AN ORDER PRESERVING FINITE BINARY ENCODING OF THE RATIONALS*

David W. Matula
Southern Methodist University

Peter Kornerup
Aarhus University

ABSTRACT

We describe a new binary encoding for numbers termed lexicographic continued fraction (LCF) representation that provides a one-to-one order preserving finite bit string representation for every rational. Conversion either way between binary integer numerator-denominator pair representation and LCF representation is shown feasible in time linear with bit string length, given registers of length sufficient to hold the numerator and denominator. LCF bit string length is about $2 \max\{\log_2 p, \log_2 q\}$ for the irreducible fraction p/q. Realization of arithmetic (+, -, ×, ÷) on LCF bit string encoded operands is shown feasible. Some relations between the theory of best rational approximation and the values represented by truncated LCF bit strings are noted to assess the feasibility of a finite precision arithmetic based on LCF representation.

I. INTRODUCTION AND SUMMARY

The binary fixed-point representation of the fraction 19/44 is 0.01(1011101000), where the quantity in parenthesis repeats indefinitely. The binary integer representation of numerator and denominator for 19/44 is 10011/101100. For many applications it would be desirable to have a finite bit string representation for each rational where lexicographic order of the strings corresponds to the numeric order of the rationals. Neither the fixed-point nor the numerator-denominator pair representations provide both these features.

We propose a new binary encoding of numbers based on the continued fraction representation of a number that provides such an order preserving finite bit string representation for every rational. This lexicographic continued fraction (LCF) representation can be converted to and from the binary integer numerator-denominator pair representation in time linear with the number of bits, given full length registers of size sufficient to hold the numerator and denominator values. LCF representation of the fraction p/q yields a bit string of length roughly $2 \max\{\log_2 p, \log_2 q\}$, comparable to the combined length of the binary integer representations for numerator and denominator when p and q are of nearly equal magnitude. For example, the LCF representation for 19/44 is 00111010011.

Utilizing the notation $[a_0/a_1/a_2/\ldots]$ for the continued fraction

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}, \quad a_i > 0$$

where the partial quotients a_i are assumed to be integral, it is known from continued fraction theory [HW79, Ch 10] that any non-negative rational number p/q has a finite expansion

$$\frac{p}{q} = [a_0/a_1/\ldots/a_m]$$

which is unique (canonical) with the added requirements $a_0 \geq 0$; $a_i \geq 1$ for $i \geq 1$, and $a_m \geq 2$ whenever $m \geq 1$. Thus

$$\frac{19}{44} = [0/2/3/6] = \cfrac{1}{2 + \cfrac{1}{3 + \frac{1}{6}}} .$$

Any particular partial quotient a_i may be arbitrarily large, e.g.

$e = [2/1/2/1/1/4/1/1/6/\ldots/1/1/2j/\ldots]$.

Thus concatenation of fixed length fields each representing a partial quotient is not feasible in the manner standard for encoding digit sequence representations in positional radix representation to any base. In Section II we describe a variable length lexicographic encoding of the (positive) integers by bit strings of self limiting length that provides the foundation for our continued fraction representation.

In Section III we describe the formation of the LCF representation of $p/q = [a_0/a_1/\ldots/a_m]$ by concatenation, after specific modifications, of the variable length bit string encodings of the partial

* Research supported in part by the National Science Foundation under grant No. MCS8012704 and the Aarhus University Research Foundation.

quotients $a_0, a_1, \ldots, a_m$. We first show that LCF representation provides an order preserving one-to-one correspondence between the non-negative rationals (under numeric order) and the finite length bit strings (under lexicographic order). Conversion between LCF representation and standard binary integer pair representation for numerator and denominator is then shown to be straightforward in time linear in bit string length given registers sufficient to hold the numerator and denominator values. Employing continued fraction arithmetic ideas of Gosper [Go72] (see also [Kn81, p 602]) and procedures similar to the arithmetic unit for rational arithmetic described by Kornerup and Matula [KM83], we describe efficient direct algorithms for determining the LCF representation of the result of the arithmetic operations +, −, ×, ÷ on LCF represented operands. The possibility of supporting on-line as well as fixed precision arithmetic in LCF representation by suitable architectures derived from these algorithms is noted. Regarding the encoding efficiency of LCF representation, we employ a suitable model of frequency of continued fraction partial quotient values and show LCF representation requires about 3.51 bits per partial quotient value. This is noted to be less than 2% greater than the most compact (Huffman) encoding.

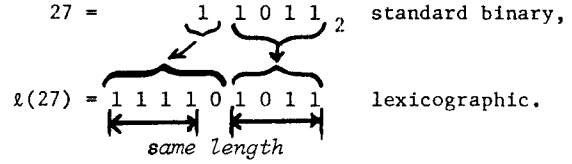
An extension to signed LCF representation (SLCF) is described in Section IV. SLCF representation conveniently admits a representation for (unsigned) infinity. Both unary operators NEGATE and INVERT are shown to be efficiently implemented by 2's complementation of the SLCF bit string with appropriate treatment of the "sign-bit".

