	b_i	1	0	0	1	1	0	1	0	1
CCL	x_i	$\frac{26}{7}$	$\frac{7}{6}$	6	2	∞				
	k_i	1	0	2	1	∞				

corresponding definitions, we set $x_0 = x$ and iterate (1) or (3). In both cases, the iteration terminates once $x_i = \infty$.

For the binary representation, the first two iterations are shown below, complete results are given in Table 1. Excluding the last digit (allowed by definition), we get $x = (10011010)_{\mathrm{CL}}$.

$$x_0 = \frac{26}{7} \geq 2 \rightarrow b_0 = 1 \rightarrow x_1 = \frac{26/7}{2},$$
$$x_1 = \frac{13}{7} < 2 \rightarrow b_1 = 0 \rightarrow x_2 = \frac{1}{13/7 - 1}.$$

Table 1 also summarizes results for the canonical representation, first two of which are computed below. We shall remark that a k_i satisfying (2) can be computed as $\lfloor \log_2 x_i \rfloor$. The CCL representation is, thus, $x = (1, 0, 2, 1)_{\mathrm{CCL}}$.

$$x_0 = \frac{26}{7} \rightarrow k_0 = \lfloor 1.8 \ldots \rfloor = 1 \rightarrow x_1 = \frac{2}{26/7 - 2},$$
$$x_1 = \frac{7}{6} \rightarrow k_1 = \lfloor 0.2 \ldots \rfloor = 0 \rightarrow x_2 = \frac{1}{7/6 - 1}.$$

Immediate disadvantage of continued logarithms is their restriction to real numbers from the *base domain* $\mathbb{B} = [1, \infty]$. This can be easily handled through an *intuitive extension* by additive and multiplicative inverse [7]. There are, of course, other ways to extend Definition 1 to the whole real line, e.g., [2], [20], but the intuitive extension naturally fits the principle of redundancy introduced later in the text. From now on, we will further consider CL in this extended form.

Definition 3. Extended *binary continued logarithm representation of a real number $x \in \mathbb{R}^*$ is a sequence of* digits $\{b_i\}$, $b_i \in \{0, 1, /, -\}$, $i = 0, 1, \ldots$, *such that there is a corresponding sequence* $\{x_i\}$, $x_i \in \mathbb{R}^*$, *and* $x_0 = x$, *satisfying*

$$x_{i+1} = \begin{cases} x_i/2, & \text{for } x_i \in \mathbb{B}_1 = [2, \infty], \\ 1/(x_i - 1), & \text{for } x_i \in \mathbb{B}_0 = [1, 2), \\ 1/x_i, & \text{for } x_i \in \mathbb{B}_/ = [0, 1), \\ -x_i, & \text{for } x_i \in \mathbb{B}_- = (\infty, 0), \end{cases} \tag{5}$$

and

$$b_i = c \in \{0, 1, /, -\},$$

for a c such that $x_i \in \mathbb{B}_c$. If there is n such that $x_n = \infty$, the sequence $\{b_i\}$ may be terminated, even with omission of b_n.

Example 2. Consider the number $y = -7/26$. One could again follow the definition, but, as $y = -1/x$ with the representation of x taken from Example 1, we may directly write $y = (-/10011010)_{\mathrm{CL}}$.

Definition 3 introduced two additional cases of $\mathbb{B}_/$ and $\mathbb{B}_-$. By their character of inverses, the corresponding symbols "/" and "−" may appear only at the beginning of the representation. We say that they have a *sign character*. It follows that, from an implementation perspective, one would not need to change the existing binary encoding except for adding a sign identification. Nonetheless, after introducing the redundancy principle, the purely sign character of "/" and "−" will change, and it is, thus, better to directly assume each *binary digit* b_i (instead of *bit*) to be encoded with multiple bits (two bits would suffice at this time).

The sign character also implies that once the value of a continuation x_i falls into the base domain, which will happen no longer than for $i = 2$, then every subsequent continuation will retain this domain, i.e., $x_i \in \mathbb{B}$, $i \geq 2$. Such a domain restriction is important for continued logarithm arithmetic, as we will see in Section 3.4.

3.2 Basic Properties

Illustrated by (4), canonical continued logarithms have a form of continued fractions and the way how elements k_i are constructed suggests a similarity with simple continued fractions [1]. While elements of the latter are found via linear approximation of continuations by $a_i = \lfloor x_i \rfloor$, CCL uses logarithmic approximation by $k_i = \lfloor \log_2 x_i \rfloor$. It may be thanks to this similarity that continued logarithms possess a number of properties, for which simple continued fractions are so interesting. Namely:

- uniqueness of representation,
- finite expansions for rational numbers, and
- principle of convergents as rational approximations to a numerical quantity being represented.

From the equivalence of the two continued logarithm forms, these results may be generalized to the extended CL representation.

Uniqueness of CCL representation follows from the way of constructing elements k_i. In this property, CL representation differs slightly as Definition 1 is indefinite in terminating the sequence $\{b_i\}$. The definition says that, once $x_n = \infty$ for some n, the sequence "may" terminate. It "may," however, not terminate till some x_{n+k}, $k > 0$, which is due to equivalence $\frac{\infty}{2} = \infty$, or it "may" even go on producing "1"s forever. We, thus, get the following equivalence:

$$\infty = ()_{\mathrm{CL}} = (1)_{\mathrm{CL}} = (1 \ldots 1)_{\mathrm{CL}} = (1 \ldots 1 \ldots)_{\mathrm{CL}}.$$

This result is indeed similar to the aspect of "trailing zeros" known from the conventional decimal representation.

To partially remove that ambiguity, we hereinafter agree to avoid "unnecessary" trailing "1"s of CL representation whenever possible. That is, being sure that $x_n = \infty$, we immediately terminate the computed CL representation. Thanks to this agreement, we may, e.g., consider CL representation of rational numbers to be finite. We have reasons, however, for which not to remove the ambiguity completely. It is because of the additional information on accuracy that the trailing "1"s provide in cases where we cannot be sure of the real value of the represented number (see Example 4 in Section 4).

Another useful property of continued logarithms is finiteness of expansions of rational numbers. It is obvious

that any finite expansion, whether CL or CCL, represents a rational number. The opposite implication is thanks to the uniqueness (recall our agreement made for CL). This also means that irrational numbers have infinite (i.e., nonterminating) expansions. This property makes continued logarithms more interesting over traditional positional representations as in purely rational arithmetic we always deal with finite objects.

The last property we consider is the rational approximation by means of convergents. An *nth order convergent* of a CCL representation of $x = (k_0, k_1, \ldots, k_n, \ldots)_{\mathrm{CCL}}$ is the rational number $\frac{p_n}{q_n}$ represented by the expansion prefix of length $n+1$, i.e., $\frac{p_n}{q_n} = (k_0, k_1, \ldots, k_n)_{\mathrm{CCL}}$. These convergents can be shown [20] to satisfy the following recurrent formula

$$\frac{p_n}{q_n} = \frac{2^{k_n} p_{n-1} + 2^{k_{n-1}} p_{n-2}}{2^{k_n} q_{n-1} + 2^{k_{n-1}} q_{n-2}}, \quad n = 0, 1, \ldots,$$

where $p_{-1} = q_{-2} = 1$ and $p_{-2} = q_{-1} = 0$.

Importance of convergents comes from the fact that they form a sequence of approximations with a distance from the approximated quantity x given by [20]:

$$\left| x - \frac{p_n}{q_n} \right| < \left| \frac{p_n}{q_n} - \frac{p_{n-1}}{q_{n-1}} \right| = \frac{\prod_{i=0}^{n-1} 2^{k_i}}{q_n q_{n-1}}. \tag{6}$$

This distance estimates the error we commit by prematurely terminating x's expansion. As the error bound decreases with increasing n, it gives continued logarithms the *incremental character*. That is, extending a prefix of x's expansion with another element improves the accuracy of the approximation.

The notion of convergents can be generalized for the binary continued logarithms too. Due to the finer granularity, however, the binary convergents form a super set of those defined above.

3.3 LFTs and Linear Algebra

In context of continued logarithm arithmetic, linear fractional transformations play an important role. Their importance is actually seen twice—first, as a character of the representation, and second, as the essential principle of the arithmetic. By similarity of linear fractional transformations and two-by-two matrices, most of the principles can be expressed in a form of matrix algebra, which simplifies understanding of these principles.

Definition 4. *An LFT is a function* $\phi : \mathbb{R}^* \to \mathbb{R}^*$ *of the form*

$$\phi(x) = \frac{ax + b}{cx + d},$$

where $a, b, c, d \in \mathbb{Z}$ *are parameters (called* fractional coefficients*) such that* $ad - bc \neq 0$, $\phi(\infty) = \frac{a}{c}$, *and* $\phi(-\frac{d}{c}) = \infty$ *if* $c \neq 0$.

LFTs have a number of interesting properties. For us, the most important is that the set of all LFTs forms a group under composition $\circ$ and that every LFT $\phi(x) = \frac{ax+b}{cx+d}$ has an inverse $\phi^{-1}(x) = \frac{dx-b}{-cx+a}$, such that $\phi^{-1} \circ \phi(x) = \phi \circ \phi^{-1}(x) = x$.

