JFP **16** (3): 281–291, 2006.
doi:10.1017/S0956796806005880 First published online 3 March 2006 Printed in the United Kingdom

FUNCTIONAL PEARL
Enumerating the rationals

JEREMY GIBBONS[1], DAVID LESTER[2] and RICHARD BIRD[1]

[1]*University of Oxford, Parks Road, Oxford, UK*
{jeremy.gibbons,richard.bird}@comlab.ox.ac.uk
[2]*University of Manchester, Manchester, UK*
dlester@cs.man.ac.uk

1 Introduction

Every lazy functional programmer knows about the following approach to enumerating the positive rationals: generate a two-dimensional matrix (an infinite list of infinite lists), then traverse its finite diagonals (an infinite list of finite lists). Each row of the matrix has the positive rationals with a given denominator, and each column those with a given numerator:

$$\begin{array}{l} {}^{1}\!/_{1}\ {}^{2}\!/_{1}\ {}^{3}\!/_{1}\ \cdots\ {}^{m}\!/_{1}\ \cdots \\ {}^{1}\!/_{2}\ {}^{2}\!/_{2}\ {}^{3}\!/_{2}\ \cdots\ {}^{m}\!/_{2}\ \cdots \\ \quad\vdots \\ {}^{1}\!/_{n}\ {}^{2}\!/_{n}\ {}^{3}\!/_{n}\ \cdots\ {}^{m}\!/_{n}\ \cdots \\ \quad\vdots \end{array}$$

Since each row is infinite, the rows cannot simply be concatenated. However, each of the diagonals from upper right to lower left, containing rationals with numerator and denominator of a given sum, is finite, so these can be concatenated:

$$\begin{array}{l} rats_1 :: [Rational] \\ rats_1 = concat\ (diags\ [[{}^{m}\!/_{n} \mid m \leftarrow [1..]] \mid n \leftarrow [1..]]) \\ diags = diags'\ [\,] \\ \qquad \textbf{where}\ diags'\ xss\ (ys : yss) = map\ head\ xss : diags'\ (ys : map\ tail\ xss)\ yss \end{array}$$

Equivalently, one can deforest the matrix altogether, and generate the diagonals directly:

$$\begin{array}{l} rats_2 :: [Rational] \\ rats_2 = concat\ [[{}^{m}\!/_{d-m} \mid m \leftarrow [1..d-1]] \mid d \leftarrow [2..]] \end{array}$$

All very well, but the resulting enumeration of the positive rationals contains duplicates – in fact, infinitely many duplicates of every rational.

One could enumerate the rationals without duplication indirectly, by filtering the co-prime pairs from those generated as above. In this paper, however, we explain an elegant technique for enumerating the positive rationals *directly, without duplicates.* Moreover, we show how to do so as a simple *iteration*, generating each element

of the enumeration from the previous one alone, with constant cost (in terms of number of arbitrary-precision simple arithmetic operations) per element. Best of all, the resulting programs are extremely simple – simpler even than the two programs above. The mathematical results are not new (Calkin & Wilf, 2000; Newman, 2003); however, we believe that they deserve wider appreciation in the functional programming community. Besides, the exercise provides some compelling examples of unfolds on infinite trees.

2 Greatest common divisor

The diagonalization approach to enumerating the rationals is based on generating the pairs of positive integers. The essence of the problem with this approach is that the natural correspondence via division between integer pairs and rationals is not a bijection: although every rational is represented, many integer pairs represent the same rational. Obviously, therefore, enumerating the rationals by generating the integer pairs yields duplicates.

Equally obviously, a solution to the problem can be obtained by finding a simple-to-enumerate set with a simple-to-compute bijection to the rationals. Both constraints on simplicity are necessary. The naturals are simple to enumerate, and there clearly exists a bijection between the naturals and the rationals; but this bijection is not simple to compute. On the other hand, there is a simple bijection from the rationals to themselves, but that still begs the question of how to enumerate the rationals.