The extension of LCF (or SLCF) representation to reals by infinite bit strings follows in a natural way from the limit of the encodings of continued fraction approximations to the particular real value. In Section V some elements of the theory of best rational approximation are recounted to analyze the properties of a finite precision number system derived from the set of values representable by LCF bit strings of limited length. Directions for further research are noted, such as: a modified LCF representation with finite precision range/accuracy tradeoffs comparable to floating-point systems; incorporation with multiple precision floating-slash systems; and on-line computation with LCF bit strings.

Our principal results are summarized in a series of highlighted observations throughout the text. The properties appear to us sufficient to consider LCF representation a worthwhile addition to the design possibilities in the architecture of number systems.

II. LEXICOGRAPHIC ENCODING OF THE INTEGERS

A full <u>k-bit integer</u> is an integer i whose standard binary representation contains k bits with leading bit unity, so then $2^{k-1} \le i \le 2^k - 1$. The variable length <u>lexicographic bit string</u> for the full k-bit integer i, denoted $\ell(i)$, is formed from the standard binary representation simply by replacing the leading unit by a string of (k-1) units and a zero. For example,



Throughout this paper we shall apply the subscript 2 to the bit strings for standard binary representation to distinguish them from the lexicographic bit strings, which are not subscripted.

Note that standard binary representation of an integer is right-adjusted where the left-hand boundary must be assumed known. Lexicographic bit strings are left-adjusted with a right-hand boundary determinable by a left-to-right scan of the string. Specifically, in reading the string 111101011 note that there are 5 bits up to and including the first zero 11110 which determines both the leading bit and the stopping rule, i.e. read 4 more bits 1011_2 in standard binary, obtaining $11011_2 = 27$. The left-to-right scan of the bit string $\ell(i)$ yields valuable information as each successive bit is read, first about the magnitude of i, and then a narrowing as in binary search of the range including i. Letting $b_1 b_2 b_3 \ldots$ denote in order the successive bits of the string $\ell(i)$, we obtain for $\ell(i) = 111101011$ the following information as each bit is read:

Bit Processed	Interpretation
$b_1 = 1$	$i \ge 2$
$b_2 = 1$	$i \ge 4$
$b_3 = 1$	$i \ge 8$
$b_4 = 1$	$i \ge 16$
$b_5 = 0$	$16 \le i \le 31$
$b_6 = 1$	$24 \le i \le 31$
$b_7 = 0$	$24 \le i \le 27$
$b_8 = 1$	$26 \le i \le 27$
$b_9 = 1$	$i = 27$

In summary we note the following.

<u>Observation 1</u>. *String Length:* The lexicographic bit string $\ell(i)$ has length $\lfloor 2 \log_2 i \rfloor + 1$ for all $i \ge 1$. □

Two bitstrings

$$a = a_1 a_2 a_3 \cdots ,$$

$$b = b_1 b_2 b_3 \cdots ,$$

can be compared in lexicographic order if we first assume the shorter string to be extended with 0's to the right as needed so they are both of the same length (finite or infinite). Then we say

$a \lessdot b$ (a lexicographically precedes b or b lexicographically follows a)

if and only if there exists an index i such that

$$a_j = b_j \quad \text{for all } j < i,$$

$$a_i < b_i .$$

Furthermore

$a \equiv b$ (a lexicographically equals b)

if and only if $a_j = b_j$ for all $j \geq 1$.

Observation 2. *Order:* $\ell(i)$ lexicographically follows $\ell(j)$ if and only if $i > j \geq 1$. □

Table 1 gives the standard binary and lexicographic bit string representations of the integers 1,2, ... ,20 .

Integer	Standard Binary	Lexicographic
1	1_2	0
2	10_2	100
3	11_2	101
4	100_2	11000
5	101_2	11001
6	110_2	11010
7	111_2	11011
8	1000_2	1110000
9	1001_2	1110001
10	1010_2	1110010
11	1011_2	1110011
12	1100_2	1110100
13	1101_2	1110101
14	1110_2	1110110
15	1111_2	1110111
16	10000_2	111100000
17	10001_2	111100001
18	10010_2	111100010
19	10011_2	111100011
20	10100_2	111100100

Table 1. Right-adjusted standard binary representation and left-adjusted lexicographic bit string representation of the integers 1, 2, ... , 20 .

III. LEXICOGRAPHIC ENCODING OF THE RATIONALS

Utilizing lexicographic integer encoding it is possible to represent any finite sequence of positive integers $p_1, p_2, \ldots, p_n$ by the finite bit string $\ell(p_1)\cdot\ell(p_2)\cdot \ldots \cdot\ell(p_n)$, where "·" denotes string concatenation. This string can readily be decoded back into the sequence of integer values by a left-to-right scan. For example, reading the first nine bits of the string 111101011110101100011101011100 as a lexicographic integer uniquely determines $p_1 = 27$ with the stopping rule indicating reading of this integer value is complete. Reading then continues with the tenth bit to determine the next integer value. The full string is decoded into the sequence 27,6,4,13,2.

