Recounting the Rationals: Twice!

Roland Backhouse
João Ferreira

March 30, 2007

Abstract

We derive an algorithm for enumerating the rationals in two different ways. One is known as the Calkin-Wilf-Newman enumeration; the second enumerates the rationals in Stern-Brocot order. The second method has recently been described as "not at all obvious how to do". In contrast, we show that it is obvious. More importantly, we show that both enumerations stem from the same simple algorithm.

Recently, there has been a spate of interest in the construction of bijections between the natural numbers and the (positive) rationals [GLB06, AZ04, New03, CW00]. Gibbons *et al* [GLB06] describe as "startling" the observation that the rationals can be straightforwardly enumerated by effectively "flattening" the Calkin-Wilf tree of rationals. However, they claim that it is "not at all obvious" how to flatten the Stern-Brocot tree of rationals. (For information on the Stern-Brocot tree, see [GKP89].)

In this paper, we derive an algorithm for enumerating the rationals both in Calkin-Wilf and Stern-Brocot order. The algorithm is based on a bijection between the rationals and invertible 2×2 matrices. The key to the algorithm's derivation is the reformulation of Euclid's algorithm in terms of matrices.

1 Euclid's Algorithm

A positive rational in so-called "lowest form" is an ordered pair of positive, coprime integers. Every rational $\frac{m}{n}$ has unique lowest-form representation $\frac{m/(m\nabla n)}{n/(m\nabla n)}$. (We use "$\nabla$" to denote "greatest common divisor". We prefer to use an infix notation whenever —as in this case— the operator is symmetric and associative. As we see below, the exploitation of symmetry and associativity is extremely important to effective reasoning.)

Because computing the lowest-form representation involves computing greatest common divisors, it seems sensible to investigate Euclid's algorithm to see whether it gives insight into how to enumerate the rationals. Indeed it does.

Below we present Euclid's algorithm as it might be presented in a modern textbook. (We use Dijkstra's Guarded Command Language [Dij75] to express the algorithm because it allows us to fully express the symmetry between m and n. The "do-od" statement is executed repeatedly. Termination occurs when both of the two guards $x>y$ and $y>x$ are false (i.e. when x and y are equal). When $x>y$ evaluates to true, the assignment $x := x-y$ is executed, and then the do-od is executed again. Similarly, when $y>x$ the assignment $y := y-x$ is executed before repeated execution of the do-od statement.

$\{\ 0<m \wedge 0<n\ \}$

$x,y := m,n\ ;$

$\{$ **Invariant**: $0<x \wedge 0<y \wedge x\triangledown y = m\triangledown n$

Bound function: $x+y\ \}$

do $x>y \rightarrow x := x-y$

$\square\ \ y>x \rightarrow y := y-x$

od

$\{\ x = y = m\triangledown n\ \}$

The algorithm below is a somewhat unusual, but very effective way, of rewriting Euclid's algorithm when the goal is to establish the theorem that the greatest common divisor of two numbers is a linear combination of the numbers.

The algorithm is expressed in matrix terms. The input to the algorithm is a vector $(m\ n)$ of strictly positive integers. The vector $(x\ y)$ is initialised to $(m\ n)$ and, on termination, its value is the vector $(m\triangledown n\ \ m\triangledown n)$. In addition to computing the greatest common divisor, it also computes a matrix $\mathbf{C}$. An invariant of the algorithm is that the vector $(x\ y)$ equals $(m\ n)\times\mathbf{C}$. In words, $(x\ y)$ is a "linear combination" of $(m\ n)$. Specifically, $\mathbf{I}$, $\mathbf{A}$ and $\mathbf{B}$ are 2×2 matrices; $\mathbf{I}$ is the identity matrix $\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$, $\mathbf{A}$ is the matrix $\left(\begin{smallmatrix}1&0\\-1&1\end{smallmatrix}\right)$ and $\mathbf{B}$ is the matrix $\left(\begin{smallmatrix}1&-1\\0&1\end{smallmatrix}\right)$. The assignment $(x\ y) := (x\ y)\times\mathbf{A}$ is equivalent to $x,y := x-y,y$, as can be easily checked. (Note that verifying the invariant is a simple consequence of the associativity of matrix multiplication.)