The crucial insight is the relationship between rationals and greatest common divisors. Recall Euclid's subtractive algorithm for computing greatest common divisor:

$$\begin{aligned} &gcd && :: (Integer, Integer) \to Integer \\ &gcd\ (m,n) &&= \textbf{if } m < n \textbf{ then } gcd\ (m, n-m) \textbf{ else} \\ &&&\quad \textbf{if } m > n \textbf{ then } gcd\ (m-n, n)\ \textbf{ else } m \end{aligned}$$

Consider the following 'instrumented version', that returns not only the greatest common divisor, but also a trace of the execution by which it is computed:

$$\begin{aligned} &igcd && :: (Integer, Integer) \to (Integer, [Bool]) \\ &igcd\ (m,n) &&= \textbf{if } m < n \textbf{ then } step\ False\ (igcd\ (m, n-m)) \textbf{ else} \\ &&&\quad \textbf{if } m > n \textbf{ then } step\ True\ (igcd\ (m-n, n))\ \textbf{ else } (m, []) \\ &&&\quad\quad \textbf{where } step\ b\ (d, bs) = (d, b : bs) \end{aligned}$$

Given a pair (m,n), the function $igcd$ returns a pair (d,bs), where d is $gcd\ (m,n)$ and bs is the list of booleans recording the 'execution path' – that is, a list of the branches taken – when evaluating $gcd\ (m,n)$. Let us introduce the function $pgcd$, so that $bs = pgcd\ (m,n)$. These two pieces of data together are sufficient to invert the computation and reconstruct m and n – that is, given:

$$\begin{aligned} &ungcd :: (Integer, [Bool]) \to (Integer, Integer) \\ &ungcd\ (d, bs) = foldr\ undo\ (d,d)\ bs \\ &\quad\quad \textbf{where } undo\ False\ (m,n) = (m, n+m) \\ &\quad\quad\quad\quad\ undo\ True\ (m,n) = (m+n, n) \end{aligned}$$

then $ungcd$ and $igcd$ are each other's inverses, and so there is a bijection between integer pairs (m,n) and their images (d,bs) under $igcd$.

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

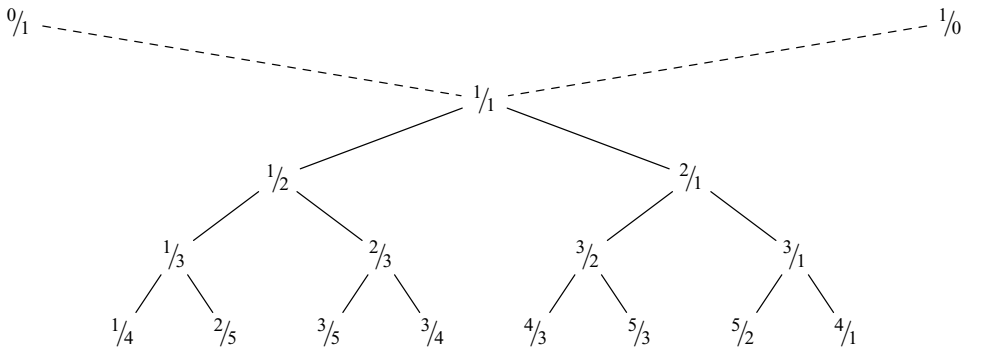

Fig. 1. The first few levels of the Stern-Brocot tree.

Now, $gcd\ (m,n)$ is exactly what is superfluous in the mapping from (m,n) to the rational $^m/_n$, and $pgcd\ (m,n)$ is exactly what is relevant in this mapping, since two pairs (m,n) and (m',n') represent the same rational iff they have the same $pgcd$:

$$^m/_n = {}^{m'}/_{n'} \iff pgcd\ (m,n) = pgcd\ (m',n')$$

Moreover, $pgcd$ is surjective: every finite boolean sequence is the $pgcd$ of some pair. The function $ungcd$ gives a constructive proof of this, by reconstructing such pairs. Therefore we can enumerate the rationals by enumerating the finite boolean sequences: the enumeration is easy enough, and the bijection to the rationals is simple to compute, via $ungcd$:

$$\begin{array}{ll} rats_3 & :: [Rational] \\ rats_3 & = map\ (mkRat \circ curry\ ungcd\ 1)\ boolseqs \\ boolseqs & = [\,] : [b : bs \mid bs \leftarrow boolseqs, b \leftarrow [False, True]] \\ mkRat\ (m,n) & = {}^m/_n \end{array}$$