In order to encode the sequence of integers $a_0, a_1, \ldots, a_m$ of the continued fraction $[a_0/a_1/a_2/ \ldots /a_m]$ as a bit string corresponding in a one-to-one order preserving manner with the rationals, three properties of continued fractions deserve special consideration in fabricating the "LCF representation." The example 19/44 = [0/2/3/6] will illustrate these considerations. We further make special note of the LCF representation for zero, which follows readily even though there is no lexicographic integer representation for zero.

i) Leading partial quotient of zero: From the definition of a continued fraction $[a_0/a_1/\ldots/a_m]$, $a_i \geq 1$ for $i \geq 1$ and $a_0 \geq 0$. We shall use the leading bit b_0 of the LCF representation for $[a_0/a_1/\ldots/a_m]$ to denote by $b_0 = 1$ that $a_0 \geq 1$ and the partial quotients that follow start with a_0; and by $b_0 = 0$ that $a_0 = 0$ and the partial quotients that follow start with a_1. Each partial quotient to be represented than satisfies $a_i \geq 1$, so $\ell(a_i)$ is uniquely determined for each a_i. Note that $b_0 = 1$ conveniently determines that the LCF represented rational value is ≥ 1, and $b_0 = 0$ indicates a value < 1.

Example: 19/44 = [0/2/3/6] must then have a leading zero bit in LCF representation, with further consideration than devoted to encoding the sequence 2,3,6.

ii) Continued fraction order: From the definition of continued fractions it follows for any $0 \leq i \leq m$ that

$$[a_0/a_1/\ldots/a_{i-1}/a_i/\ldots/a_m] \begin{cases} \leq [a_0/a_1/\ldots/a_{i-1}/a_i+1] & \text{for } i = 2j, \\ \geq [a_0/a_1/\ldots/a_{i-1}/a_i+1] & \text{for } i = 2j+1. \end{cases} \quad (1)$$

Simply stated, increasing a partial quotient in any even indexed position increases the value represented, **e.g.** [0/2/4/6] = 25/56 > 19/44 = [0/2/3/6]. Increasing **a partial quotient** in any odd indexed **position decreases** the value represented,

e.g. [0/2/3/7] = 22/51 < 19/44 = [0/2/3/6]. To impart lexicographic order to the bit string representations for the continued fractions $\{[a_0/a_1/\ldots/a_m]\}$ consistent with the numeric order of their rational values, we shall utilize the component substrings $\ell(a_i)$ whenever $i=2j$, and the 1's complement strings, denoted by $\bar{\ell}(a_i)$, whenever $i=2j+1$.

Example: Thus we are led to consider $0, \bar{\ell}(2), \ell(3), \bar{\ell}(6)$, for representation of 19/44 = [0/2/3/6] except for one final consideration.

iii) Trailing zeros: The continued fraction $[a_0/a_1/\ldots/a_m/\infty]$ is taken to be identical to $[a_0/a_1/\ldots/a_m]$. For lexicographic integer consistency we must assume $\ell(\infty)$ denotes an infinite sequence of 1's, hence $\bar{\ell}(\infty)$ denotes an infinite sequence of 0's. To properly assume that the finite bit string for any continued fraction $[a_0/a_1/\ldots/a_m]$ can be extended to the right with an infinite sequence of zeros, we are then limited to the representation of continued fractions where m is even. Thus we must choose to represent every non-negative rational with the alternative unique [HW79, Ch 10] terminal index even continued fraction $p/q = [a_0/a_1/\ldots/a_{2j}]$. Note that the standard canonical continued fraction $p/q = [a_0/a_1/\ldots/a_m]$ has $a_m \geq 2$ and will be the same as the terminal index even continued fraction when m is even. Since $p/q = [a_0/a_1/\ldots/a_m] = [a_0/a_1/\ldots/a_m - 1/1]$, the latter is the canonical terminal index even continued fraction for p/q for m odd. As a special case note that $[0/\infty] = 0$ is a terminal index even continued fraction. Hence the zero bitstring is the "natural" representation of zero. Since the LCF representation can be extended to the right with an arbitrary number of zeros without changing the lexicographic ordering, we will use the representation LCF(0) = 0.

Example: Thus for our desired LCF representation of 19/44 we note [0/2/3/6] = [0/2/3/5/1] and obtain $0\cdot\bar{\ell}(2)\cdot\ell(3)\cdot\bar{\ell}(5)\cdot\ell(1) = 00111010011100$, or equivalently simply 00111010011. LCF representation will be assumed to have an infinite string of zeros to the right and thus can be terminated at the last unit entry.