$\{\ 0<m \wedge 0<n\ \}$

$(x\ y),\mathbf{C} := (m\ n),\mathbf{I}\ ;$

$\{$ **Invariant**: $(x\ y) = (m\ n)\times\mathbf{C}\ \}$

do $y<x \rightarrow (x\ y),\mathbf{C} := (x\ y)\times\mathbf{A}\ ,\ \mathbf{C}\times\mathbf{A}$

$\square\ \ x<y \rightarrow (x\ y),\mathbf{C} := (x\ y)\times\mathbf{B}\ ,\ \mathbf{C}\times\mathbf{B}$

od

$$\{\ (x\ y)\ =\ (m\nabla n\ \ m\nabla n)\ =\ (m\ n)\times\mathbf{C}\ \}$$

It is this form of the algorithm that is the starting point for our enumeration of the rationals.

2 Enumerating the Rationals

Beginning with an arbitrary pair of positive integers m and n, the above algorithm calculates an invertible matrix $\mathbf{C}$ such that

$$(m\nabla n\ \ m\nabla n)\ =\ (m\ n)\times\mathbf{C}\ \ .$$

It follows that

$$(1)\quad (1\ 1)\times\mathbf{C}^{-1}\ =\ (m/(m\nabla n)\ \ n/(m\nabla n))\ \ .$$

Because the algorithm is deterministic, positive integers m and n uniquely define the matrix $\mathbf{C}$. That is, there is a function from pairs of positive integers to finite products of the matrices $\mathbf{A}$ and $\mathbf{B}$.

Also, because the matrices $\mathbf{A}$ and $\mathbf{B}$ are constant and invertible, $\mathbf{C}^{-1}$ is a finite product of the matrices $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$ and (1) uniquely defines a rational $\frac{m}{n}$. We may therefore conclude that there is a bijection between the rationals and the finite products of the matrices $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$ provided that we can show that all such products are different.

The finite products of matrices $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$ form a tree with root the identity matrix (the empty product). Renaming $\mathbf{A}^{-1}$ as $\mathbf{L}$ and $\mathbf{B}^{-1}$ as $\mathbf{R}$, the tree can be displayed with "L" indicating a left branch and "R" indicating a right branch. Fig. 1 displays the first few levels of the tree.

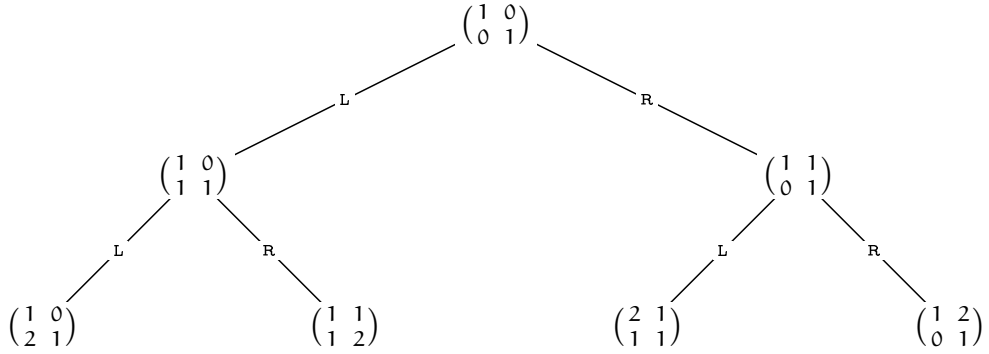

Figure 1: Tree of Products of $\mathbf{L}$ and $\mathbf{R}$

That all matrices in the tree are different is proved by showing that the tree is a binary search tree (as formalised shortly). The key element of the proof is that the

determinants of $\mathbf{A}$ and $\mathbf{B}$ are both equal to 1 and, hence, the determinant of any finite product of $\mathbf{L}$s and $\mathbf{R}$s is also 1 (because of the algebraic properties of determinants)[1].