3 The Stern-Brocot tree

A standard way of representing a mapping from finite strings over some alphabet is with a *trie*: a tree of degree equal to the size of the alphabet, in which the paths form the (prefixes of all the) strings in the domain of the mapping, and the image of every string is located in the tree at the end of the corresponding path (Knuth, 1998; Thue, 1912). In this case, the alphabet is binary, with the two symbols *False* and *True*, so the tree is binary too; and every finite string is in the domain of the mapping, so every node of the tree is the location of some rational. The first few levels are shown in Figure 1 (the significance of the two pseudo-nodes labelled $^0/_1$ and $^1/_0$ will be made clear shortly). For example, $pgcd\ (3,4)$ is $[False, True, True]$, so the rational $^3/_4$ appears at the end of the path $[L,R,R]$, that is, as the rightmost grandchild of the left child of the root; the root is labelled $^1/_1$, since $(1,1)$ yields the empty execution path. This tree turns out to be well-known; Graham, Knuth and Patashnik (1994, section 4.5) call it the *Stern-Brocot tree*, after its two independent nineteenth-century discoverers. It enjoys the following two properties, among many

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

others:

- The tree is an infinite binary search tree, so any finite pruning has an increasing inorder traversal.

For example, pruning to include the level with $1/3$ and $3/1$ but nothing deeper yields a tree with inorder traversal $1/3, 1/2, 2/3, 1/1, 3/2, 2/1, 3/1$, which is increasing.

- Every node is labelled with a rational $\frac{m+m'}{n+n'}$, the 'intermediary' of $\frac{m}{n}$, the label of its rightmost left ancestor, and $\frac{m'}{n'}$, that of its leftmost right ancestor.

For example, the node labelled $3/4$ has ancestors $2/3, 1/2, 1/1, 0/1, 1/0$, of which $1/1$ and $1/0$ are to the right and the others to the left. The rightmost left ancestor is $2/3$, and the leftmost right ancestor $1/1$, and indeed $3/4 = \frac{2+1}{3+1}$. That is why we included the two pseudo-nodes $0/1$ and $1/0$ in Figure 1: they are needed to make this relationship work for nodes like $1/3$ and $3/1$ on the boundary of the tree proper.

The latter property explains how to generate the tree directly, dispensing with the sequences of booleans. The seed from which the tree is grown consists of its rightmost left and leftmost right ancestors, initially the two pseudo-nodes. The tree root is their intermediary, which then acts as one half of the seed for each subtree.

```
data Tree a          = Node (a, Tree a, Tree a)
foldt f (Node (a,x,y)) = f (a, foldt f x, foldt f y)
unfoldt f x          = let (a,y,z) = f x in Node (a, unfoldt f y, unfoldt f z)

rats4                :: [Rational]
rats4                = bf (unfoldt step ((0,1),(1,0)))
                         where step (l,r) = let m = adj l r in
                                                (mkRat m, (l,m), (m,r))
adj (m,n) (m',n')    = (m + m', n + n')
bf                   = concat ∘ foldt glue
                         where glue (a,xs,ys) = [a] : zipWith (++) xs ys
```

Alternatively, one could deforest the tree itself and generate the levels directly. Start with the first level, consisting of the two pseudo-nodes, and repeatedly insert new nodes $\frac{m+m'}{n+n'}$ between each existing adjacent pair $\frac{m}{n}, \frac{m'}{n'}$.

```
rats5                :: [Rational]
rats5                = concat (unfolds infill [(0,1),(1,0)])
unfolds f a          = let (b,a') = f a in b : unfolds f a'
infill xs            = (map mkRat ys, interleave xs ys)
                         where ys = zipWith adj xs (tail xs)
interleave (x : xs) ys = x : interleave ys xs
interleave []       [] = []
```

An additional interesting property of the Stern-Brocot tree is that it forms the basis for a number representation system (credited by Graham, Knuth and Patashnik to Minkowski in 1904, exactly a century ago at the time of writing). Every rational is represented by the unique finite boolean sequence recording the path to it in the tree. An irrational number is represented by the unique infinite boolean sequence that

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

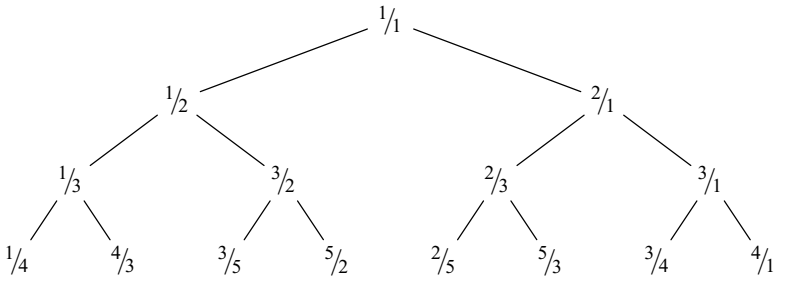

Fig. 2. The first few levels of the Calkin-Wilf tree.

converges on where it belongs; for example, $5/2 < e < 3/1$, so e has a representation starting $[True, True, False, True, \ldots]$.