Formally, the lexicographic continued fraction (LCF) bit string representation corresponding to the unique terminal index even continued fraction $p/q = [a_0/a_1/a_2/\ldots/a_{2j}]$ for any rational number $p/q \geq 0$ is given by

$$\mathrm{LCF}(p/q) = \begin{cases} 1\cdot\ell(a_0)\cdot\bar{\ell}(a_1)\cdot\ldots\cdot\ell(a_{2i})\cdot\bar{\ell}(a_{2i+1})\cdot \\ \cdot\ldots\cdot\ell(a_{2j})\cdot\bar{\ell}(\infty) & \text{for } p \geq q \geq 1, \\ 0\cdot\bar{\ell}(a_1)\cdot\ldots\cdot\ell(a_{2i})\cdot\bar{\ell}(a_{2i+1})\cdot \\ \cdot\ldots\cdot\ell(a_{2j})\cdot\bar{\ell}(\infty) & \text{for } 0 \leq p < q, \end{cases} \qquad (2)$$

where the standard finite form of LCF(p/q) will be truncated at the last unit bit except for LCF(0) = 0.

For example,

LCF(22/7) = LCF([3/6/1])
determine continued fraction,
$= 1\cdot\ell(3)\cdot\bar{\ell}(6)\cdot\ell(1)\cdot\bar{\ell}(\infty)$
substitute lexicographic, complementing odd index positions,
$= 1\cdot101\cdot00101\cdot0\cdot0\ldots$

= 110100101
concatenate and truncate,

and

LCF(314/100) = LCF([3/7/7])

$= 1\cdot\ell(3)\cdot\bar{\ell}(7)\cdot\ell(7)\cdot\bar{\ell}(\infty)$

$= 1\cdot101\cdot00100\cdot11011\cdot0\ldots$

= 11010010011011 .

The process is clearly reversible. Specifically, by extending any finite bit string with an infinite sequence of zeros, a left-to-right scan uniquely determines a terminal index even continued fraction, hence a unique non-negative rational value. Note that the leading bit of the string indicates whether to commence decoding the next portion of the string as a lexicographic integer in standard or complemented form, after which the interpretation alternates between standard lexicographic and complemented lexicographic form. For example:

$11010010011101110\ldots = 1\cdot101\cdot00100\cdot1110111\cdot0\ldots$

$= 1\cdot\ell(3)\cdot\bar{\ell}(7)\cdot\ell(15)\cdot\bar{\ell}(\infty)$

= LCF([3/7/15])

= LCF(333/106) .

These reversible encoding and decoding procedures established the following.

Observation 3. *Uniqueness and Completeness:* LCF representation provides a one-to-one correspondence between all finite bit strings (assumed right extended with zeros) and all finite irreducible fractions $p/q \geq 0$. □

Note that the lexicographic order of the LCF bit strings of our preceding examples, $11010010011011 < 1101001001110111 < 110100101$, corresponds to the numeric order of the rational numbers the fractions represent,

$$\frac{314}{100} = 3.14 < \frac{333}{106} = 3.1415\ldots < \frac{22}{7} = 3.1428\ldots .$$

Lemma: $p/q > p'/q'$ if and only if LCF(p/q) lexicographically follows LCF(p'/q').

Proof: The result is immediate if $p/q \geq 1 > p'/q'$, so assume either $p/q > p'/q' \geq 1$ or $p'/q' < p/q < 1$. Let $p/q = [a_0/a_1/\ldots/a_m]$, $p'/q' = [a'_0/a'_1/\ldots/a'_n]$, where i is the smallest index for which the partial

quotients differ, i.e. $a_j = a'_j$ for $j < i$, $a_i \neq a'_i$. Append ∞ to the shorter partial quotient sequence if it is a subsequence of the other to determine i. Then from the definition of continued fractions and the lexicographic order for $\ell(k)$,

i) for i odd: $p/q > p'/q'$ iff $a_i < a'_i$ iff $\bar{\ell}(a'_i) \prec \bar{\ell}(a_i)$ iff $LCF(p'/q') \prec LCF(p/q)$,

ii) for i even: $p/q > p'/q'$ iff $a_i > a'_i$ iff $\ell(a'_i) \prec \ell(a_i)$ iff $LCF(p'/q') \prec LCF(p/q)$. □

Observation 4. *Lexicographic Order:* The lexicographic order on the LCF bit strings corresponds to the numeric order on the rational values they represent. □

Our next observation extracts significant architectural content from the LCF(p/q) bit string relevant to the operation of the Euclidean GCD algorithm on the fraction p/q.