The close correspondence between LFTs and two-by-two matrices is established by making elements of a matrix $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ represent coefficients of an LFT $\phi_A(x) = \frac{ax+b}{cx+d}$, writing $A \cong \phi_A$. If we consider the equivalence

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} ax + b \\ cx + d \end{pmatrix},$$

and extend $\cong$ for vectors too, meaning $\left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right) \cong \frac{a}{b}$, we may write

$$A \cdot \begin{pmatrix} x \\ 1 \end{pmatrix} \cong \phi_A(x).$$

This result just supports the correspondence between LFTs and matrices. Further similarity follows from the fact that matrices form a group closed under multiplication and it is easy to verify for matrices $A \cong \phi_A$ and $B \cong \phi_B$ that

$$A \cdot B \cong \phi_A \circ \phi_B.$$

That is, matrix multiplication is equivalent to LFT composition. As with LFTs, it makes sense to consider nonsingular matrices such that $ad - bc = \det A \neq 0$. A small difference, making the mapping $\cong$: $A \mapsto \phi_A$ surjective, is seen in that $kA \cong \phi_A$ for a constant $k \in \mathbb{Q} \setminus \{0\}$. That is, any common factor of a matrix may be canceled without affecting the related LFT. For this reason, we consider a matrix *pseudoinverse* A^* [17] instead of the real inverse A^{-1}:

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

It is easy to verify that

$$A \cdot A^* = A^* \cdot A = \det A \cdot E,$$

where $E = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. This inverse relation is preserved by $\cong$ mapping and, thus, $\phi_A \circ \phi_{A^*}(x) = \phi_{A^*} \circ \phi_A(x) = x$.

Thinking of LFTs as matrices has advantages especially in replacing compositions with matrix multiplication, where the latter is easier at least when computed by hand. In the following text, we will, thus, use matrices and LFTs interchangeably, preferring the one that will be more appropriate in a situation.

Extremely helpful is also the interval notation. For $\mathbb{X} \subseteq \mathbb{R}^*$ and a function $\phi : \mathbb{R}^* \to \mathbb{R}^*$, we define $\phi(\mathbb{X})$ as

$$\phi(\mathbb{X}) = \{\phi(x) \mid x \in \mathbb{X}\}.$$

Extension to multiple dimensions is immediate. Although $\mathbb{X}$ can be a general set, we will be mostly working with intervals (or their union). It is no surprise, that if $\mathbb{X}$ is an interval and ϕ is a monotone and continuous on it, then $\phi(\mathbb{X})$ is delimited by values of ϕ in end points of the interval $\mathbb{X}$.

3.4 Arithmetic Algorithms

Continued logarithms, in either form, have an *LFT character*, meaning that they can be written as a composition of LFTs. This comes form the recursive relations in (3) and (5). Herein, we will consider only the latter. Thus, if we define $\beta_{b_i} \cong B_{b_i}$, with $B_{b_i} \in \{\left(\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right)\}$ for $b_i \in \{1, 0, /, -\}$, we may rewrite (5) as $x_i = \beta_{b_i}(x_{i+1})$ and following the recursion we obtain

$$x_0 = \beta_{b_0}(x_1) = \beta_{b_0} \circ \beta_{b_1}(x_2) = \beta_{b_0} \circ \beta_{b_1} \circ \cdots \circ \beta_{b_i}(x_{i+1}) = \ldots$$

With the matrix notation, we may "encode" an ith digit b_i into a corresponding *digit matrix*[1] B_{b_i} and write a binary continued logarithm x as $x \cong \prod_i B_{b_i} \cdot \binom{\infty}{1}$, or just simply $x \cong \prod_i B_{b_i}$.

The LFT character is essential for continued logarithm arithmetic as it ensures that no rational transformation will ever change this character. More formally,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \prod_{i=0} B_{b_i} = \begin{pmatrix} a^{(0;n)} & b^{(0;n)} \\ c^{(0;n)} & d^{(0;n)} \end{pmatrix} \cdot \prod_{i=n} B_{b_i} = \prod_{i=0}^{m-1} B_{b'_i} \cdot \begin{pmatrix} a^{(m;n)} & b^{(m;n)} \\ c^{(m;n)} & d^{(m;n)} \end{pmatrix} \cdot \prod_{i=n} B_{b_i},$$

where $x \cong \prod_i B_{b_i}$, $\phi \cong \binom{a\ b}{c\ d}$, and $\phi(x) \cong \prod_i B_{b'_i}$. The notation $\cdot^{(m;n)}$ indicates that $\cdot$ has been affected by processing n digits of x and m digits of $\phi(x)$.

This simple idea is especially due to Gosper [21] and it allows computing any rational function of one variable by properly initializing coefficients a, b, c, d. We may eventually drop the condition of $ad - bc \neq 0$, but exclude combinations of a, b, c, d and x leading to undefined expressions.

Example 3. Let us compute $\phi(x) = \frac{1}{x} - 1$ for $x = \sqrt{2} = (0101101\ldots)_{\mathrm{CL}}$. We consider x's prefix of length $n = 4$ obtaining

$$\begin{aligned} \phi^{(0;4)} &\cong \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \cdot B_0 \cdot B_1 \cdot B_0 \cdot B_1 \\ &\cong \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} -2 & 0 \\ 6 & 2 \end{pmatrix}. \end{aligned}$$

In this state, it is $\phi(x) = \phi(x_0) = \phi^{(0;4)}(x_4)$. Since $x_i \in \mathbb{B} = [1, \infty]$ for $i \geq 2$ and $\phi^{(0;4)}$ is monotone and continuous on $\mathbb{B}$, it is $\phi^{(0;4)}(\mathbb{B}) \subseteq [\phi^{(0;4)}(\infty), \phi^{(0;6)}(1)] = [-\frac{1}{3}, -\frac{1}{4}] \subset \mathbb{B}_-$.

Hence, we can output $b'_0 = -$. To factor the corresponding digit matrix out of an LFT, we use the following equality: $\phi^{(m;n)} = \beta_{b'_m} \circ \phi^{(m+1;n)}$. Thus,

$$\begin{aligned} \phi^{(1;4)} = \beta_-^{-1} \circ \phi^{(0;4)} &\cong B_-^* \cdot \begin{pmatrix} -2 & 0 \\ 6 & 2 \end{pmatrix} = \begin{pmatrix} -2 & 0 \\ -6 & -2 \end{pmatrix} \\ &\cong \begin{pmatrix} 2 & 0 \\ 6 & 2 \end{pmatrix}. \end{aligned}$$

Let us repeat the process as long as there is a digit to emit:

$$\phi^{(1;4)}(\mathbb{B}) \subseteq \left[\frac{1}{4}, \frac{1}{3}\right] \quad \rightarrow \quad b'_1 = /, \quad \phi^{(2;4)} \cong \begin{pmatrix} 6 & 2 \\ 2 & 0 \end{pmatrix},$$

$$\phi^{(2;4)}(\mathbb{B}) \subseteq [3, 4] \quad \rightarrow \quad b'_2 = 1, \quad \phi^{(3;4)} \cong \begin{pmatrix} 3 & 1 \\ 2 & 0 \end{pmatrix}.$$

Since $\phi^{(3;4)}(\mathbb{B}) \subseteq [\frac{3}{2}, 2]$, we cannot decide whether to emit "0" or "1" and we would have to process more input digits.

The example identified the three processing steps summarized below. These steps are usually interleaved in the lazy evaluation scheme, in which output occurs as long as possible. Then input follows, but only until the next output is possible. Iteration of the whole process stops either after reaching the specified error bound (see (6)), or once the result's expansion has terminated.

TABLE 2
Transformations of Fractional Coefficients of an LFT with Respect to an Absorbed/Emitted Digit

Digit	Absorption	Decision	Emission
$-$	$\binom{-a\ b}{-c\ d}$	$\phi^{(m;n)}(\mathbb{B}) \subseteq \mathbb{B}_-$	$\binom{-a\ -b}{c\ d}$
$/$	$\binom{b\ a}{d\ c}$	$\phi^{(m;n)}(\mathbb{B}) \subseteq \mathbb{B}_/$	$\binom{c\ d}{a\ b}$
0	$\binom{a+b\ a}{c+d\ c}$	$\phi^{(m;n)}(\mathbb{B}) \subseteq \mathbb{B}_0$	$\binom{c\ d}{c-a\ d-b}$
1	$\binom{2a\ b}{2c\ d}$	$\phi^{(m;n)}(\mathbb{B}) \subseteq \mathbb{B}_1$	$\binom{a\ b}{2c\ 2d}$

- *Absorption* combines the leading, yet unprocessed digit of the argument with the actual state $\phi^{(m;n)}$ of an LFT into a new LFT, i.e.,

$$\phi^{(m;n)} \circ \beta_{b_n} = \phi^{(m;n+1)}.$$

  After processing, the digit b_n is "discarded" from the argument's representation, saying that it has been *consumed* or *absorbed* into the new LFT.
- *Decision* step estimates the range of the LFT $\phi^{(m;n)}$ to see, whether an output digit of the result may be determined. The estimation considers an implicit constraint of the domain of LFT's argument, such as $\mathbb{B}$ for continued logarithms. A potential output digit is then determined by comparing the estimated range with output conditions, i.e., checking if there is a $c \in \{0, 1, -, /\}$ such that $\phi^{(m;n)}(\mathbb{B}) \subseteq \mathbb{B}_c$.
- *Emission* outputs (or *emits*) the previously determined digit of the result and, using an inverse transformation, updates the computed LFT. That is

$$\beta_{b'_m}^{-1} \circ \phi^{(m;n)} = \phi^{(m+1;n)}.$$

Automating this process is almost straightforward. We keep the current state of the LFT $\phi^{(m;n)}$ in four integer variables a, b, c, d. Absorption and emission changes these variables accordingly to a considered digit (see Table 2). For decision, we first need to know domain $\mathbb{X}$ of the input argument. If $n \geq 2$, then $\mathbb{X} \subseteq \mathbb{B}$, and $\mathbb{X} = \{\infty\}$ if the argument was finite and has been absorbed completely; otherwise, we assume $\mathbb{X} = \mathbb{R}^*$. All that remains is to estimate $\phi^{(m;n)}(\mathbb{X})$, which makes sense only when $\phi^{(m;n)}$ has no singularity on $\mathbb{X}$, thus, being monotone and continuous. Thus, if $\mathbb{X} \subseteq \mathbb{B}$, the estimated range is delimited by values of $\phi^{(m;n)}(1) = \frac{a+b}{c+d}$ and $\phi^{(m;n)}(\infty) = \frac{a}{c}$. Once $\mathbb{X} = \{\infty\}$, the LFT collapses to a rational number $\frac{a}{c}$.

