# On Euclid's Algorithm and Elementary Number Theory

Roland Backhouse, João F. Ferreira[1]

*School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, England*

**Abstract**

Algorithms can be used to prove and to discover new theorems. This paper shows how algorithmic skills in general, and the notion of invariance in particular, can be used to derive many results from Euclid's algorithm. We illustrate how to use the algorithm as a verification interface (i.e., how to verify theorems) and as a construction interface (i.e., how to investigate and derive new theorems).

The theorems that we verify are well-known and most of them are included in standard number-theory books. The new results concern distributivity properties of the greatest common divisor and a new algorithm for efficiently enumerating the positive rationals in two different ways. One way is known and is due to Moshe Newman. The second is new and corresponds to a deforestation of the Stern-Brocot tree of rationals. We show that both enumerations stem from the same simple algorithm. In this way, we construct a Stern-Brocot enumeration algorithm with the same time and space complexity as Newman's algorithm. A short review of the original papers by Stern and Brocot is also included.

*Key words:* number theory, calculational method, greatest common divisor, Euclid's algorithm, invariant, Eisenstein array, Eisenstein-Stern tree (aka Calkin-Wilf tree), Stern-Brocot tree, algorithm derivation, enumeration algorithm, rational number

*Email addresses:* `rcb@cs.nott.ac.uk` (Roland Backhouse), `joao@joaoff.com` (João F. Ferreira)

[1]Funded by Fundação para a Ciência e a Tecnologia (Portugal) under grant SFRH/BD/24269/2005

## 1. Introduction

An algorithm is a sequence of instructions that can be systematically executed in the solution of a given problem. Algorithms have been studied and developed since the beginning of civilisation, but, over the last 50 years, the unprecedented scale of programming problems and the consequent demands on precision and concision have made computer scientists hone their algorithmic problem-solving skills to a fine degree.

Even so, and although much of mathematics is algorithmic in nature, the skills needed to formulate and solve algorithmic problems do not form an integral part of contemporary mathematics education; also, the teaching of computer-related topics at pre-university level focuses on enabling students to be effective users of information technology, rather than equip them with the skills to develop new applications or to solve new problems.

A blatant example is the conventional treatment of Euclid's algorithm to compute the greatest common divisor (gcd) of two positive natural numbers, the oldest nontrivial algorithm that involves iteration and that has not been superseded by algebraic methods. (For a modern paraphrase of Euclid's original statement, see [1, pp. 335–336].) Most books on number theory include Euclid's algorithm, but rarely use the algorithm directly to reason about properties of numbers. Moreover, the presentation of the algorithm in such books has benefited little from the advances that have been made in our understanding of the basic principles of algorithm development. In an article such as this one, it is of course not the place to rewrite mathematics textbooks. Nevertheless, our goal in this paper is to demonstrate how a focus on algorithmic method can enrich and re-invigorate the teaching of mathematics. We use Euclid's algorithm to derive both old and well-known, and new and previously unknown, properties of the greatest common divisor and rational numbers. The leitmotiv is the notion of a loop invariant — how it can be used as as a verification interface (i.e., how to verify theorems) and as a construction interface (i.e., how to investigate and derive new theorems).

We begin the paper in section 2 with basic properties of the division relation and the construction of Euclid's algorithm from its formal specification. In contrast to standard presentations of the algorithm, which typically assume the existence of the gcd operator with specific algebraic properties, our derivation gives a constructive proof of the existence of an infimum operator in the division ordering of natural numbers.

The focus of section 3 is the systematic use of invariant properties of

Euclid's algorithm to verify known identities. Section 4, on the other hand, shows how to use the algorithm to derive new results related with the greatest common divisor: we calculate sufficient conditions for a natural-valued function[2] to distribute over the greatest common divisor, and we derive an efficient algorithm to enumerate the positive rational numbers in two different ways.