Formally, we define the relation $\prec$ on 2×2 matrices by

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \prec \begin{pmatrix} a' & c' \\ b' & d' \end{pmatrix} \quad \equiv \quad \frac{a+c}{b+d} < \frac{a'+c'}{b'+d'} \quad .$$

We prove that, for all finite products of $\mathbf{L}$s and $\mathbf{R}$s and matrices $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$:

$$(2) \quad \mathbf{X}\times\mathbf{L}\times\mathbf{Y} \prec \mathbf{X} \prec \mathbf{X}\times\mathbf{R}\times\mathbf{Z} \quad .$$

It immediately follows that there are no duplicates in the tree of matrices because the relation $\prec$ is clearly transitive and a subset of the inequality relation. (Property (2) formalises precisely what we mean by the tree of matrices forming a binary search tree: the entries are properly ordered by the relation $\prec$, with matrices in the left branch being "less than" the root matrix which is "less than" matrices in the right branch.)

In order to show that

$$(3) \quad \mathbf{X}\times\mathbf{L}\times\mathbf{Y} \prec \mathbf{X} \qquad ,$$

suppose $\mathbf{X} = \left(\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}\right)$ and $\mathbf{Y} = \left(\begin{smallmatrix} a' & c' \\ b' & d' \end{smallmatrix}\right)$. Then, since $\mathbf{L} = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, (3) is easily calculated to be

$$\frac{(a+c)\times a' + (c\times b') + (a+c)\times c' + (c\times d')}{(b+d)\times a' + (d\times b') + (b+d)\times c' + (d\times d')} < \frac{a+c}{b+d} \quad .$$

That this is true is also a simple calculation; as observed earlier, the key property is that the determinant of $\mathbf{X}$ is 1, i.e. $a\times d - b\times c = 1$. The proof that $\mathbf{X} \prec \mathbf{X}\times\mathbf{R}\times\mathbf{Z}$ is similar.

Of course, we can also express Euclid's algorithm in terms of transpose matrices. Instead of writing assignments to the vector $(x\ y)$, we can write assignments to its transpose $\begin{pmatrix} x \\ y \end{pmatrix}$. Noting that $\mathbf{A}$ and $\mathbf{B}$ are transpose matrices, the assignment

$$(x\ y)\,,\mathbf{C} \quad := \quad (x\ y)\times\mathbf{A}\ ,\ \mathbf{C}\times\mathbf{A}$$

in the body of Euclid's algorithm becomes

$$\begin{pmatrix} x \\ y \end{pmatrix}\ ,\mathbf{C} \quad := \quad \mathbf{B}\times\begin{pmatrix} x \\ y \end{pmatrix}\ ,\ \mathbf{B}\times\mathbf{C} \quad .$$

Similarly, the assignment

$$(x\ y)\,,\mathbf{C} \quad := \quad (x\ y)\times\mathbf{B}\ ,\ \mathbf{C}\times\mathbf{B}$$

[1] The proof is an adaptation of the proof in [GKP89] that the rationals in the Stern-Brocot tree are all different. Our use of determinants corresponds to the use of "the fundamental fact" (4.31) [GKP89].

becomes

$$\binom{x}{y}\,,\mathbf{C} \;:=\; \mathbf{A}\times\binom{x}{y}\,,\mathbf{A}\times\mathbf{C}\quad.$$

On termination, the matrix $\mathbf{C}$ computed by the revised algorithm will of course be different; the pair $\binom{m/(m\nabla n)}{n/(m\nabla n)}$ is recovered from it by the identity

$$\mathbf{C}^{-1}\times\binom{1}{1} \;=\; \binom{m/(m\nabla n)}{n/(m\nabla n)}\quad.$$

In this way, we get a second bijection between the rationals and the finite products of the matrices $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$. This is the basis for our second method of enumerating the rationals.