Principles of LFTs may be extended to multiple variables, thus, introducing multilinear fractional transformations, i.e., fractional transformations linear in all variables. Of particular interest here is a *bilinear fractional transformation* (BLFT), which can compute an arbitrary rational function $\phi : \mathbb{R}^* \times \mathbb{R}^* \rightarrow \mathbb{R}^*$ of two variables. It has the form of

$$\phi(x, y) = \frac{axy + by + cx + d}{exy + fy + gx + h}, \tag{7}$$

1. Some other representations may also be found to have an LFT character. For example, the conventional decimal representation of a number $x \in [0, 1]$, written as $0.d_0d_1d_2\ldots$, is equivalent to $\prod \binom{1\ d_i}{0\ 10} \cdot \binom{0}{1}$, where $d_i \in \{0, 1, \ldots, 9\}$ is a digit. Therefore, the *digit matrix*.

Input: ϕ – a BLFT given as an eight-tuple of its coefficients.
Input: x, y – arguments in CL representation.
Input: k – required length of $\phi(x, y)$'s expansion.
Output: CL expansion of $\phi(x, y)$.

```
1:  m = n = o = 0;
2:  X_n = range(x, n);
3:  Y_o = range(y, o);
4:  D_m = X_n × Y_o;
5:  b = ∅;
6:  while (φ(D_m) ⊈ {∞} && m < k) do
7:      b_m = decide(φ, D_m);
8:      if ((min(n, o) ≥ 2 || D_m ⊆ {∞}^2) && b_m ≠ ∅) then
9:          φ = emit(φ, b_m);
10:         b = b ∘ b_m;
11:         m = m + 1;
12:         continue
13:     end if
14:     S = select(x, n, y, o);
15:     if x ∈ S then
16:         φ = absorb(φ, x, n);
17:         n = n + 1;
18:     end if
19:     if y ∈ S then
20:         φ = absorb(φ, y, o);
21:         o = o + 1;
22:     end if
23:     D_m = range(x, n) × range(y, o)
24: end while
25: return b;
```

Fig. 1. Process of computing k output digits of the BLFT $\phi(x, y)$.

with coefficients $a, b, c, d, e, f, g, h \in \mathbb{Z}$ and arguments $x, y \in \mathbb{IR}^*$.

Even the multilinear transformations are closed under composition with LFTs, which means for a BLFT that its bilinear character is preserved under absorption and emission of digit matrices. Similarly to LFTs, we can see a BLFT as a two-by-four matrix, writing $\phi \cong \left(\begin{smallmatrix} a & b & c & d \\ e & f & g & h \end{smallmatrix}\right)$, but some of the linear algebra properties are lost. For example, composition with LFTs does no longer correspond to a matrix multiplication.

The principles of arithmetic remain essentially the same, but handling more input arguments adds some complexity. Fig. 1 provides a sample implementation of BLFT processing. It builds upon five core functions. `Absorb` and `emit` compute transformations of the eight fractional coefficients $a, \ldots, h$. If $\phi^{(m;n,o)}$ is the actual state of the BLFT, then consuming k digits of x and l digits of y will change the BLFT to $\phi^{(m;n+k,o+l)}$. Emitting j digits will then change $\phi^{(m;n,o)}$ to $\phi^{(m+j;n,o)}$. These transformations, for $j, k, l = 1$, are summarized in Table 3. They can be derived either directly, by composition of the BLFT with LFTs of individual digits, or using matrix operations. In the latter case, if $A = \left(\begin{smallmatrix} a & b & c & d \\ e & f & g & h \end{smallmatrix}\right)$ is a BLFT and $D = \left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)$ is a digit matrix, then output transformations are obtained by $D^* \cdot A$, and input transformations for x and y by $A \cdot X$ and $A \cdot Y$, where X and Y are derived from D:

$$X = \begin{pmatrix} p & q & 0 & 0 \\ r & s & 0 & 0 \\ 0 & 0 & p & q \\ 0 & 0 & r & s \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} p & 0 & q & 0 \\ 0 & p & 0 & q \\ r & 0 & s & 0 \\ 0 & r & 0 & s \end{pmatrix}.$$

`Select` determines, from which argument to absorb another digit. Different strategies are possible, the simplest being simultaneous absorption from both arguments or their alternation during iterations of the algorithm. `Range` function estimates the domain of an input argument and returns $\mathbb{B}$, $\{\infty\}$, or $\mathbb{IR}^*$ as described above.

The `decide` process inspects the same decision conditions as before, but this time ϕ's range evaluation is more complex as it is done over two dimensions. This also complicates detection of singularity of a computed BLFT. We say a function is *well defined* [11] if it is monotone and without singularity on the assumed domain. A sufficient condition for a BLFT ϕ to be well defined is that neither its numerator, nor its denominator changes a sign over the assumed domain. Since the largest domain worth of considering is $\mathbb{B} \times \mathbb{B}$, we define n_i, d_i, $i \in \{0, 1, 2, 3\}$ as the values of ϕ's numerator and denominator, respectively, in the four corners of the assumed domain (see Table 4). Then, the fractions $\frac{n_i}{d_i}$ delimit the range of ϕ if the following condition holds:

$$\begin{aligned} &\operatorname{sgn}(n_0) = \operatorname{sgn}(n_1) = \operatorname{sgn}(n_2) = \operatorname{sgn}(n_3), \\ &\operatorname{sgn}(d_0) = \operatorname{sgn}(d_1) = \operatorname{sgn}(d_2) = \operatorname{sgn}(d_3) \neq 0. \end{aligned} \tag{8}$$

Table 4 also shows how values n_i, d_i change when one or both input arguments are finite and get completely absorbed, restricting their associated domain to $\{\infty\}$. Fig. 2 then provides a sample implementation of the complete decision process.

3.5 Failure of Computability on Real Numbers

It is easy to demonstrate that continued logarithm arithmetic may fail when computing with real numbers. The usual argument shows a contradiction that a finite representation of a perfect square cannot be computed as an infinite product of its irrational square roots [19]. Especially nice example is $(\sqrt{2})^2$, in which case the decision procedure always fails, never emitting a single output digit.

TABLE 3
Transformations of Fractional Coefficients of (7) with Respect to an Absorbed/Emitted Digit [7]

Digit	x-absorption	y-absorption	Emission
$-$	$\left(\begin{smallmatrix} -a & b & -c & d \\ -e & f & -g & h \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} -a & -b & c & d \\ -e & -f & g & h \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} -a & -b & -c & -d \\ e & f & g & h \end{smallmatrix}\right)$
$/$	$\left(\begin{smallmatrix} b & a & d & c \\ f & e & h & g \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} c & d & a & b \\ g & h & e & f \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} e & f & g & h \\ a & b & c & d \end{smallmatrix}\right)$
0	$\left(\begin{smallmatrix} a+b & a & c+d & c \\ e+f & e & g+h & g \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} a+c & b+d & a & b \\ e+g & f+h & e & f \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} e & f & g & h \\ a-e & b-f & c-g & d-h \end{smallmatrix}\right)$
1	$\left(\begin{smallmatrix} 2a & b & 2c & d \\ 2e & f & 2g & h \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} 2a & 2b & c & d \\ 2e & 2f & g & h \end{smallmatrix}\right)$	$\left(\begin{smallmatrix} a & b & c & d \\ 2e & 2f & 2g & 2h \end{smallmatrix}\right)$

TABLE 4
Numerators n_i and Denominators d_i of a BLFT at the Corners of Its Potential Domains

Variable	$\mathbb{B} \times \mathbb{B}$	$\mathbb{B} \times \{\infty\}$	$\{\infty\} \times \mathbb{B}$	$\{\infty\} \times \{\infty\}$
n_0	a	a	a	a
n_1	$a+b$	$a+b$	a	a
n_2	$a+c$	a	$a+c$	a
n_3	$a+b+c+d$	$a+b$	$a+c$	a
d_0	e	e	e	e
d_1	$e+f$	$e+f$	e	e
d_2	$e+g$	e	$e+g$	e
d_3	$e+f+g+h$	$e+f$	$e+g$	e

Input: ϕ – a BLFT given as an eight-tuple of its coefficients.
Input: $\mathbb{D}$ – a domain in $\mathbb{R}^* \times \mathbb{R}^*$ over which to consider ϕ.
Output: A digit potential for emission, if it exists, or $\emptyset$ otherwise.

```
c = ∅;
for i=0 to 3 do
    n_i = numerator(i, 𝔻);          // see Table 4
    d_i = denominator(i, 𝔻);        // see Table 4
end for
if (φ is well-defined on 𝔻) then
    // see (8)
    if (sgn(n_0) ≠ sgn(d_0)) then
        c = −;                      // φ(𝔻) ⊆ [∞, 0)
    else if (n_i < d_i for ∀i) then
        c = /;                      // φ(𝔻) ⊆ [0, 1)
    else if (d_i ≤ n_i < 2d_i for ∀i) then
        c = 0;                      // φ(𝔻) ⊆ [1, 2)
    else if (2d_i ≤ n_i for ∀i) then
        c = 1;                      // φ(𝔻) ⊆ [2, ∞]
    end if
end if
return c;
```

Fig. 2. Decision process `decide(`ϕ`,`$\mathbb{D}$`)`.