4 The Calkin-Wilf tree

The Stern-Brocot tree is the trie of the mapping from boolean sequences $pgcd\ (m, n)$ to rationals m/n. But since all boolean sequences appear in the domain of this mapping (the tree is complete), so do their reverses, and we might just as well build the mapping from the reverse of $pgcd\ (m, n)$ to the same rational m/n. We call this tree the Calkin-Wilf tree, after its two explorers (Calkin & Wilf, 2000), whose work is promoted as one of Aigner and Ziegler's *Proofs from The Book* (2004, Chapter 16). The first few levels of the Calkin-Wilf tree are shown in Figure 2.

Whereas in the Stern-Brocot tree the path from the root to a node m/n records the trace of the computation of $gcd\ (m, n)$, in the Calkin-Wilf tree it is the path *to* the root *from* that node that records the trace. One might argue that this orientation is more natural.

Of course, a given level k of the Calkin-Wilf tree and of the Stern-Brocot tree contain the same collection of rationals (namely, those on which Euclid's subtractive algorithm takes k steps); but the two collections are generally in a different order: the Calkin-Wilf tree is not a binary search tree.

In fact, each level of the Calkin-Wilf tree is the *bit-reversal permutation* (Hinze, 2000; Bird *et al.*, 1999) of the corresponding level of the Stern-Brocot tree. For example, if the elements of the lowest level shown in Figure 1 are numbered in binary 000 to 111 from left to right, they appear in Figure 2 in the order $000, 100, 010, 110, 001, 101, 011, 111$, which are the reversals of the binary numbers 000 to 111. Bit-reversal of the levels arises naturally from reversal of the paths.

The binary search tree property of the Stern-Brocot tree is appealing, so it is a shame to lose it. However, the loss has its compensations. For one thing, indexing the tree by the reverses of the execution paths means that executions with common endings, rather than common beginnings, are grouped together. A consequence of this is that the ancestors in the Calkin-Wilf tree of a rational m/n record all the states that Euclid's algorithm visits when starting at the pair (m, n). For example, one execution path of Euclid's algorithm is the sequence of pairs $(3, 4), (3, 1), (2, 1), (1, 1)$, and indeed the ancestors in the Calkin-Wilf tree of $3/4$ are $3/1, 2/1, 1/1$. (Compare this

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

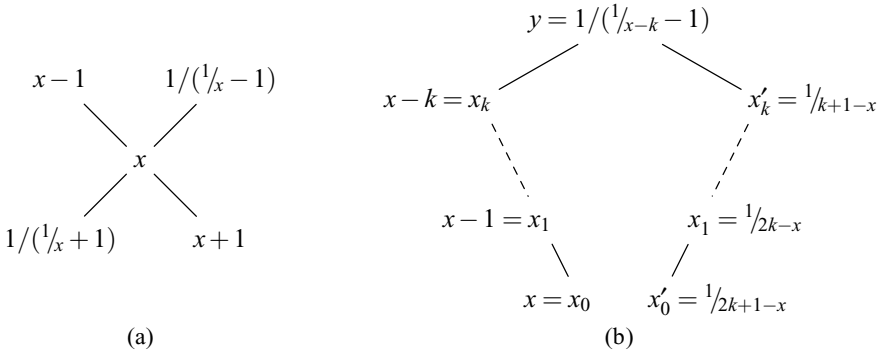

Fig. 3. The neighbours (a) and successor (b) of an element x in the Calkin-Wilf tree.

with the Stern-Brocot tree, in which there is no obvious relationship between parents and children.) Thus, a rational m/n with $m < n$ is the left child of the rational $m/n-m$, whereas if $m > n$ it is the right child of $m-n/n$. Equivalently, a rational m/n has left child $m/m+n$ and right child $n+m/n$. This shows how to generate the Calkin-Wilf tree:

$$
\begin{aligned}
&\mathit{rats}_6 :: [\mathit{Rational}] \\
&\mathit{rats}_6 = \mathit{bf}\ (\mathit{unfoldt}\ \mathit{step}\ (1,1)) \\
&\qquad \textbf{where}\ \mathit{step}\ (m,n) = (m/n, (m, m+n), (n+m, n))
\end{aligned}
$$