Observation 5. *Primitive Euclidean GCD Program:* The bit string LCF(p/q) may be interpreted as a program encoding the operation of a binary Euclidean GCD "machine" on a dividend register (initialized to p) and a divisor register (initialized to q). Primitive operations are shift-left, shift-right, subtract-into, and shift-right-and-subtract-into. The initial bit sets the state as to whether the initial quotient digit is in an even index or odd index position. Interpretation of $\ell(a_{2i})$, the even index state, has each of the leading ones correspond to a single shift-left of the divisor register with the first zero then denoting subtract-into the dividend register. Succeeding register bits then encode for each 0 a single shift-right of the divisor register and for each 1 a shift-right-and-subtract-into of the divisor register into the dividend register until the divisor register reaches its original (unit) position. Interpretation then changes to the odd index state. Processing of $\bar{\ell}(a_{2i+1})$, the odd index state, interprets the complement of each successive bit in like manner to the even index state but on the opposite registers. The latter situation corresponds to the observation that the contents of the dividend register and divisor register must implicitly be interchanged after each partial quotient. The length of the LCF bit string is then a measure of the complexity of the GCD computation in terms of the binary shift, subtract, and shift-and-subtract primitives. Note that this encoding does not reflect the trial subtractions that need to be restored (or the equivalent in non restoring division) during a typical binary division instruction, but rather indicates only the successful primitives enroute to completion of the GCD algorithm (much like the encoding of a direct successful path through a maze). □

Observation 5 utilizes the fact that conversion from numerator-denominator pair representation to LCF representation is implicit from the Euclidean algorithm. Alternatively, assume given the continued fraction $[a_0/a_1/a_2/\ldots/a_m]$. Then with $p_{-2} = 0$, $p_{-1} = 1$, $q_{-2} = 1$, $q_{-1} = 0$, and p_i, q_i for $0 \leq i \leq m$ determined by

$$p_i = a_i p_{i-1} + p_{i-2}, \qquad q_i = a_i q_{i-1} + q_{i-2}, \tag{3}$$

we obtain $p_m/q_m = [a_0/a_1/a_2/\ldots/a_m]$. Thus conversion from LCF representation to numerator-denominator pair representation is equally straightforward.

We illustrate this conversion by interpretation of the LCF bit string 001011001. We employ two registers each for parallel construction of the numerator and denominator, where each bit of the LCF string denotes an appropriate "microcode instruction" on the registers implicitly computing the $\{p_i, q_i\}$ by equations (3).

Step	LCF String	Interpretation	Numerator Registers	Denominator Registers
0:	0	Initialize to R0 State;	00000 R0 00001 R1	00001 R0 00000 R1
		(Set to odd state)		
1:	0	Shift R0 Left;	0 0000 R0 00001 R1	0 0001 R0 00000 R1
2:	1	Add R0 Into R1;	0 0000 R0 00001 R1	0 0001 R0 00010 R1
3:	0	Shift R0 Right; Add R0 Into R1;	00000 R0 00001 R1	00001 R0 00011 R1
		(Set to even state)		
4:	1	Shift R1 Left;	00000 R0 0 0001 R1	00001 R0 0 0011 R1
5:	1	Shift R1 Left;	00000 R0 00 001 R1	00001 R0 00 011 R1
6:	0	Add R1 Into R0;	00100 R0 00 001 R1	01101 R0 00 011 R1
7:	0	Shift R1 Right;	00100 R0 0 0001 R1	01101 R0 0 0011 R1
8:	1	Shift R1 Right, Add R1 Into R0.	00101 R0 00001 R1	10000 R0 00011 R1
		(Set to odd state)		

For this example note that after Step 0 the R0 registers contain $p_0 = 0$, $q_0 = 1$; after Step 3 the R1 registers contain $p_1 = 1$, $q_1 = 3$; and after Step 8 the R0 registers contain $p_2 = 5$, $q_2 = 16$. Hence we obtain 5/16 as the rational value in numerator-denominator form corresponding to the LCF bit string 001011001.

Assume a computer model with arbitrarily large registers capable of bit wise parallel operations to effect SHIFT and the register pair logical operations AND and OR. Then each shift instruction as described in Observation 5 and employed in our preceding example requires only constant time.

Utilizing carry-save techniques each subtraction and/or addition between registers can be implemented in constant time, so we obtain the following.

Observation 6. *Conversion:* Conversion between LCF representation and binary integer numerator-denominator representation is possible in time linear in bit length, i.e. O(log p+log q) for the irreducible fraction p/q.

The conversion procedure derives from the Euclidean algorithm and assumes registers sufficiently large to hold p and q. □

The Euclidean algorithm can be extended to support arithmetic on continued fractions [Kn81, p. 602], [KM83]. Specifically, by initializing p_i, q_i for $i = -2, -1$ with the seed matrix

$$\begin{bmatrix} p_{-2} & q_{-2} \\ p_{-1} & q_{-1} \end{bmatrix} = \begin{bmatrix} a & c \\ b & d \end{bmatrix},$$

the recurrence equations (3) yield $f(p_m/q_m)$ where f is the bilinear form $f(x) = (a+bx)/(c+dx)$. The seed matrices

$$\begin{bmatrix} r & s \\ s & 0 \end{bmatrix}, \begin{bmatrix} r & s \\ s & 0 \end{bmatrix}, \begin{bmatrix} 0 & s \\ r & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & r \\ s & 0 \end{bmatrix},$$

implement the operations Add r/s, Subtract r/s, Multiply by r/s, and Divide by r/s, respectively, applied to the operand $[a_0/a_1/a_2/.../a_m]$. Arithmetic in this form on LCF encoded operands is asymmetric in that one of the operands must first be fully converted to numerator-denominator form, as in the example converting 001011001 to the pair 101/10000. Then the other LCF encoded operand is decoded with interpretation applied to a seed matrix constructed from the converted value (i.e. 101/10000) and the operation to be performed. The final result is then in numerator-denominator form and must be converted back to LCF form.