Such failure of computability is the result of combining uniqueness and discrete character of the representation. Fig. 3a shows that the extended binary CL representation is "discretized" in three points ($x = 0, 1, 2$), dividing its domain into four separate intervals of different character defined by (5). Each of those three points represents a potential place of undecidability as we "oscillate" around its border. By oscillating, we understand that observed quantities n_i and d_i change with absorption so that all fractions $\frac{n_i}{d_i}$ alternate in a proximity of the undecidability border, but all of them can never satisfy the same output condition (see Table 5 for illustration). Resolving this undecidability is possible by letting the neighboring definition intervals overlap and, thus, introduce some redundancy that smoothes their strict borders (see Fig. 3). Details are discussed in the following section.

4 Speculative Redundancy

Addressing the uncomputability by making the representation redundant is a common approach [4], [9], [11]. However, such redundancy must be suitable in the sense of meeting the *interior containment property* [5]. This property guarantees that having a stream-like representation of a numerical quantity (such as CL expansion), a sequence of its finite prefixes forms a sequence of open, contracting intervals containing that quantity. The intervals contract so that any extension of a representation prefix forms a strict subset of its associated interval, thus, ensuring convergence of such intervals. Intuitive understanding of this property is that once we have a prefix approximating the number, its extension can only improve this approximation. We will show that the speculative concept developed in this section is consistent with this natural understanding.

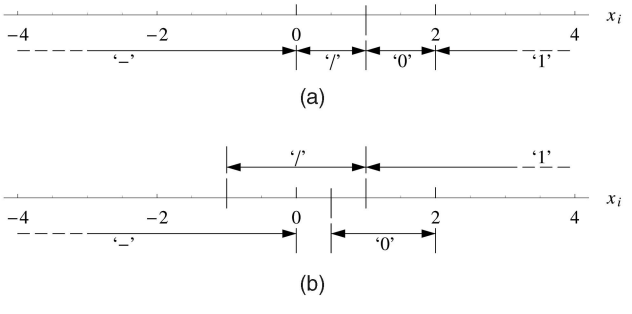


Fig. 3. Redundancy introduced by interval overlaps. (a) Nonredundant representation. (b) Redundant representation.

TABLE 5
Snapshot of Observed Parameters for $\phi = xy = \sqrt{2} \cdot \sqrt{2}$

Parameter	$\phi^{(0;0,0)}$	$\phi^{(0;5,5)}$	$\phi^{(0;20,20)}$	$\phi^{(0;40,40)}$
n_0/d_0	1.0	2.25	1.9999649	2.000000000153
n_1/d_1	2.0	2.10	1.9999855	2.0
n_2/d_2	2.0	2.10	1.9999855	2.0
n_3/d_3	4.0	1.96	2.0000060	1.999999999847
a	1	36	228 484	$130\,515 \cdot 10^5$
e	1	16	114 244	$65\,257 \cdot 10^5$

n_i/d_i are ϕ's values at the corners of $\mathbb{B} \times \mathbb{B}$, a, e are coefficients of ϕ.

Speculative redundancy follows Gosper's idea of eager emission of premature output [2] and it can be applied in situations where the range of a computed function ϕ is bounded in close neighborhood of some numerical quantity, e.g., $\widetilde{\phi}$. By speculating $\phi = \widetilde{\phi}$, we may proceed in computation without actually knowing if it is true. Such behavior is of course safe only if there is a correction mechanism in cases where the speculation missed. It is the purpose of this section to show that such approach can be easily applied to continued logarithms, because the correction mechanism naturally follows from their LFT character.

4.1 Case Study

Example 4. Let us start with an example of $\phi(x, y) = xy$, where $x = y = \sqrt{2}$. Using Fig. 1, we got results reported in Table 5. We can see that the range of ϕ (bounded by values of $\frac{n_i}{d_i}$) spreads over a shrinking neighborhood of two, which is actually one of the three decision borders. Supported by the argument from Section 3.5, this neighborhood does not seem to ever find itself completely above or below this border. And since quantities below and above this border have different representation prefixes, we cannot resolve the representation of ϕ.

As ϕ could be, however, well approximated by two, let us speculatively emit a prefix of $2 = (10)_{\mathrm{CL}}$. The emission changes ϕ into $\phi^{(2;n,o)}$, where n, o as usual denote the number of absorbed digits from x and y, respectively. Table 6 shows how range of this transformed function

TABLE 6
Snapshot of Observed Parameters for $\phi = xy = \sqrt{2} \cdot \sqrt{2}$
After Speculative Emission of Prefix $(10)_{\mathrm{CL}}$

Parameter	$\phi^{(2;5,5)}$	$\phi^{(2;10,10)}$	$\phi^{(2;20,20)}$	$\phi^{(2;40,40)}$
n_0/d_0	-50.0	288.0	-57 122	$130\,515 \cdot 10^5$
n_1/d_1	90.0	∞	-137 904	∞
n_2/d_2	90.0	∞	-137 904	∞
n_3/d_3	23.14	-289.0	332 928	$-130\,515 \cdot 10^5$
a	50	288	228 488	$130\,515 \cdot 10^5$
e	-1	1	-4	1

changes with absorption. All the ranges have a form of a *split interval* $[\infty,-p]\cup[q,\infty]$, where $p,q>0$ and their value increases with the number of absorptions. In this form, the range does not meet any output condition and it seems that no further emission is possible. However, it turns out that the fractional coefficients e,f,g,h repeatedly take on small integer values forming a recurrent pattern, while a,b,c,d constantly increase. This leads us to a belief that the borders p, q of the split interval will approach ∞ as long as $x=y=\sqrt{2}$.

Since ∞ can be represented by an infinite stream of "1" digits, we could potentially extend our speculation and emit "1"s as long as $|\phi^{(m;n,o)}|\geq 2$. With this assumption, we get an infinite expansion of $(1011\ldots1\ldots)_{\mathrm{CL}}$, which correctly represents the value of two.

This is though not as good as a finite expansion of two, but we were at least able to emit a result that approximates this value. In fact, the approximation improves with the number of trailing "1"s emitted. For this example, the approximation error is $|\frac{p_n}{q_n}-2|\leq 2^{3-n}$, where $\frac{p_n}{q_n}$ is the nth order convergent. We may, thus, at least control the precision of the result.

Using ideas of Example 4, we may generally need to apply speculation at any of the three decision borders 0, 1, and 2. The individual cases are covered by observations below and, as shown, they all necessarily introduce a potential singularity (in the point of speculation), leading to a split interval; the speculation may though go on, if care is taken. In the following, we use $\pm\epsilon$, $\epsilon>0$, to denote a number in an interval $[-\epsilon,+\epsilon]$.

Observation 1 ($\phi^{(m;n,o)}\approx 2$). *Suppose that we are computing a BLFT $\phi^{(m;n,o)}$, value of which is bounded in an ϵ-neighborhood of two. Speculating that its value is actually two, we emit "1" getting a new BLFT $\phi^{(m+1;n,o)}=\beta_1^{-1}\circ\phi^{(m;n,o)}=\phi^{(m;n,o)}/2$ so that, if $\phi^{(m;n,o)}=2\pm\epsilon$, then $\phi^{(m+1;n,o)}=1\pm\frac{\epsilon}{2}=1\pm\epsilon'$ and its range is bounded in an ϵ'-neighborhood of 1. That is, further development of the speculation is covered by Observation 2.*

Observation 2 ($\phi^{(m;n,o)}\approx 1$). *Suppose that we are computing a BLFT $\phi^{(m;n,o)}$, value of which is bounded in an ϵ-neighborhood of one. Speculating that its value is actually one, we emit "0" getting a new BLFT $\phi^{(m+1;n,o)}=\beta_0^{-1}\circ\phi^{(m;n,o)}=1/(\phi^{(m;n,o)}-1)$ so that, if $\phi^{(m;n,o)}=1\pm\epsilon$, then $\phi^{(m+1;n,o)}=1/(1\pm\epsilon-1)=\frac{1}{\pm\epsilon}\subseteq[\infty,-\frac{1}{\epsilon}]\cup[\frac{1}{\epsilon},\infty]$. Range of the new BLFT becomes a split interval and any further development of the speculation is covered by Observation 4.*

Observation 3 ($\phi^{(m;n,o)}\approx 0$). *Suppose that we are computing a BLFT $\phi^{(m;n,o)}$, value of which is bounded in an ϵ-neighborhood of zero. Speculating that its value is actually zero, we emit "/" getting a new BLFT $\phi^{(m+1;n,o)}=\beta_/^{-1}\circ\phi^{(m;n,o)}=1/\phi^{(m;n,o)}$ so that, if $\phi^{(m;n,o)}=\pm\epsilon$, then $\phi^{(m+1;n,o)}=1/\pm\epsilon\subseteq[\infty,-\frac{1}{\epsilon}]\cup[\frac{1}{\epsilon},\infty]$. Range of the new BLFT becomes a split interval and any further development of the speculation is covered by Observation 4.*

Lemma 1. *The LFT $\beta_-\cong\left(\begin{smallmatrix}-1&0\\0&1\end{smallmatrix}\right)$ associated with a CL digit "$-$" commutes under operation of composition with the LFTs $\beta_1\cong\left(\begin{smallmatrix}2&0\\0&1\end{smallmatrix}\right)$ and $\beta_/\cong\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$ associated with digits "1" and "/," respectively. That is, $\beta_-\circ\beta_1=\beta_1\circ\beta_-$ and $\beta_-\circ\beta_/=\beta_/\circ\beta_-$.*

Corollary 1. *If $B_-\cong\beta_-$ and $B_1\cong\beta_1$, then for all n, $0<n\leq m$, it is*

$$B_-\cdot\prod_{i=0}^{m}B_1=\prod_{i=0}^{n-1}B_1\cdot B_-\cdot\prod_{i=n}^{m}B_1=\prod_{i=0}^{m}B_1\cdot B_-.$$

In other words, there is no difference among expansions of the same length and form of $(-11\ldots1)_{\mathrm{CL}}$, $(11\ldots1-1\ldots1)_{\mathrm{CL}}$, or $(11\ldots1-)_{\mathrm{CL}}$, since they all represent the same composed transformation.