In summary, we have:

Theorem 4 Define the matrices $\mathbf{L}$ and $\mathbf{R}$ by

$$\mathbf{L} \;=\; \begin{pmatrix}1&0\\1&1\end{pmatrix}\quad\text{and}\quad \mathbf{R} \;=\; \begin{pmatrix}1&1\\0&1\end{pmatrix}\quad.$$

Then the following algorithm computes a bijection between the rationals and the finite products of $\mathbf{L}$ and $\mathbf{R}$. Specifically, the bijection is given by the function that maps rational $\frac{m}{n}$ to the matrix $\mathbf{C}$ constructed by the algorithm together with the function from a finite product, $\mathbf{C}$, of $\mathbf{L}$s and $\mathbf{R}$s to $(1\;1)\times\mathbf{C}$. (The comments added to the algorithm supply the information needed to verify this assertion.)

$$\begin{array}{l}
\{\;0<m\;\wedge\;0<n\;\}\\
(x\;y)\,,\mathbf{C}\;:=\;(m\;n)\,,\mathbf{I}\;;\\
\{\;\textbf{Invariant}\colon\quad (m\;n)\;=\;(x\;y)\times\mathbf{C}\;\}\\
\mathrm{do}\;\;y<x\;\rightarrow\;(x\;y)\,,\mathbf{C}\;:=\;(x\;y)\times\mathbf{L}^{-1}\,,\mathbf{L}\times\mathbf{C}\\
\Box\;\;x<y\;\rightarrow\;(x\;y)\,,\mathbf{C}\;:=\;(x\;y)\times\mathbf{R}^{-1}\,,\mathbf{R}\times\mathbf{C}\\
\mathrm{od}\\
\{\;(x\;y)\;=\;(m\nabla n\;\;m\nabla n)\quad\wedge\quad(\,m/(m\nabla n)\;\;n/(m\nabla n)\,)\;=\;(1\;1)\times\mathbf{C}\;\}
\end{array}$$

Similarly, by applying the rules of matrix transposition to all expressions in the above, Euclid's algorithm constructs a second bijection between the rationals and finite products of the matrices $\mathbf{L}$ and $\mathbf{R}$. Specifically, the bijection is given by the function that maps rational $\frac{m}{n}$ to the matrix $\mathbf{C}$ constructed by the revised algorithm together with the function from finite products, $\mathbf{C}$, of $\mathbf{L}$ and $\mathbf{R}$ to $\mathbf{C}\times\binom{1}{1}$.

□

2.1 Enumerating Products of L and R

The problem of enumerating the rationals has been transformed to the problem of enumerating all finite products of the matrices $\mathbf{L}$ and $\mathbf{R}$. As observed earlier, the matrices are naturally visualised as a tree —recall fig. 1— with left branching corresponding to multiplying (on the right) by $\mathbf{L}$ and right branching to multiplying (on the right) by $\mathbf{R}$.

By premultiplying each matrix in the tree by $(1\ 1)$, we get a tree of rationals. (Premultiplying by $(1\ 1)$ is accomplished by adding the elements in each column.) This tree is called the Calkin-Wilf tree [GLB06, AZ04, CW00]. The tree is shown in fig. 2. In this figure the vector $(x\ y)$ has been displayed as $\frac{y}{x}$. (Note the order of x and y. This is to aid comparison with existing literature.)