Although the identities in section 3 are well-known, we believe that our derivations improve considerably on standard presentations. One example is the proof that the greatest common divisor of two numbers is a linear combination of the numbers; by the simple device of introducing matrix arithmetic into Euclid's algorithm, it suffices to observe that matrix multiplication is associative in order to prove the theorem. This exemplifies the gains in our problem-solving skills that can be achieved by the right combination of precision and concision. The introduction of matrix arithmetic at this early stage was also what enabled us to derive a previously unknown algorithm to enumerate the rationals in so-called Stern-Brocot order (see section 4), which is the primary novel result (as opposed to method) in this paper.

Included in the appendix is a brief summary of the work of Stern and Brocot, the 19th century authors after whom the Stern-Brocot tree is named. It is interesting to review their work, particularly that of Brocot, because it is clearly motivated by practical, algorithmic problems. The review of Stern's paper is included in order to resolve recent misunderstandings about the origin of the Eisenstein-Stern and Stern-Brocot enumerations of the rationals.

## 2. Divisibility Theory

Division is one of the most important concepts in number theory. This section begins with a short, basic account of the division relation. We observe that division is a partial ordering on the natural numbers and pose the question whether the infimum, in the division ordering, of any pair of numbers exists. The algorithm we know as Euclid's gcd algorithm is then derived in order to give a positive (constructive) answer to this question.

[2] We call a function natural-valued if it has range the natural numbers.

*2.1. Division Relation*

The division relation, here denoted by an infix "\" symbol, is the relation on integers defined to be the converse of the "is-a-multiple-of" relation[3]:

$$[\ m\backslash n\ \equiv\ \langle\exists k : k\in\mathbb{Z} : n=k\times m\rangle\ ]\quad .$$

In words, an integer $m$ divides an integer $n$ (or $n$ is divisible by $m$) if there exists some integer $k$ such that $n=k\times m$. In that case, we say that $m$ is a divisor of $n$ and that $n$ is a multiple of $m$.

The division relation plays a prominent role in number theory. So, we start by presenting some of its basic properties and their relation to addition and multiplication. First, it is reflexive because multiplication has a unit (i.e., $m=1\times m$) and it is transitive, since multiplication is associative. It is also (almost) preserved by linear combination because multiplication distributes over addition:

$$(1)\qquad [\ k\backslash x \wedge k\backslash y\ \equiv\ k\backslash(x+a\times y) \wedge k\backslash y\ ]\qquad .$$

(We leave the reader to verify this law; take care to note the use of the distributivity of multiplication over addition in its proof.) Reflexivity and transitivity make division a *preorder* on the integers. It is not anti-symmetric but the numbers equivalent under the preordering are given by

$$[\ m\backslash n \wedge n\backslash m\ \equiv\ \mathsf{abs}.m=\mathsf{abs}.n\ ]\quad ,$$

where abs is the absolute value function. Each equivalence class thus consists of a natural number and its negation. If the division relation is restricted to natural numbers, division becomes anti-symmetric, since abs is the identity function on natural numbers. This means that, restricted to the natural numbers, division is a *partial* order with 0 as the greatest element and 1 as the smallest element.

---

[3]The square so-called "everywhere" brackets are used to indicate that a boolean statement is "everywhere" true. That is, the statement has the value true for all instantiations of its free variables. Such statements are often called "facts", or "laws", or "theorems".

When using the everywhere brackets, the domain of the free variables has to be made clear. This is particularly important here because sometimes the domain of a variable is the integers and sometimes it is the natural numbers. Usually, we rely on a convention for naming the variables, but sometimes we preface a law with a reminder of the domain.

*2.1.1. Infimum in the Division Ordering*

The first question that we consider is whether two arbitrary natural numbers $m$ and $n$ have an infimum in the division ordering. That is, can we solve the following equation[4]?

$$(2) \quad x:: \quad \langle \forall k \;::\; k\backslash m \wedge k\backslash n \;\equiv\; k\backslash x \rangle \;.$$

The answer is not immediately obvious because the division ordering is partial. (With respect to a total ordering, the infimum of two numbers is their minimum; it is thus equal to one of them and can be easily computed by a case analysis.)