Observation 4 (Split Interval). *Suppose that we are computing a BLFT $\phi^{(m;n,o)}$ that has a singularity on an assumed domain $\mathbb{D}\subseteq\mathbb{B}\times\mathbb{B}$ of its arguments. Due to Corollary 1, it is then safe, ignoring the well-definedness condition, to emit "1"s as long as there is a $p\geq 2$ such that $\phi^{(m;n,o)}(\mathbb{D})\subseteq[\infty,-p]\cup[p,\infty]$. Otherwise, we need to absorb more input digits (say k from x, l from y) until meeting any of these conditions:*

- *$\exists p\geq 2$ such that $\phi^{(m;n+k,o+l)}(\mathbb{D})\subseteq[\infty,-p]\cup[p,\infty]$,*
- *$\exists q\geq 0$ such that $\phi^{(m;n+k,o+l)}(\mathbb{D})\subseteq[q,\infty]$ or $\phi^{(m;n+k,o+l)}(\mathbb{D})\subseteq[\infty,-q)$.*

In the latter case, where the split range restricts to either positive or negative values, $\phi^{(m;n+k,o+l)}$ becomes well defined again, and we say that the speculation got positively *or* negatively *resolved.*

Example 5. We applied speculation again on $\phi(x,y)=xy$, but this time for arguments:

1. $x=y=\sqrt{2}+\frac{1}{16}=(0101111010\ldots)_{\mathrm{CL}}$ and
2. $x=y=\sqrt{2}-\frac{1}{16}=(010101101\ldots)_{\mathrm{CL}}$.

We obtained $(101110\ldots)_{\mathrm{CL}}$ and $(10111{-}0\ldots)_{\mathrm{CL}}$, respectively, where in both cases first five digits were emitted speculatively. In case 1, the arguments are greater than the speculated value of two and the output came equal to as if it were computed nonspeculatively. The case 2 is opposite and the real result is lower than that of speculated two. Hence, "$-$" digit occurred when the speculation got resolved negatively, correcting the speculative result. We say that a *misspeculation* occurred.

4.2 Formal Definition

The previous section gave a foundation for formalizing the speculatively redundant continued logarithms. Unlike in previous definitions, Definition 5 uses the notation of intervals as our primary intention was to handle interval ranges of real functions being computed. In fact, using redundant representation for single points makes no sense.

Definition 5. Speculatively redundant continued logarithm representation *of an interval $\mathbb{X}\subseteq\mathbb{R}^*$ is a sequence of digits $\{b_i\}$, $b_i\in\{0,1,/,-,\underline{1},\underline{1},\underline{0},\underline{/}\}$, $i=0,1,\ldots$, such that there is a corresponding sequence $\{\mathbb{X}_i\}$, $\mathbb{X}_i\subseteq\mathbb{R}^*$ and $\mathbb{X}_0=\mathbb{X}$, satisfying*

$$\mathbb{X}_{i+1}=\beta_{b_i}^{-1}(\mathbb{X}_i),$$

TABLE 7
Summary of Definition 5

c	$\mathbb{B}_c$	β_c	$\beta_c^{-1}(\mathbb{B}_c)$
1	$[2,\infty]$	$\left(\begin{smallmatrix}2&0\\0&1\end{smallmatrix}\right)$	$[1,\infty]$
0	$[1,2)$	$\left(\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right)$	$(1,\infty]$
/	$[0,1)$	$\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$	$(1,\infty]$
−	$(\infty,0)$	$\left(\begin{smallmatrix}-1&0\\0&1\end{smallmatrix}\right)$	$(0,\infty)$
$\underline{1}$	$(1,4)$	$\left(\begin{smallmatrix}2&0\\0&1\end{smallmatrix}\right)$	$(\frac{1}{2},2)$
$\underline{0}$	$(\frac{1}{2},2)$	$\left(\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right)$	$[\infty,-2)\cup(1,\infty]$
$\underline{/}$	$(-1,1)$	$\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$	$[\infty,-1)\cup(1,\infty]$
$\underline{\underline{1}}$	$[\infty,-2)\cup(2,\infty]$	$\left(\begin{smallmatrix}2&0\\0&1\end{smallmatrix}\right)$	$[\infty,-1)\cup(1,\infty]$

where

$$b_i = \begin{cases} \text{nonspecul.} \begin{cases} 1, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_1 = [2,\infty], \\ 0, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_0 = [1,2), \\ /, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_/ = [0,1), \\ -, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_- = (\infty,0), \end{cases} \\ \text{spec.} \begin{cases} \underline{1}, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_{\underline{1}} = (1,4), \\ \underline{0}, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_{\underline{0}} = (\frac{1}{2},2), \\ \underline{/}, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_{\underline{/}} = (-1,1), \\ \underline{\underline{1}}, & \text{for } \mathbb{X}_i \subseteq \mathbb{B}_{\underline{\underline{1}}} = [\infty,-2)\cup(2,\infty], \end{cases} \end{cases} \tag{9}$$

and

$$\beta_1 = \beta_{\underline{1}} = \beta_{\underline{\underline{1}}} \cong \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \quad \beta_0 = \beta_{\underline{0}} \cong \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix},$$
$$\beta_/ = \beta_{\underline{/}} \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \beta_- \cong \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The sequence $\{\mathbb{X}_i\}$ terminates for $i = n$, if $\mathbb{X}_n$ does not fall within any of the decision intervals*, in which case b_n is left undetermined and the representation is* incomplete*. The sequence may also terminate once $\mathbb{X}_n = \{\infty\}$.*

Because of its complexity, we summarized the definition in Table 7 that lists the individual digits c and conditions $\mathbb{B}_c$, under which they are emitted, and corresponding LFTs β_c. The column $\beta_c^{-1}(\mathbb{B}_c)$ shows how the decision interval $\mathbb{B}_c$ changes with emission of the corresponding digit.

Example 6. Use of the above definition is as straightforward as it was using those for CL or CCL, with an exception that we transform intervals instead of single points. Indeed, a single point x may now be considered as a degenerate interval $[x,x]$.

Let us find a representation of $(\frac{4}{3},\frac{5}{3}]$. We get the following sequence of steps:

$$\begin{aligned}
&\mathbb{X}_0 = \left(\frac{4}{3},\frac{5}{3}\right] \subseteq \mathbb{B}_0 && \to b_0 = 0,\ \mathbb{X}_1 = \frac{1}{\mathbb{X}_0 - 1},\\
&\mathbb{X}_1 = \left[\frac{3}{2},3\right) \subseteq \mathbb{B}_{\underline{1}} && \to b_1 = \underline{1},\ \mathbb{X}_2 = \mathbb{X}_1/2,\\
&\mathbb{X}_2 = \left[\frac{3}{4},\frac{3}{2}\right) \subseteq \mathbb{B}_{\underline{0}} && \to b_2 = \underline{0},\ \mathbb{X}_3 = \frac{1}{\mathbb{X}_2 - 1},\\
&\mathbb{X}_3 = [\infty,-4]\cup(2,\infty] \subseteq \mathbb{B}_{\underline{\underline{1}}} && \to b_3 = \underline{\underline{1}},\ \mathbb{X}_4 = \mathbb{X}_3/2,\\
&\mathbb{X}_4 = [\infty,-2]\cup(1,\infty].
\end{aligned}$$

Input: ϕ – a BLFT given as an eight-tuple of its coefficients.
Input: $\mathbb{D}$ – a domain in $\mathbb{R}^* \times \mathbb{R}^*$ over which to consider ϕ.
Input: *singularity* – a boolean flag denoting a singularity of ϕ on domain $\mathbb{D}$ introduced by previous emission of $\underline{0}$ or $\underline{/}$ digits.
Output: A digit potential for emission, if it exists, or $\emptyset$ otherwise.

```
c = ∅;
for i=0 to 3 do
    n_i = numerator(i, 𝔻);                    // see Table 4
    d_i = denominator(i, 𝔻);                  // see Table 4
end for
if (φ(x,y) is well-defined on 𝔻) then
    if (sgn(n_0) ≠ sgn(d_0)) then
        c = −;                                // φ(𝔻) ⊆ [∞,0)
    else if (n_i < d_i for ∀i) then
        c = /;                                // φ(𝔻) ⊆ [0,1)
    else if (d_i ≤ n_i < 2d_i for ∀i) then
        c = 0;                                // φ(𝔻) ⊆ [1,2)
    else if (2d_i ≤ n_i for ∀i) then
        c = 1;                                // φ(𝔻) ⊆ [2,∞]
    else if (d_i < n_i < 4d_i for ∀i) then
        c = 1;                                // φ(𝔻) ⊆ (1,4)
    else if (½d_i < n_i < 2d_i for ∀i) then
        c = 0;                                // φ(𝔻) ⊆ (½,2)
    else if (|n_i| < |d_i| for ∀i) then
        c = /;                                // φ(𝔻) ⊆ (−1,1)
    end if
else if (singularity && (2|d_i| < |n_i| for ∀i)) then
    c = 1;                                    // φ(𝔻) ⊆ [∞,−2)∪(2,∞]
end if
return c;
```

Fig. 4. Speculative version of `decide(`ϕ`,`$\mathbb{D}$`)` function.