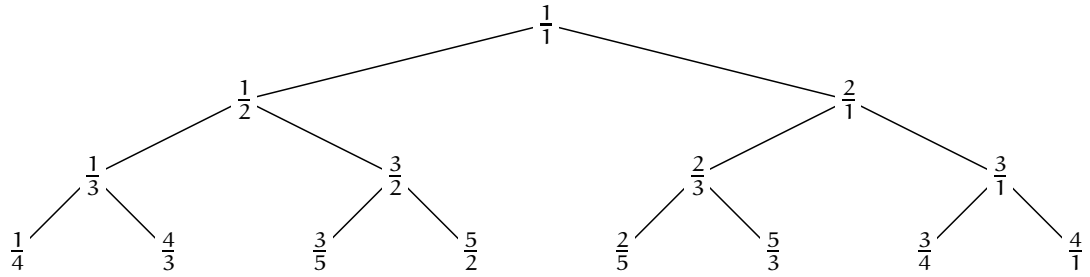

Figure 2: Calkin-Wilf Tree of Rationals

By postmultiplying each matrix in the tree by $\binom{1}{1}$, we also get a tree of rationals. (Postmultiplying by $\binom{1}{1}$ is accomplished by adding the elements in each row.) This tree is called the Stern-Brocot tree [GKP89]. See fig. 3. In this figure, the vector $\binom{x}{y}$ has been displayed as $\frac{x}{y}$.

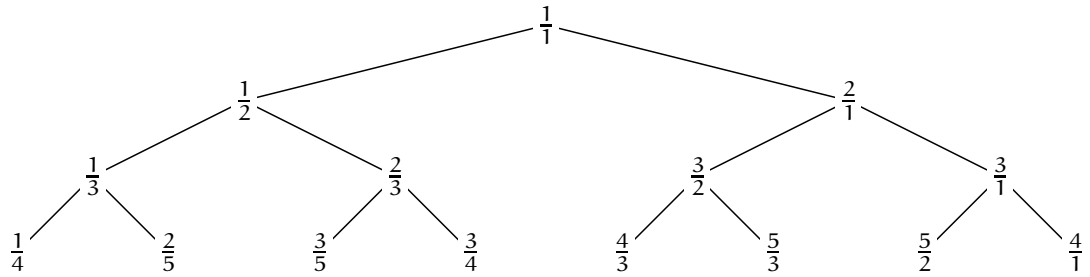

Figure 3: Stern-Brocot Tree of Rationals

Of course, if we can find an effective way of enumerating the matrices in fig. 1, we immediately get an enumeration of the rationals as displayed in the Calkin-Wilf tree and

as displayed in the Stern-Brocot tree — as each matrix is enumerated, simply premultiply by $(1\ 1)$ or postmultiply by $\binom{1}{1}$. Formally, the matrices are enumerated by enumerating all strings of Ls and Rs in lexicographic order, beginning with the empty string; each string is mapped to a matrix by the homomorphism that maps "L" to $\mathbf{L}$, "R" to $\mathbf{R}$, and string concatenation to matrix product. It is easy to enumerate all such strings; as we see shortly, converting strings to matrices is also not difficult, for the simple reason that $\mathbf{L}$ and $\mathbf{R}$ are invertible.

Clearly, the enumeration proceeds level-by-level. Beginning with the unit matrix (level 0), the matrices on each level are enumerated from left to right. There are 2^k matrices on level k, the first of which is $\mathbf{L}^k$. The problem is to determine for a given matrix, which is the matrix "adjacent" to it. That is, given a matrix $\mathbf{C}$, which is a finite product of $\mathbf{L}$ and $\mathbf{R}$, and is different from $\mathbf{R}^k$ for all k, what is the matrix that is to the immediate right of $\mathbf{C}$ in fig. 1.

Consider the lexicographic ordering on strings of Ls and Rs of the same length. The string immediately following a string s (that is not the last) is found by identifying the rightmost L in s. Supposing s is the string tLR^j, where R^j is a string of j Rs, its successor is tRL^j.