Gosper has noted [Go72] that this scheme can be generalized by introducing a $2 \times 2 \times 2$ matrix and working in three dimensions. Entries in the matrix correspond to coefficients of the expression:

$$f(x,y) = \frac{axy + bx + cy + d}{exy + fx + gy + h}$$

where x and y may be fed into the computation as continued fractions, whose partial quotients are used to perform computations as in (3) along two dimensions of the matrix. The entries of the $2 \times 2 \times 2$ cube thus constructed represent the value of f(x,y), corresponding to whatever parts of x and y have been fed into the computation so far. Along the third dimension one plane represents the numerator, and the other the denominator, and along this dimension the partial quotients of the continued fraction expansion of f(x,y) may be computed by the Euclidean Algorithm as in Observation 5.

Hence the cube represents an "on-line" model computation-cell which can receive two LCF bitstreams and emit the resulting LCF bitstream corresponding to the value of any expression of the form f(x,y) above (including the add, subtract, multiply and divide operations).

Assuming either the constant time bit parallel operations on arbitrarily long registers criteria for the asymmetric arithmetic or a constant time operation for bits input and output from the computation-cell model we obtain the following.

Observation 7. *Arithmetic:* The standard arithmetic operations (+, −, ×, ÷) can be performed on operands represented in LCF bit string form to yield the result in LCF form in time linear in the bit string lengths. □

Several important unary operations on LCF represented operands are natural and very straightforward to compute. For inversion note from the definition of continued fractions:

$$1/[a_0/a_1/a_2/.../a_m] = \begin{cases} [a_1/a_2/...a_m] & \text{for } a_0 = 0, \\ [0/a_0/a_1/.../a_m] & \text{for } a_0 \neq 0. \end{cases} \quad (4)$$

Inversion shifts the even/odd status of each partial quotient, and we simply obtain:

Observation 8. *Inversion by 2's Complementation:* For any finite non zero fraction p/q the 2's complement of LCF(p/q) is LCF(q/p). The 2's complement of LCF(1/1) = 1 is again 1, which is the unique representation of unity. The 2's complement of LCF(0/1) = 0 would improperly again yield 0 with, however, a carry out of the leading bit to the left, which could be used to recognize infinity as the computed result. □

Observation 9. *Extraction of Integral and Fractional Parts:* The values of the integral portion and the fractional or (mod 1) portion of p/q are readily obtained from reading out and separating the leading partial quotient a_0. □

Our final concern of this section: How efficient is the LCF representation viewed as an encoding given appropriate frequency data for the partial quotients? For the distribution of partial quotients over the continued fraction representation of all fractions i/j with $1 \leq i, j \leq n$, Knuth [Kn81] demonstrated that, as $n \to \infty$, the frequency with which a partial quotient has value j is $\log_2[1+1/j(j+2)]$. In particular,

Partial Quotient Value	Frequency
1	$\log_2(4/3) = 0.4150$
2	$\log_2(9/8) = 0.1699$
3	$\log_2(16/15) = 0.0931$
4	$\log_2(25/24) = 0.0589$
5	$\log_2(36/35) = 0.0406$

This leads to an expected partial quotient bit length in LCF representation of

$$\sum_{j=1}^{\infty} (1+2\lfloor \log_2 j \rfloor) \log_2[1+1/j(j+2)] = 3.51 .$$

An optimal Huffman encoding of characters with the frequency $\log_2[1+1/j(j+2)]$ for $j \geq 1$ is readily shown to require at least 3.45 bits per character where these encodings are not even required to preserve lexicographic order.

Observation 10. *Encoding Efficiency:* LCF representation requires approximately 3.51 bits per partial quotient over the expected distribution of partial quotients, which is less than 2% greater than the most compact encoding possible even if order preservation were not required. □

IV. SIGNED LCF ENCODING

A natural choice for the extension of LCF representation to include negative rationals is to attach a sign-bit to the left of the representation. To preserve lexicographic order it is necessary to:

i) use 1 for positive and 0 for negative numbers as the sign bit.

ii) complement the bit-pattern representing the absolute value of the number (complementation must be by 2's complementation in order that the representation ends in an infinite string of zeros).