Since $\mathbb{X}_4$ does not fall within any decision interval, we got an incomplete prefix $(0\underline{1}01)_{\mathrm{CL}}$ of $(\frac{4}{3},\frac{5}{3}]$.

Let us now take an inverse process and find the largest interval having the obtained expansion as its prefix. We, thus, replace all but the last digit with a corresponding LFT, and substitute the last digit with its associated interval. That is,

$$\beta_0 \circ \beta_{\underline{1}} \circ \beta_{\underline{0}}(\mathbb{B}_{\underline{\underline{1}}}) \cong \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \mathbb{B}_{\underline{\underline{1}}} \\ 0 \end{pmatrix}$$
$$\subseteq \left(\frac{4}{3},2\right).$$

As the intervals used in conditions of (9) overlap, Definition 5 is implicitly ambiguous, allowing several representations of a single numerical quantity to exist. For example, $x = -7$ has expansions

$$(-11001010)_{\mathrm{CL}}, \quad (-11\underline{\underline{1011}}{-}10)_{\mathrm{CL}}, \quad (-110\underline{\underline{101}}{-}010)_{\mathrm{CL}},$$
$$(-110\underline{\underline{101}}{-}\underline{\underline{101}}{-}10)_{\mathrm{CL}}, \quad (\underline{\underline{1}}{-}100\underline{\underline{1}}010)_{\mathrm{CL}},$$

etc. From algorithmic perspective, however, we would like to have a deterministic algorithm that can find at least one of those. An intuitive guideline on using the overlapping cases from speculative and nonspeculative cases of (9) follows the concept of speculation, which is used to handle exceptional situations. It means that a nonspeculative case is default and the decision algorithm should examine its possibility first; only if none of the nonspeculative cases applies, the decision algorithm would attempt the speculation.

This principle was algorithmized in Fig. 4 as a replacement of Fig. 2. The new algorithm also explicitly declares the use of case for "$\underline{\underline{1}}$." While for other digits we require a BLFT ϕ to be well defined, we use "$\underline{\underline{1}}$" when there is a singularity

introduced by a previously started speculation, indicated by the *singularity* flag. As seen from column $\beta_c^{-1}(\mathbb{B}_c)$ in Table 7, only emission of "$\underline{0}$" or "$\underline{/}$" can introduce singularity in a well defined BLFT, setting the *singularity* flag. The singularity will be preserved by emissions of "$\underline{\underline{1}}$," until absorption of some input digits will transform the computed BLFT into a well defined function again and will clear the *singularity* flag. As mentioned, the flag denotes a singularity introduced on behalf of speculation, and as such, it distinguishes these cases from other not well defined BLFTs, thus, preventing inappropriate use of "$\underline{\underline{1}}$."

4.3 Normal versus Speculative Digits

Meaning of digits "1," "0," "−," and "/," whether emitted speculatively or nonspeculatively, is not affected by speculation. That is, "1" still indicates that a next continuation is half of the previous, "/" denotes a multiplicative inverse, etc. However, to indicate a speculatively emitted digit, we introduced in (9) *underline notation*. It is, thus, easy to comprehend the difference between prefixes $(101)_{\mathrm{CL}}$ and $(\underline{101})_{\mathrm{CL}}$ directly from their visual aspect. The former, nonspeculative prefix represents an interval of $\beta_1 \circ \beta_0(\mathbb{B}_1) = [2,3]$, while the speculative prefix corresponds to a value within a certain tolerance of two, namely $\beta_{\underline{1}} \circ \beta_{\underline{0}}(\mathbb{B}_{\underline{\underline{1}}}) = (1,3)$. That is, encountering start of underline in an expansion indicates the start of speculation, while detecting the end of underline signals that the speculation has been resolved, either positively or negatively. Thus, e.g., $(\underline{101})_{\mathrm{CL}}$ is a prefix of $[2,3]$, while $(\underline{10}{-}1)_{\mathrm{CL}}$ denotes interval $[1,2]$.

A special note regards the use of *double underline*. Consider the following prefixes: $(\underline{101})_{\mathrm{CL}}$ and $(\underline{101})_{\mathrm{CL}}$. Interpreting them would yield intervals of $(2.5,4)$ and $(1,3)$, respectively. This clearly indicates that there should be a difference in interpreting single and double underlines. The case of $(\underline{101})_{\mathrm{CL}}$ should really be viewed as $(\underline{10}\ \underline{1})_{\mathrm{CL}}$, where the underline interruption delimits two speculations—the first $(\underline{10})_{\mathrm{CL}}$ has been positively resolved, while the second $(\underline{1})_{\mathrm{CL}}$ is potentially still pending. The case of $(\underline{101})_{\mathrm{CL}}$, on the other hand, represents a single and potentially still pending speculation as in Example 4—digits "$\underline{\underline{1}}$" should be, thus, seen as the trailing "1"s signaling the distance from the speculated value; the more of them, the closer to the speculated value. Therefore, encountering change from single to double underline, as in $(\ldots \underline{01} \ldots)_{\mathrm{CL}}$, should be understood as "going on" in speculation, while changing from double to single underline, as in $(\ldots \underline{\underline{11}} \ldots)_{\mathrm{CL}}$, means positive resolution of one speculation and immediate start of a new speculation.

Knowing how to "visually" interpret a stream of redundant digits, we may revise their interpretation from an algorithmic perspective. This is indeed easy as the set of associated transformation has not changed (see Table 7), and so has not the set of absorption and emission transformations given in Table 3. Therefore, almost no change is necessary in Fig. 1, at least with respect to `absorb` and `emit` procedures. This may cause some confusion as we on one hand emphasize importance of the underline notation, and on the other hand say that we may collapse the total of eight redundant digits into a set of four digit matrices. Details are given in Section 4.4, but such "collapsing" is possible (and correct) only if we keep track of speculation on input arguments.

We also remark on encoding of redundant digits. As there are eight of them, the straightforward encoding would require three bits per digit. The storage requirements would, thus, be one and half times of those for nonredundant continued logarithms. Alternatively, one may consider using some "start" and "stop" symbols to signal entering and leaving the speculative mode and interpret the four basic symbols accordingly. Six symbols does not fit within two bits either, but as some of them will be used less frequently, considering run-length encoding might help decrease the additional storage overhead.

4.4 Arithmetic Algorithms

The nice property of the speculatively redundant extension is the fact, that it actually affects only the decision process. Because the set of transformations associated with the representation digits has remained the same, nothing changes in absorption and emission transformations given in Table 3. It would, thus, seem that Fig. 1 could be applied without a change, except of using new `decide` procedure of Fig. 4. This is true only partially; due to the self-correcting mechanism[2] of the representation, the naive use of Fig. 1 will always produce a correct result at the end. However, the intermediate results may possibly exhibit unwanted artifacts such as correction or retraction, resulting from misspeculation in some of the input arguments. The effects of such input misspeculation on the output of a computed BLFT $\phi(x,y)$ are as follows:

- *No change:* The misspeculation does not anyhow affect the outgoing stream. This happens if ϕ is flat within the interval of speculation of the considered argument.
- *Correction:* This situation occurs if ϕ changes in the interval of speculation more rapidly. The correction then introduces itself like a redundant representation by issuing "−," which is however not preceded by any speculative output digits.
- *Retraction:* The correction may eventually promote to retraction of the generated output. It appears as a sequence of inverse transformations (usually identified by occurrence of "/" symbol), which withdraw one, few, or eventually all symbols output so far; e.g., $(111/110\ldots)_{\mathrm{CL}}$, where the two "1"s after "/" withdraw the same number of preceding "1"s. Retraction typically occurs as a result of singularity within or close to the interval of speculation of the input argument, making ϕ especially sensitive to even small changes of its input.

Illustrative examples of undesired behavior are in Fig. 5, displaying rational functions computed for $x = 1.8 = (\underline{\underline{10111}}{-}0110)_{\mathrm{CL}}$. The plots show that character of the functions changes around the speculated value of two.

2. This character is a result of continued logarithms' reversibility. For any LFT associated with a CL digit, there is an inverse LFT composed of other CL digits. For example, $\beta_1^{-1} = \beta_/ \circ \beta_1 \circ \beta_/$ and $\beta_0^{-1} = \beta_- \circ \beta_/ \circ \beta_0 \circ \beta_/ \circ \beta_-$. Thus, any sequence of digits emitted can be "taken back" by a suitable reverse sequence.

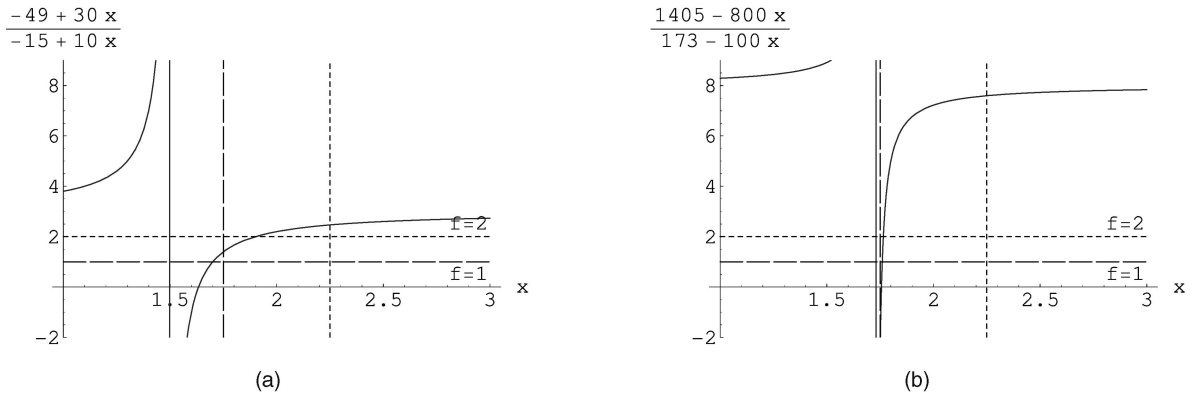


Fig. 5. Examples, where the naive use of speculative approach produces undesired results due to misspeculation within input expansion of x. The shown representation of functional values were computed for the input $x = (\underline{10\overline{111}}{-}0110)_{\mathrm{CL}}$. The vertical lines delimit the interval of speculation corresponding to $(\underline{10\overline{111}})_{\mathrm{CL}}$, i.e., the speculative prefix of x. The horizontal lines represent the decision borders for emission. (a) Correction, $\phi_1(x) = (1011{-}010)_{\mathrm{CL}}$. (b) Retraction, $\phi_2(x) = (110011/111010)_{\mathrm{CL}}$.