5 Iterating through the rationals

However, there is an even better compensation for the loss of the ordering property in moving from the Stern-Brocot to the Calkin-Wilf tree: it becomes possible to deforest the tree altogether, and generate the rationals directly, maintaining no additional state beyond the 'current' rational. This startling observation is due to Moshe Newman (Newman, 2003). In contrast, it is not at all obvious how to do this for the Stern-Brocot tree; the best we can do seems to be to deforest the tree as far as its levels, but this still entails additional state of increasing size.

We will generate the rationals using the *iterate* operator, computing each from the previous one.

$$
\begin{aligned}
&\mathit{iterate} \quad :: (a \to a) \to a \to [a] \\
&\mathit{iterate}\ f\ x = x : \mathit{iterate}\ f\ (f\ x)
\end{aligned}
$$

It is clear how to do this in some cases; for example, if m/n is a left child, then $m < n$, the parent is $m/n-m$, and the successor is the right child of the parent, namely $n/n-m$. In terms of $x = m/n < 1$, the parent is $1 / (1/x - 1)$, and the successor is the right child of this, or $1 + 1 / (1/x - 1) = 1/1-x$. (The relationship between a node and its possible neighbours is illustrated in Figure 3(a).)

More generally, x and its successor x' have a more distant ancestor in common. This situation is illustrated in Figure 3(b). Here, $x_0 = x$ is a right child of a parent $x_1 = x - 1$, itself the right child of $x_2 = x_1 - 1 = x - 2$, and so on up to $x_k = x - k$, which is a left child. Therefore $x_k < 1$, and so $k = \lfloor x \rfloor$, the integer part of x. Element x_k is the left child of the common ancestor $y = 1 / (1/x-k - 1)$, whose right child is

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

$x'_k = {}^1\!/_{1-(x-k)} = {}^1\!/_{k+1-x}$. Element x'_k has left child $x'_{k-1} = 1 \,/\, {}^1\!/_{x'_k+1} = {}^1\!/_{k+2-x}$, which has left child $x'_{k-2} = {}^1\!/_{k+3-x}$, and so on down to $x' = x'_0 = {}^1\!/_{2\times k+1-x} = {}^1\!/_{\lfloor x\rfloor+1-\{x\}}$ (where $\{x\} = x - \lfloor x\rfloor$ is the fractional part of x), which is the successor of x.

The formula $x' = {}^1\!/_{\lfloor x\rfloor+1-\{x\}}$ for the successor of x even works in the last remaining case, when x is on the right boundary and x' on the left boundary one level lower: then x is an integer, so $\lfloor x\rfloor = x$ and $\{x\} = 0$, and indeed $x' = {}^1\!/_{\lfloor x\rfloor+1-\{x\}}$. This motivates the following enumeration of the rationals:

$$
\begin{aligned}
&\mathit{rats}_7 \quad :: [\mathit{Rational}]\\
&\mathit{rats}_7 \quad = \mathit{iterate}\ \mathit{next}\ 1\\
&\mathit{next}\ x = \mathit{recip}\ (\mathit{fromInteger}\ n + 1 - y)\ \textbf{where}\ (n, y) = \mathit{properFraction}\ x
\end{aligned}
$$

Each term is generated from its predecessor with a constant number of rational arithmetic operations. (The Haskell standard library functions *properFraction* and *recip* take x to $(\lfloor x\rfloor, \{x\})$ and ${}^1\!/_x$, respectively.)

Could there be any simpler way to enumerate the positive rationals?

Calkin & Wilf (2000) discuss some additional properties of this enumeration. It is not hard to show that the numerator of the successor $\mathit{next}\ x$ of a rational x is the denominator of x, so in fact the sequence of numerators $1, 1, 2, 1, 3, 2, 3\ldots$ determines the sequence of rationals. This sequence is actually the solution to a natural counting problem: the ith element, starting from zero, counts the number of ways to write i in a redundant binary representation in which each digit may be 0, 1 or 2. For example, the fourth element is 3, and indeed there are three such ways of writing 4, namely 100, 20 and 12. Dijkstra also explored this sequence (Dijkstra, 1982a; Dijkstra, 1982b), which he called *fusc*; he showed, among other things, that $\mathit{fusc}\ n = \mathit{fusc}\ n'$ where n' is the bit-reversal of n — another connection with bit-reversal permutations.