It's now easy to see how to transform the matrix identified by s to its successor matrix. Simply postmultiply by $\mathbf{R}^{-j} \times \mathbf{L}^{-1} \times \mathbf{R} \times \mathbf{L}^j$. This is because, for all $\mathbf{T}$ and j,

$$(\mathbf{T} \times \mathbf{L} \times \mathbf{R}^j) \times (\mathbf{R}^{-j} \times \mathbf{L}^{-1} \times \mathbf{R} \times \mathbf{L}^j) \;=\; \mathbf{T} \times \mathbf{R} \times \mathbf{L}^j \quad .$$

Also, it is easy to calculate $\mathbf{R}^{-j} \times \mathbf{L}^{-1} \times \mathbf{R} \times \mathbf{L}^j$. Specifically,

$$\mathbf{R}^{-j} \times \mathbf{L}^{-1} \times \mathbf{R} \times \mathbf{L}^j \;=\; \begin{pmatrix} 2j+1 & 1 \\ -1 & 0 \end{pmatrix} \quad .$$

(We omit the details. By induction, $\mathbf{L}^j$ equals $\left(\begin{smallmatrix} 1 & 0 \\ j & 1 \end{smallmatrix}\right)$. Also $\mathbf{R}$ is the transpose of $\mathbf{L}$. The rest is mechanical.)

The final task is to determine, given a matrix $\mathbf{C}$, which is a finite product of $\mathbf{L}$s and $\mathbf{R}$s, and is different from $\mathbf{R}^k$ for all k, the unique value j such that $\mathbf{C} = \mathbf{T} \times \mathbf{L} \times \mathbf{R}^j$ for some $\mathbf{T}$. This can be determined by examining Euclid's algorithm once more.

The matrix form of Euclid's algorithm, discussed above, computes a matrix $\mathbf{C}$ given a pair of positive numbers m and n; it maintains the invariant

$$(m\ n) \;=\; (x\ y) \times \mathbf{C} \quad .$$

$\mathbf{C}$ is initially the identity matrix and x and y are initialised to m and n, respectively; immediately following the initialisation process, $\mathbf{C}$ is repeatedly premultiplied by $\mathbf{R}$ so long as x is less than y. Simultaneously, y is reduced by x. The number of times that $\mathbf{C}$ is premultiplied by $\mathbf{R}$ is thus the greatest number j such that $j \times m$ is less than n, which

is $\left\lfloor \frac{n-1}{m} \right\rfloor$. Now, on termination of the algorithm, $(1\ 1) \times \mathbf{C}$ equals $(\ \frac{m}{m \nabla n}\ \ \frac{n}{m \nabla n}\ )$. That is, if

$$\mathbf{C} \;=\; \begin{pmatrix} \mathbf{C}_{00} & \mathbf{C}_{01} \\ \mathbf{C}_{10} & \mathbf{C}_{11} \end{pmatrix} \quad ,$$

then

$$\left\lfloor \frac{n-1}{m} \right\rfloor \;=\; \left\lfloor \frac{\mathbf{C}_{01}+\mathbf{C}_{11}-1}{\mathbf{C}_{00}+\mathbf{C}_{10}} \right\rfloor \quad .$$

It remains to decide how to keep track of the levels in the tree. For this purpose, it is not necessary to maintain a counter. It suffices to observe that the entries of $\mathbf{C}$ are all non-zero unless $\mathbf{C}$ is a power of $\mathbf{L}$ or a power of $\mathbf{R}$. So, it is easy to test whether the last matrix on the current level has been reached. If so, the first matrix on the next level is easily calculated. We get the following (non-terminating) program which computes the successive values of $\mathbf{C}$.

$$\begin{array}{l} \mathbf{C} := \mathbf{I}; \\ \text{do}\ \ \mathbf{C}_{10}{=}0 \;\rightarrow\; \mathbf{C} \;:=\; \begin{pmatrix} 1 & 0 \\ \mathbf{C}_{01}{+}1 & 1 \end{pmatrix} \\ \Box\ \ \mathbf{C}_{10}{\neq}0 \;\rightarrow\; j := \left\lfloor \dfrac{\mathbf{C}_{01}+\mathbf{C}_{11}-1}{\mathbf{C}_{00}+\mathbf{C}_{10}} \right\rfloor \;;\; \mathbf{C} \;:=\; \mathbf{C} \times \begin{pmatrix} 2j+1 & 1 \\ -1 & 0 \end{pmatrix} \\ \text{od} \end{array}$$