The principal cause of this behavior is the procedure `range`, which we said would return $\mathbb{B}$ as the largest domain for an argument with enough digits absorbed, or $\{\infty\}$ for a completely absorbed argument. Let us agree on notation $-\mathbb{B} = [\infty, -1]$, $+\mathbb{B} = [1, \infty]$, and $\pm\mathbb{B} = -\mathbb{B} \cup +\mathbb{B}$. Table 7 then shows that emission of a speculative symbol potentially introduces a domain $\pm\mathbb{B} \not\subseteq \mathbb{B}$. In other words, when speculation is detected on an input, `range`, as considered so far, does not take into account a potential misspeculation, which, when resolved negatively, will restrict the domain $\pm\mathbb{B}$ into $-\mathbb{B}$. Hence, any output emitted in the meantime might be incorrect, requiring a correction or retraction.

Adapting the `range` procedure accordingly will address the issue. All we need is to keep track of speculation on input arguments[3]—when started, `range` will return $\pm\mathbb{B}$, and when resolved, $\mathbb{B}$ will be returned again. For a completely absorbed argument, `range` will return $\{\infty\}$ as usual. Beware of "$\underline{1}$" emission, after which $\beta_{\underline{1}}^{-1}(\mathbb{B}_{\underline{1}}) = (\frac{1}{2}, 2) \not\subseteq \pm\mathbb{B}$. As "$\underline{0}$" will immediately follow "$\underline{1}$," handling this temporary special case is easy.

Changing the domain will also affect the decision algorithm, or more precisely the range evaluation of a computed BLFT. So far, the range evaluation has been done using the values in the four corners of $\mathbb{B} \times \mathbb{B}$. Now, with the largest possible domain of $\pm\mathbb{B} \times \pm\mathbb{B}$, we got more than twice the number of corners. We may, thus, either evaluate the value of a computed BLFT in all those corners, or use an alternative approach we are about to describe. Advantage of the latter is in using the original four fractions $\frac{n_i}{d_i}$ to estimate the range of BLFT.

When misspeculation occurs, a digit "$-$" will occur indicating a negative continuation. Thus, when dealing with a speculation on input argument, we should consider that the speculation will either succeed or fail. We may simulate both possibilities to see, whether they lead to the same decision—if they do, it is safe to emit an output digit. The simulation utilizes Fig. 4 that evaluates the range of a BLFT in four corners. The algorithm is first executed normally, simulating that the speculation succeeds. Then, we simulate a change of sign of the speculated argument—the sign change is canceled after simulation—and if the decision outcome is consistent with the previous one, emission may proceed. One must consider that if speculation occurs on both inputs at the same time, the decision algorithm has to be rerun four times to cover all possible combinations of signs of the two arguments. It is also important to understand that this approach does not anyhow affect the lazy nature of the algorithm.

3. Here, it becomes clear that the four digits matrices do not convey all the information and an additional signal "speculation in progress" is necessary.

4.5 Correctness

We saw in previous section that computing a function of arguments in a redundant representation may lead to retraction. We also explained that this result was due to misinterpretation of the redundant representation. One may, however, wonder, whether, even if inputs are interpreted correctly, a speculative emission based on (9) cannot produce a retracting output.

In fact, the emission cannot lead to retraction because of meeting interior containment property [5]. This claim is relatively easy to prove with canonical form of redundant representation [19], and it can be generalized to the binary form of Definition 5 (see online appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2010.110 [12]).

Meeting the interior containment has several implications. First of all, a speculatively produced output preserves the incremental character and converges to a single real number. This holds even for expansions composed of only speculatively produced digits such as $(\underline{10\overline{11\ldots}})_{\mathrm{CL}} = 2$, $(\underline{0\overline{11\ldots}})_{\mathrm{CL}} = 1$, etc. Secondly, arithmetic algorithms using speculation and operating on speculatively produced arguments will compute next output digit in a finite time. This is because absorbing more input digits will contract the input domain so that the range of a computed BLFT will contract as well, eventually meeting some output condition. Furthermore, thanks to overlaps of decision intervals, undecidability cannot occur any more. Therefore, speculative algorithms cannot get stuck like the nonredundant ones, even if composed to compute more complicated functions, and they will always be able to produce results with the desired precision. Unfortunately, there is no simple

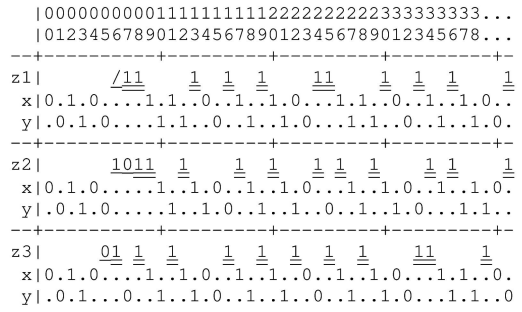


Fig. 6. Example of computing with irrational numbers using speculative redundancy: $z_1 = x - y$, $z_2 = xy$, $z_3 = x/y$, $x = y = \sqrt{2}$.

metric to predict the number of steps (absorptions and emissions) required; some results were obtained in [16], but these unlikely apply to our case (see Section 2).

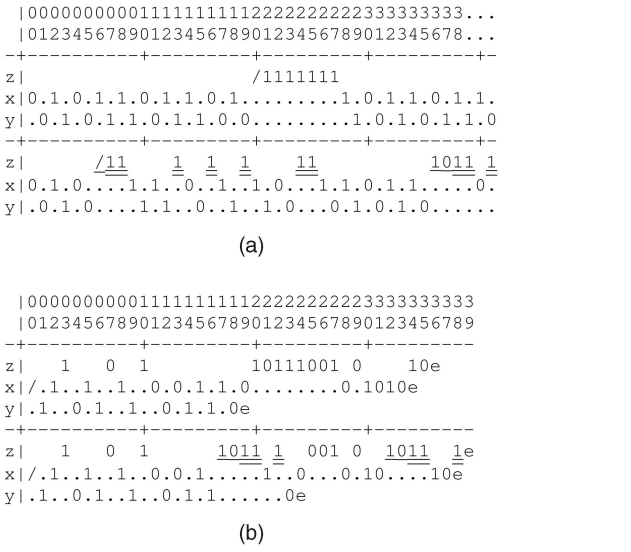
5 Practical Effects of Redundancy

This section discusses some advantages and drawbacks that the redundant representation (Definition 5) offers over the nonredundant one (Definition 3). The discussion is somewhat general and we support it with few examples and measured figures. A thorough, quantitative comparison of the two representations, as well as a comparison with related work, is planned for a separate paper. Meanwhile, readers can refer to [13] and [22] for some preliminary results. Here, we focus on the following aspects:

- computability,
- rate of computing output elements,
- effect on parameters such as expansion length, and
- performance of arithmetic algorithms.

Introducing the speculative redundancy to address the real number computability was our principal objective. In Fig. 6, we provide few examples of computing functions, for which nonredundant continued logarithms cannot resolve even a single bit. The simulation footprints were constructed with the notation taken from [11]. Each line illustrates the process of computing individual symbols of each argument or function. A horizontal position of a symbol represents a time step at which the symbol has been consumed by its parent function (or just emitted for a top-level function). There is also a time axis above the simulation outputs; e.g., at step 6 the leading symbol of the product z_2 was computed. The periods indicate that an expansion symbol is available, but has not been consumed yet, such as for argument y at step 0. The absorption process was chosen so that symbols from both operands are consumed in a strictly alternating fashion. This is certainly not the best absorption scheme, but it is one of the simplest. Different schemes were examined in [15], but their application to continued logarithms may be limited by differences in representation used therein. Nonetheless, finding a scheme that will be (near) optimal and simple enough still seems as an open problem.

Another positive effect the proposed redundancy offers is the increased output responsiveness. It means that the redundant output of a computed function is usually generated faster than a nonredundant one. Fig. 7 shows two examples; e.g., for $z_4 = \sqrt{2} - (\sqrt{2} - 2^{-8})$ the leading



(a)

(b)

Fig. 7. Comparison of nonredundant and redundant representations in terms of responsiveness and expansion length. Bottom parts of simulation outputs are for redundant representation. Symbol "e" denotes end of expansion. (a) $z_4 = x - y$, $x = \sqrt{2}$, $y = \sqrt{2} - 2^{-8}$. (b) $z_5 = x + y$, $x = 1/15$, $y = 24/10$.

symbol of z was generated in cycle 6 for redundant and in cycle 20 for nonredundant representation. This difference is usually larger for harder problems; e.g., more than 100 cycles for Rump's polynomial [13]. Better responsiveness is an important property for parallelization of continued fraction arithmetic. The sooner one arithmetic unit produces its output, the sooner another unit may start to process it and produce its own output. As retraction cannot occur on output of any unit, the complete computation process will preserve its laziness and incremental character. While the parallel execution might considerably shorten the overall computation time, it would first need to address many unresolved issues, as we discuss in Section 6.