Of course, it is not difficult to generate all the rationals, zero and negative as well as positive, in the same way — zero is a special initial case, and after that the positive rationals alternate with their negations:

$$
\begin{aligned}
&\mathit{rats}_8 :: [\mathit{Rational}]\\
&\mathit{rats}_8 = \mathit{iterate}\ \mathit{next}'\ 0\\
&\qquad \textbf{where}\ \mathit{next}'\ 0 \qquad\qquad\quad = 1\\
&\qquad\qquad \mathit{next}'\ x \mid x > 0 \quad\ = \mathit{negate}\ x\\
&\qquad\qquad\qquad\quad \mid \mathit{otherwise} = \mathit{next}\ (\mathit{negate}\ x)
\end{aligned}
$$

6 The continued fraction connection

Some additional insights into these algorithms for enumerating the rationals may be obtained by considering the continued fraction representation of the rationals. We write the finite continued fraction:

$$
a_0 + \cfrac{1}{a_1 + \cfrac{1}{\cdots + \cfrac{1}{a_n}}}
$$

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

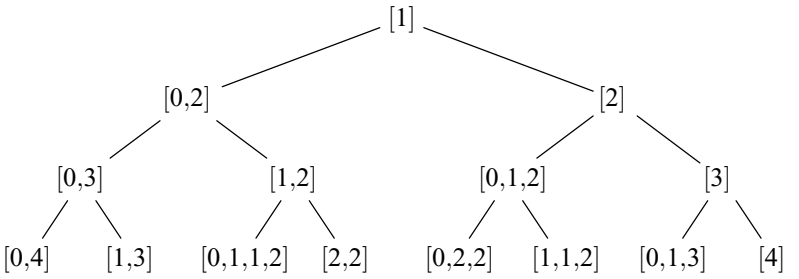

Fig. 4. The first few levels of the Calkin-Wilf tree, as continued fractions.

as the sequence of integer coefficients $[a_0, a_1, \ldots, a_n]$. For example, $\frac{3}{4}$ is $0+1/(1+\frac{1}{3})$, so is represented by $[0, 1, 3]$. Every rational has a unique normal form as a *regular* continued fraction; that is, as a finite sequence $[a_0, a_1, \ldots, a_n]$ under the constraints that $a_i > 0$ for $i > 0$ and that $a_n > 1$ if $n > 0$. Figure 4 shows the first few levels of the Calkin-Wilf tree with rationals expressed as continued fractions.

We have shown that the positive rationals are the iterates of the function taking x to $\frac{1}{\lfloor x \rfloor + 1 - \{x\}}$, whose computation requires a constant number of arithmetic operations on rationals. Division is required in order to compute $\lfloor x \rfloor$. However, if we represent rationals by regular continued fractions, then this division can be avoided: the integer part of a rational is simply the first term of the continued fraction. In fact, most of the required operations are easy to implement: the fractional part is obtained by setting the first term to zero, incrementing is a matter of incrementing the first term, and reciprocating either removes a leading zero (if present) or prefixes a leading zero (if not). Only negation is not so obvious. However, it turns out that a straightforward case analysis suffices, as the reader may check:

$$\begin{aligned}
&\mathit{negatecf}\ [n_0] &&= [-n_0] \\
&\mathit{negatecf}\ [n_0, 2] &&= [-n_0 - 1, 2] \\
&\mathit{negatecf}\ (n_0 : 1 : n_2 : ns) &&= (-n_0 - 1) : (n_2 + 1) : ns \\
&\mathit{negatecf}\ (n_0 : n_1 : ns) &&= (-n_0 - 1) : 1 : (n_1 - 1) : ns
\end{aligned}$$

Given this implementation of negation, it is straightforward to derive the following data refinement of rats_7. That is, if c is the continued fraction representation of rational x, then $\mathit{nextcf}\ c$ is the continued fraction representation of $\lfloor x \rfloor + 1 - \{x\}$.