As remarked earlier, we immediately get an enumeration of the rationals as displayed in the Calkin-Wilf tree and as displayed in the Stern-Brocot tree — as each matrix is enumerated, simply premultiply by $(1\ 1)$ or postmultiply by $\binom{1}{1}$, respectively.

Optimisations of the algorithm are possible. For example, since for arbitrary z,

$$\begin{pmatrix} z & 1 \\ -1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \;=\; \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad ,$$

the assignment

$$\mathbf{C} \;:=\; \mathbf{C} \times \begin{pmatrix} 2j+1 & 1 \\ -1 & 0 \end{pmatrix}$$

replaces the second column of $\mathbf{C}$ by its first column. So, at each iteration, only two values of the matrix $\mathbf{C}$ need to be recomputed. Further details of such optimisations are discussed by the authors elsewhere [FB07].

2.2 Discussion

This paper was motivated by reading two publications, [GKP89] and [GLB06]. Gibbons, Lester and Bird [GLB06] show how to enumerate the elements of the Calkin-Wilf tree, but claim that "it is not at all obvious how to do this for the Stern-Brocot tree". Specifically, they say:

> 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 [New03]. 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.

In this paper, we have shown that it is obvious how to enumerate the rationals in Stern-Brocot order. Our derivation also demonstrates the duality between the Calkin-Wilf and Stern-Brocot trees.

Gibbons, Lester and Bird's goal seems to have been to show how to implement in the functional programming language Haskell the various constructions – the construction of the tree structures and the Calkin-Wilf-Newman enumeration of the rationals. In doing so, they repeat the existing mathematical presentations of the algorithms as given in [GKP89, CW00, New03]. None of these publications properly exploits the abundance of symmetry in the tree structures. Moreover, both [GKP89] and [GLB06] mention a link between enumerating the rationals and Euclid's algorithm but almost as an afterthought, at the end of the discussion.

The fact that expressing the rationals in "lowest form" is essential to the avoidance of duplication in any enumeration immediately suggests the relevance of Euclid's algorithm. The key to our exposition is that Euclid's algorithm can be expressed in terms of matrix multiplications, where —significantly— the underlying matrices are invertible. Transposition and inversion of the matrices capture the symmetry properties in a precise, calculational framework. As a result, the bijection between the rationals and the tree elements is immediate and we do not need to give separate, inductive proofs for both tree structures. Also, the determination of the next element in an enumeration of the tree elements has been reduced to one unifying construction.

Acknowledgements Thanks go to Jeremy Gibbons for his comments on earlier drafts of this paper, and for help with TEX commands. Thanks also to our colleagues in the Nottingham Tuesday Morning Club for helping iron out omissions and ambiguities.

References

[AZ04] Martin Aigner and Günter Ziegler. *Proofs From The Book, 3rd Edition.* Springer-Verlag, 2004.

[CW00] Neil Calkin and Herbert S. Wilf. Recounting the rationals. *The American Mathematical Monthly*, 107(4):360–363, 2000.

[Dij75] Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18:453–457, 1975.

[FB07] João Ferreira and Roland Backhouse. Programming Pearl, Enumerating the Rationals: Twice! (Available from the authors), 2007.

[GKP89] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics : a Foundation for Computer Science.* Addison-Wesley Publishing Company, 1989.

[GLB06] Jeremy Gibbons, David Lester, and Richard Bird. Enumerating the rationals. *Journal of Functional Programming*, 16(4), 2006.

[New03] Moshe Newman. Recounting the rationals, continued. *American Mathematical Monthly*, 110:642–643, 2003.