Increased responsiveness also leads to lower demands on size of fractional coefficients $a, \ldots, h$ of a computed BLFT. This happens in situations when the value of a computed function oscillates long time around a decision border. This is the case of $z_4(x, y) = x - y$ in Fig. 7, where the range of z_4 covers an ϵ-neighborhood of value 0. The nonredundant representation has to process more symbols from inputs to see, on which side of 0 the actual value lies. The redundant representation may produce "/" speculatively spending two bits on the largest of coefficients $a, \ldots, h$, while the nonredundant algorithm needs nine bits.

On the other hand, redundant emission may be too rash, thus, resulting in longer expansions. In [13], experiments were performed computing BLFTs with randomly initialized coefficients $a, \ldots, h$ and with rational arguments from interval $[0, 1]$. The average length of the result was 299 digits with speculative algorithms, while 274 digits with nonspeculative algorithms. The length of expansions may negatively affect performance as the computation time increases with the number of produced or absorbed digits—in fact, expansion length is very important parameter as demonstrated on the example of Stern-Brocot representation (Section 2). However, in case of redundant continued logarithms a part of that length is often spent on

trailing "1"s (e.g., z_5, Fig. 7). If we do not insist on computing exact values, we may not need to process all these trailing symbols as they carry only additional information on error of their prefix (see Section 4.1).

Another performance limiting factor is the increased complexity of speculative algorithms. This particularly regards the process of decision and range estimation of the computed BLFT. While the nonredundant algorithm may easily assume the input arguments to lie within $\mathbb{B} \times \mathbb{B}$, the redundant algorithm must at worst consider the domain $\pm\mathbb{B} \times \pm\mathbb{B}$. As the number of corners, where the computed BLFT must be evaluated, increases slightly more than twice, the performance degrades accordingly. However, processing nonspeculative digits incurs no performance overhead. Thus, by making the speculation less "aggressive" (i.e., tightening the overlaps in Fig. 3) we may control the final performance degradation.

6 Conclusions

Previous sections introduced the principle of speculative redundancy as a means to address computability issues of continued logarithm representation. Its formulation evolved from practical experiments based on the idea of eager emission suggested by Gosper [2]. However, while Gosper suggested to take emission in every cycle and retract if it gets wrong, our approach is less eager and avoids any retraction, thus, keeping the lazy nature of continued logarithms. The particular form (i.e., the choice of overlaps in Fig. 3) of the speculative concept presented here has a formal equivalent [19], making it easy to verify its correctness. But the idea of speculation is more general; it is indeed the key point of the whole concept. Speculation can be applied to any lazy sequence representation, for which there is a correction mechanism of incorrectly speculated answers (such as, e.g., reversibility of continued logarithms). Furthermore, the choice of speculation intervals changes the aggressivity of speculation, giving a chance to minimize or maximize its practical effects (such as better responsiveness or higher complexity).

Our intention was to make the proposed redundancy comply with the representation of [7] as much as possible, so that the hardware architecture developed therein could be easily extended. This attempt was quite successful as the speculatively redundant representation differs particularly in the decision process. Thus, using the "simulation" technique for range estimation (Section 4.4) and extending the decision conditions, the existing arithmetic unit could be well utilized. The corresponding changes of data- and control-path would be quite insignificant. However, the "simulation" approach may incur a performance overhead if speculation is used too often. Then, the hardware architecture would need to change more substantially—i.e., perform range estimation in all corners of $\pm\mathbb{B} \times \pm\mathbb{B}$ at once—to fully exploit its performance potential.

Further performance improvement is possible by running multiple units in a serialized, pipelined manner. Although suggested long time ago [11], bringing this idea to practice faces many open problems. First of all, one would need to change from the lazy top-down to a bottom-up evaluation scheme, so that the parallel architecture is efficiently utilized. This opens the question of resource allocation and scheduling, which would map portions of expression trees to the arithmetic units so as to minimize the overall computation time. Since the evaluation process is iterative and the number of iterations is hard to predict, "classical" scheduling algorithms [23] are likely not applicable. Another area that may affect the overall performance is expression tree optimization. Numerical algorithms are usually defined in terms of basic arithmetic operators, which might be compacted to a lower number of BLFT operators. This problem seems nontrivial as the experimental results show that equivalent expression trees may produce qualitatively different results, also depending on a selected absorption scheme.

Acknowledgments

The author would like to thank Peter Kornerup for his comments on an early draft of this manuscript and other works of the author. These comments discovered some weak points and helped to focus on them. Thanks go to Róbert Lórencz for suggestions on this final version. Finally, the author expresses his gratitude to anonymous reviewers for their helpful comments that made him significantly improve this manuscript.

References

[1] A. Khinchin, *Continued Fractions,* third ed. Univ. of Chicago Press, 1964, translated from Russian by P. Wynn, P. Noordhoff Ltd., 1963, and by H. Eagle, Univ. of Chicago Press, 1964.

[2] R.W. Gosper, "Continued Fraction Arithmetic," unpublished manuscript, http://www.tweedledum.com/rwg/cfup.htm, 1978.

[3] P. Gowland and D. Lester, "A Survey of Exact Arithmetic Implementations," *Proc. Computability and Complexity in Analysis (CCA '00),* J. Blanck, V. Brattka, and P. Hertling, eds., pp. 30-47, 2000.

[4] D.R. Lester, "Effective Continued Fractions," *Proc. 15th IEEE Symp. Computer Arithmetic,* pp. 163-170, 2001.

[5] H.J. Boehm, R. Cartwright, M. Riggle, and M.J. O'Donnell, "Exact Real Arithmetic: A Case Study in Higher Order Programming," *Proc. ACM Symp. LISP and Functional Programming,* pp. 162-173, 1986.

[6] O. Mencer, "Rational Arithmetic Units in Computer Systems," PhD thesis, EECS, Stanford Univ., http://www.doc.ic.ac.uk/~oskar/pubs/thesis.pdf, 2000.

[7] T. Brabec, "Hardware Implementation of Continued Logarithm Arithmetic," *Proc. 12th GAMM IMACS Int'l Symp. Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN '06) Conf. Post-Proc.,* 2006.

[8] P. Kornerup and D.W. Matula, "Finite Precision Lexicographic Continued Fraction Number Systems," *Proc. Seventh IEEE Symp. Computer Arithmetic,* pp. 207-214, 1985.

[9] M. Niqui, "Exact Arithmetic on the Stern-Brocot Tree," Technical Report NIII-R0325, Nijmeegs Instituut voor Informatica en Informateikunde, 2003.

[10] C. Mazenc, "On the Redundancy of Real Number Representation Systems," Research Report RR93-16, Laboratoire de l'Informatique du Parallélisme, ftp://ftp.ens-lyon.fr/pub/LIP/Rapports/RR/RR93/RR93-16.ps.Z, 1993.

[11] P. Kornerup and D.W. Matula, "An Algorithm for Redundant Binary Bit-Pipelined Rational Arithmetic," *IEEE Trans. Computers,* vol. 39, no. 8, pp. 1106-1115, Aug. 1990.

[12] T. Brabec, "Proof of Interior Containment," A Supplement to an Electronic Version of this Manuscript, http://doi.ieeecomputersociety.org/10.1109/TC.2010.110, 2009.

[13] T. Brabec, "On Progress of Investigations in Continued Logarithm Arithmetic," *Proc. Počítačové architektury a diagnostika 2007,* pp. 61-66, http://service.felk.cvut.cz/anc/brabect1/pub/pad2007extended.pdf, 2007.

[14] G.N. Raney, "On Continued Fractions and Finite Automata," *Mathematische Annalen,* vol. 206, no. 4, pp. 265-283, 1973.
[15] P.J. Potts, "Exact Real Arithmetic Using Möbius Transformations," PhD thesis, Imperial College, Univ. of London, http://www.doc.ic.ac.uk/~ae/papers/potts-phd.pdf, July 1998.
[16] R. Heckmann, "Contractivity of Linear Fractional Transformations," *Theoretical Computer Science,* vol. 279, nos. 1/2, pp. 65-82, 2002.
[17] R. Heckmann, "How Many Argument Digits are Needed to Produce n Result Digits?" *Electronic Notes in Theoretical Computer Science,* vol. 24, pp. 13-33, 1999.
[18] K.-I. Ko, "On the Continued Fraction Representation of Computable Real Numbers," *Theoretical Computer Science,* vol. 47, no. 3, pp. 299-313, 1986.
[19] T. Brabec, "Redundant Cont. Log. Representation: Proof of Contractivity," unpublished paper, http://service.felk.cvut.cz/anc/brabect1/pub/clproof08.pdf, 2008.
[20] T. Brabec, "Continued Logarithms," manuscript in preparation, http://service.felk.cvut.cz/anc/brabect1/pub/clreport08.pdf, 2008.
[21] R.W. Gosper, "Item 101 in Hakmem," AIM239, MIT, pp. 37-44, http://dspace.mit.edu/bitstream/1721.1/6086/2/AIM-239.pdf, Apr. 1972.
[22] T. Brabec, "Quantitative Results," A Supplement to an Electronic Version of this Manuscript, http://doi.ieeecomputersociety.org/10.1109/TC.2010.110, 2009.
[23] Y.-K. Kwok and I. Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors," *ACM Computing Surveys,* vol. 31, no. 4, pp. 406-471, 1999.

Tomáš Brabec received the MS degree in computer science and engineering from Czech Technical University (CTU) in Prague in 2004. He is currently working toward the PhD degree in computer science at CTU. His research focuses on computer arithmetic and its hardware implementation. He is also interested in computer architectures, soft cores, and FPGAs.