$$\begin{aligned}
&\textbf{type}\ CF = [\mathit{Integer}] \\
&\mathit{rats}_9 :: [CF] \\
&\mathit{rats}_9 = \mathit{iterate}\ (\mathit{recipcf} \circ \mathit{nextcf})\ [1] \\
&\quad \textbf{where}\ \mathit{nextcf}\ [n_0] &&= [n_0 + 1] \\
&\qquad\quad \mathit{nextcf}\ [n_0, 2] &&= [n_0, 2] \\
&\qquad\quad \mathit{nextcf}\ (n_0 : 1 : n_2 : ns) &&= n_0 : (n_2 + 1) : ns \\
&\qquad\quad \mathit{nextcf}\ (n_0 : n_1 : ns) &&= n_0 : 1 : (n_1 - 1) : ns \\
&\qquad\quad \mathit{recipcf}\ (0 : ns) &&= ns \\
&\qquad\quad \mathit{recipcf}\ ns &&= 0 : ns
\end{aligned}$$

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

For example, consider the third clause for *nextcf*. If x is represented by $c = n_0 : 1 : n_2 : ns$, then $\lfloor x \rfloor = n_0$, and $\{x\}$ is represented by $0 : 1 : n_2 : ns$; this negates to $(-1) : (n_2 + 1) : ns$, which when increased by $n_0 + 1$ yields $n_0 : (n_2 + 1) : ns$.

This uses a constant number of arbitrary-precision integer additions and subtractions per term, but no divisions or multiplications. Of course, the result will be a list of continued fractions. These can be converted to rationals with the following function:

$$\begin{aligned} &cf2rat :: CF \to Rational \\ &cf2rat = mkRat \circ foldr\ op\ (1,0) \\ &\qquad \textbf{where}\ op\ m\ (n,d) = (m \times n + d, n) \end{aligned}$$

This uses additions and multiplications linear in the size of the continued fraction, but again no divisions (because coprimality of the pairs (n,d) is invariant under $op\ m$).

An additional thing that strikes the observer here is that the coefficients of the continued fractions on every level of the Calkin-Wilf tree sum to the same value, which is also the depth of that level. This is easy to justify when one considers the translation of Figure 3 to continued fractions: an element x has right child $x + 1$ (and incrementing a continued fraction is a matter of incrementing the first term, and hence incrementing the sum) and left child $1 / (1/x + 1)$ (and reciprocating a continued fraction is a matter of either prefixing or removing a leading zero, neither of which changes the sum). As a corollary, note that there are exactly 2^{k-1} regular positive continued fractions that sum to k.

Graham *et al.* (1994, section 6.7) present a connection between the continued-fraction Stern-Brocot tree and Euclid's algorithm; we translate their observations here to the Calkin-Wilf tree. They show that the path to an element x in the tree is directly related to the continued fraction of x: if the path to x is $L^{a_n} R^{a_{n-1}} L^{a_{n-2}} \cdots R^{a_0}$, then x is represented by the continued fraction $[a_0, a_1, \ldots, a_n + 1]$ (which is not regular if $a_n = 0$, but normalizes then to $[a_0, a_1, \ldots, a_{n-1} + 1]$). For example, the rational $3/4$ appears at the end of the path $L^0 R^2 L^1 R^0$, so has the continued fraction representation $[0, 1, 2, 0 + 1]$, which normalizes to $[0, 1, 3]$ as expected.

This view of paths, in which consecutive steps in the same direction are grouped together, conforms to the usual presentation of Euclid's algorithm using division instead of subtraction:

$$\begin{aligned} &gcd\ \ :: (Integer, Integer) \to Integer \\ &gcd\ (m,n) = \textbf{if}\ m < n\ \textbf{then}\ gcd\ (m, n \bmod m)\ \textbf{else} \\ &\qquad\qquad \textbf{if}\ m > n\ \textbf{then}\ gcd\ (m \bmod n, n)\ \textbf{else}\ m \end{aligned}$$

Each modulus computation casts out a certain number of multiples of the modulus, which corresponds in the Calkin-Wilf tree to a certain number of consecutive steps in the same direction. Graham, Knuth and Patashnik's observation therefore demonstrates a connection between the number of terms in the continued fraction representation of m/n and the number of steps taken to compute $gcd\ (m,n)$ by Euclid's division-based algorithm.

https://doi.org/10.1017/S0956796806005880 Published online by Cambridge University Press

Acknowledgements

The authors would like to express their thanks to members and friends of the Algebra of Programming group at Oxford (especially Roland Backhouse, Sharon Curtis, Graham Hutton, Andres Löh and Bruno Oliveira), Cristian Calude, and the anonymous JFP referees, who made numerous suggestions for improving the presentation of this paper.