Denoting the signed lexicographic encoding by SLCF we define as follows:

$$SLCF(\frac{p}{q}) = \begin{cases} 1 \cdot LCF(\frac{p}{q}) & \text{for } \frac{p}{q} \geq 0 , \\ 0 \cdot TC(LCF(-\frac{p}{q})) & \text{for } \frac{p}{q} < 0 , \end{cases}$$

where $TC(\cdot)$ is the 2's complement operator defined on bit-strings interpreted as integers. Thus in general we have:

$$SLCF(-\frac{p}{q}) = TC(SLCF(\frac{p}{q})) ,$$

which also applies to the representation of zero yielding

$$SLCF(0) = 1 ,$$

where no trailing zeros need be specified.

The inversion operator as discussed in Observation 8 now has to be revised appropriately, since the sign should not be complemented. Thus with $Tail(a_1a_2a_3\cdots) = a_2a_3\cdots$,

$$SLCF(\frac{q}{p}) = Sign(\frac{p}{q}) \cdot TC(Tail(SLCF(\frac{p}{q}))$$

whenever $p \neq 0$. Notice again that an attempt to form the 2's complement of an "empty tail" (any finite or infinite string of all zeros) results in a "carry-out", which can be used to recognize infinity as the result. The only possible representation of infinity would be the empty string, or equivalently any finite or infinite string of all zeros, which in lexicographic order would "compare low" to any other bitstring, and hence act as $-\infty$ in comparisons. The following illustrates the thirty-two different rational numbers that can be distinguished by the first five bits of their SLCF representation:

8	11111	-1/8	01111
4	11110	-1/4	01110
3	11101	-1/3	01101
2	11100	-1/2	01100
5/3	11011	-3/5	01011
3/2	11010	-2/3	01010
4/3	11001	-4/5	01001
1	11000	-1	01000
4/5	10111	-4/3	00111
2/3	10110	-3/2	00110
3/5	10101	-5/3	00101
1/2	10100	-2	00100
1/3	10011	-3	00011
1/4	10010	-4	00010
1/8	10001	-8	00001
0	10000	$\pm\infty$	00000

V. LCF ENCODING OF THE REALS

LCF representation of an irrational number x as an infinite bit string follows readily as the limit of the LCF encodings of the truncated rational approximations $p_{2j}/q_{2j} = [a_0/a_1/\ldots/a_{2j}]$ of the unique infinite continued fraction $x = [a_0/a_1/a_2/\ldots]$. The set of all infinite bit strings (except for those with an infinite terminal sequence of 1's) is then readily seen to be in one-to-one correspondence with the set of all reals, and bit string lexicographic order preserves real order. This extension thus provides a foundation for approximate real arithmetic with LCF representation. Note that differing initial k-bit strings for any two infinite LCF bit strings are sufficient to order the two distinct reals these infinite strings represent. For example, $LCF(\pi) = 11010010011101111111\ldots$, which is shown with sufficient length to confirm $333/106 < \pi < 22/7$ by comparison with our earlier demonstrated bit strings for LCF(333/106) and LCF(22/7).

Observation 11. *Real Representation:* LCF representation provides a one-to-one order preserving correspondence between all infinite bit strings (having no terminal sequence all 1's) and the reals, where the rationals are those strings having only a finite number of 1's in the string (hence a terminal sequence all 0's). □

The truncated continued fractions

$$\frac{p_i}{q_i} = [a_0/a_1/\ldots/a_i], \quad i = 0,1,\ldots,m$$

derived from the infinite continued fraction $x = [a_0/a_1/a_2/\ldots]$ form a sequence of continued fraction approximations of x called convergents. The convergents to x are also termed "best rational approximations" to x and their properties can be summarized from classical material on continued fractions [HW79, Ch10] as follows.

Theorem: The convergents $p_i/q_i = [a_0/a_1/\ldots/a_i]$ of $x = [a_0/a_1/a_2/\ldots]$ for $i = 0,1,2,\ldots$ satisfy the following properties:

(i) Recursive ancestry:

With $p_{-2} = 0$, $p_{-1} = 1$, $q_{-2} = 1$ and $q_{-1} = 0$,

$$p_i = a_i p_{i-1} + p_{i-2},$$

$$q_i = a_i q_{i-1} + q_{i-2},$$

(ii) Irreducibility:

$$\gcd(p_i, q_i) = 1,$$

(iii) Adjacency:

$$q_i p_{i-1} - p_i q_{i-1} = (-1)^i,$$

(iv) Alternating convergence: for $x \neq 0$,

$$\frac{p_0}{q_0} < \frac{p_2}{q_2} < \ldots < \frac{p_{2j}}{q_{2j}} < \ldots < x < \ldots < \frac{p_{2j-1}}{q_{2j-1}} < \ldots < \frac{p_1}{q_1},$$

(v) Best rational approximation:

$$\frac{r}{s} \neq \frac{p_i}{q_i},\ s \leq q_i \Rightarrow \left|\frac{r}{s} - x\right| > \left|\frac{p_i}{q_i} - x\right|,$$

(vi) Quadratic convergence:

$$\frac{1}{q_i(q_{i+1} + q_i)} < \left|\frac{p_i}{q_i} - x\right| \leq \frac{1}{q_i q_{i+1}}.$$

Observation 12. *Rounding to Convergents:* Every convergent p_{2j}/q_{2j} to $p/q \geq 0$ has $p_{2j}/q_{2j} \leq p/q$ and $\mathrm{LCF}(p_{2j}/q_{2j})$ is an initial substring of $\mathrm{LCF}(p/q)$. Thus chopping of $\mathrm{LCF}(p/q)$ to the appropriate place can determine $\mathrm{LCF}(p_{2j}/q_{2j})$. However, not all choppings of $\mathrm{LCF}(p/q)$ yield convergents of p/q, so it is necessary to read and interpret $\mathrm{LCF}(p/q)$ from left-to-right to find those that do. Similarly, every convergent p_{2j+1}/q_{2j+1} to p/q has $p_{2j+1}/q_{2j+1} > p/q$ and can be found by a rounding by augmentation to the appropriate number of bits. Again it is necessary to read and interpret $\mathrm{LCF}(p/q)$ left-to-right to find those augmentations corresponding to convergents. □

Let a k-bit continued fraction denote any irreducible fraction p/q for which $\mathrm{LCF}(p/q)$ has no 1's beyond the initial k bits. This definition allows any k-bit string followed by an infinite string of 0's to be read left-to-right yielding a unique k-bit continued fraction. There are then 2^k distinct k-bit continued fractions which are shown for $k = 5$ and $p/q \leq 1$ in Figure 1.

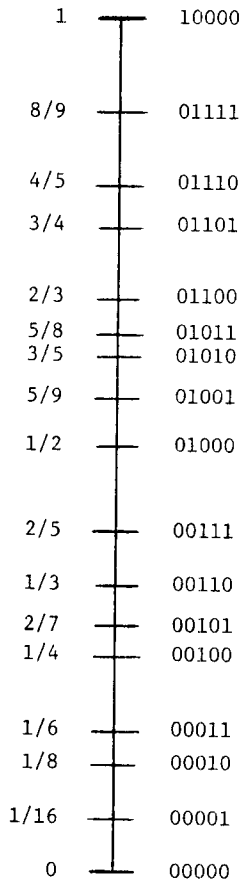


Figure 1. The 17 LCF 5-bit Strings Having Values in the Unit Interval

Note that expansion from k-bit to (k+1)-bit continued fractions adds exactly one new representable value in every previous gap. The new values serve to bisect each gap but may be off center as much as two parts to one. The off center positions serve, however, to capture all the relatively simple fractions at appropriately small values of k.

The inverse of every k-bit continued fraction other than 0/1 is again a k-bit continued fraction. Note also that the negative of any signed rational number requiring at most k+1 bits in SLCF representation is representable in k+1 bits in SLCF representation. Any convergent to a k-bit continued fraction is a k-bit continued fraction. The k-bit fractions for $k \to \infty$ do not, however, yield the property of being asymptotically uniformly dense on the interval [0,1] as do the fixed-slash factions [MK80]. This shortcoming is the result of a base dependent bias inherent in the specification of the k-bit continued fractions.

The specification and investigation of a finite precision continued fraction number system based on

k-bit LCF representation appears worthy of further study. Alternative representation of the leading partial quotient a_0 (or a_1 if $a_0 = 0$) can be employed to yield a variation of LCF representation where the resulting "k-bit" finite precision number system can achieve range/accuracy trade offs comparable to floating-point or floating-slash [MK80] number systems. Integration of some aspects of LCF representation with the fixed-and floating-slash arithmetic unit described in [KM83] provides an alternative approach to multiple precision slash arithmetic. Utilization of LCF representation for on-line continued fraction arithmetic in the manner suggested by Gosper [Go72] is still another direction for further study.

We believe the properties demonstrated and the potential for further enhancements makes LCF representation a useful tool in the design of number systems and arithmetic units.

REFERENCES

[Go72] R. W. Gosper, Item 101 in HAKMEM, MIT-AIM 239, Feb. 1972, pp 37-44.

[HW79] G. H. Hardy and E. M. Wright, An Introduction to the Theory of Numbers, 5th Ed., Clarendon Press, Oxford, 1979.

[Kn81] D. E. Knuth, The Art of Computer Programming, Vol 2, Seminumerical Algorithms, 2nd Ed., Addison-Wesley, Reading, Mass., 1981.

[KM83] P. Kornerup and D. W. Matula, Finite precision rational arithmetic: an arithmetic unit, IEEETC Vol C-32, April 1983.

[MK80] D. W. Matula and P. Kornerup, Foundations of finite precision rational arithmetic, Computing, Suppl. 2, Springer Verlag (1980) pp 85-111.