We would especially like to thank Boyko Bantchev, who in a personal communication showed us an alternative construction

$$\begin{aligned} sb &= zipW\ mkRat\ (t,u) \\ &\quad \textbf{where}\ t\ = Node\ (1,t,zipW\ (uncurry\ (+))\ (t,u)) \\ &\qquad\ \ u = mirror\ t \end{aligned}$$

of the Stern-Brocot tree, where

$$\begin{aligned} zipW\ f\ &= unfoldt\ (apply\ f) \\ &\quad \textbf{where}\ apply\ f\ (Node\ (a,t,u), Node\ (b,v,w)) = (f\ (a,b),(t,v),(u,w)) \end{aligned}$$

and

$$mirror = foldt\ switch\ \textbf{where}\ switch\ (a,t,u) = Node\ (a,u,t)$$

That is, the denominator tree is the mirror image of the numerator tree; the numerator tree has 1 at the root, itself as its left child, and the element-wise sum of the numerator and denominator trees as its right child.

Boyko Bantchev and Cristian Calude brought to our attention work by D. N. Andreev (n.d.) and Shen Yu-Ting (1980), respectively. They define yet another enumeration of the positive rationals; although neither mentions trees, they describe in effect the construction

$$\begin{aligned} &rats_{10} :: [Rational] \\ &rats_{10} = bf\ (unfoldt\ step\ (1,1)) \\ &\qquad\quad \textbf{where}\ step\ (m,n) = ({}^{m}\!/_{n}, (n+m,n), (n,n+m)) \end{aligned}$$

The elements on each level are the same as in the Stern-Brocot and Calkin-Wilf trees, but a different order again; like the Stern-Brocot tree, this tree also does not give rise to an iterative enumeration of the rationals.

We would never have embarked upon this problem at all without the inspiration of Aigner and Ziegler's beautiful book (Aigner & Ziegler, 2004), promoting, among others, the elegant work of Calkin and Wilf (Calkin & Wilf, 2000) and Newman (Newman, 2003). The code is formatted with Andres Löh's and Ralf Hinze's wonderful lhs2TEX.

References

Aigner, M. and Ziegler, G. M. (2004) *Proofs from The Book*. Third edn. Springer-Verlag.

Andreev, D. E. *On a remarkable enumeration of the positive rational numbers*. In Russian. Available at `ftp://ftp.mccme.ru/users/vyalyi/matpros/i2126134.pdf.zip`.

Bird, R., Gibbons, J. and Jones, G. (1999) Program optimisation, naturally. In: Davies, J., Roscoe, A. W. and Woodcock, J. (eds.), *Millenial Perspectives in Computer Science*, pp. 13–21. Palgrave.

Calkin, N. and Wilf, H. (2000) Recounting the rationals. *American Mathematical Monthly*, **107**(4), 360–363. `http://www.math.upenn.edu/~wilf/website/recounting.pdf`.

Dijkstra, E. W. (1982a) EWD 570: An exercise for Dr R. M. Burstall. *Selected Writings on Computing: A personal perspective*, pp. 215–216. Springer-Verlag.

Dijkstra, E. W. (1982b) EWD 578: More about function '*fusc*'. *Selected Writings on Computing: A personal perspective*, pp. 230–232. Springer-Verlag.

Graham, R. L., Knuth, D. E. and Patashnik, O. (1994) *Concrete Mathematics: A foundation for computer science*. Second edn. Addison-Wesley.

Hinze, R. (2000) Perfect trees and bit-reversal permutations. *J. Funct. Program.* **10**(3), 305–317.

Knuth, D. E. (1998) *The Art of Computer Programming*. Second edn. Vol. 3. Addison-Wesley.

Newman, M. (2003) Recounting the rationals, continued. Credited in *American Mathematical Monthly*, **110**, 642–643.

Thue, A. (1912) Uber die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Skrifter udgivne af Videnskabs-Selskabet i Christiana*, **1**, 1–67. (Reprinted in *Selected Mathematical Papers of Axel Thue*, Universitetsforlaget, Oslo, 1977, p413–477.)

Yu-Ting, S. (1980) A 'natural' enumeration of non-negative rational numbers. *American Mathematical Monthly*, **87**(1), 25–29.