If a solution to (2) exists, it is unique (because the division relation on natural numbers is reflexive and anti-symmetric). When it does have a solution, we denote it by $m\nabla n$. That is, provided it can be established that (2) has a solution,

$$(3) \quad [\; k\backslash m \wedge k\backslash n \;\equiv\; k\backslash(m\nabla n) \;] \;.$$

Because conjunction is idempotent,

$$[\; k\backslash m \wedge k\backslash m \;\equiv\; k\backslash m \;] \;.$$

That is, $m$ solves (2) when $m$ and $n$ are equal. Also, because $[\; k\backslash 0 \;]$,

$$[\; k\backslash m \wedge k\backslash 0 \;\equiv\; k\backslash m \;] \;.$$

That is, $m$ solves (2) when $n$ is 0. So, $m\nabla m$ exists as does $m\nabla 0$, and both equal $m$:

$$(4) \quad [\; m\nabla m \;=\; m\nabla 0 \;=\; m \;] \;.$$

Other properties that are easy to establish by exploiting the algebraic properties of conjunction are, first, $\nabla$ is symmetric (because conjunction is symmetric)

$$(5) \quad [\; m\nabla n \;=\; n\nabla m \;] \;,$$

[4] Recall that the domain of all variables is $\mathbb{N}$, the set of natural numbers. Note that we include 0 in $\mathbb{N}$.

and, second, $\nabla$ is associative (because conjunction is associative)

(6) $\quad [\ (m\nabla n)\nabla p = m\nabla(n\nabla p)\ ]$ .

Note that we choose infix notation for $\nabla$, since it allows us to write $m\nabla n\nabla p$ without having to choose between $(m\nabla n)\nabla p$ or $m\nabla(n\nabla p)$.

The final property of $\nabla$ that we deduce from (3) is obtained by exploiting (1), with $x$ and $y$ replaced by $m$ and $n$, respectively :

(7) $\quad [\ (m+a\times n)\nabla n \ = \ m\nabla n\ ]$ .

*2.2. Constructing Euclid's Algorithm*

At this stage in our analysis, properties (5), (6) and (7) assume that equation (2) has a solution in the appropriate cases. For instance, (5) means that, if (2) has a solution for certain natural numbers $m$ and $n$, it also has a solution when the values of $m$ and $n$ are interchanged.

In view of properties (4) and (5), it remains to show that (2) has a solution when both $m$ and $n$ are strictly positive and unequal. We do this by providing an algorithm that computes the solution. Equation (2) does not directly suggest any algorithm, but the germ of an algorithm is suggested by observing that it is equivalent to

(8) $\quad x,y:: \quad x=y \ \wedge\ \langle\forall k::\ k\backslash m \wedge k\backslash n \ \equiv\ k\backslash x \wedge k\backslash y\rangle$ .

This new shape strongly suggests an algorithm that, initially, establishes the truth of

$$\langle\forall k::\ k\backslash m \wedge k\backslash n \ \equiv\ k\backslash x \wedge k\backslash y\rangle$$

—which is trivially achieved by the assignment $x,y := m,n$— and then, reduces $x$ and $y$ in such a way that the property is kept invariant whilst making progress to a state satisfying $x=y$. When such a state is reached, we have found a solution to the equation (8), and the value of $x$ (or $y$ since they are equal) is a solution of (2). Thus, the structure of the algorithm we are trying to develop is as follows:

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

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

$\{$ **Invariant**: $\ \langle\forall k::\ k\backslash m \wedge k\backslash n \ \equiv\ k\backslash x \wedge k\backslash y\rangle\ \}$

$\mathsf{do}\ x \neq y \rightarrow x\,,y\ :=\ A\,,B$

$\mathsf{od}$

$\{\ x=y\ \ \wedge\ \ \langle\forall k::\ \ k\backslash m \wedge k\backslash n\ \ \equiv\ \ k\backslash x \wedge k\backslash y\rangle\ \ \}\ \ .$

Now we only have to define $A$ and $B$ in such a way that the assignment in the loop body leads to a state where $x=y$ is satisfied while maintaining the invariant. Exploiting the transitivity of equality, the invariant is maintained by choosing $A$ and $B$ so that

(9) $\langle\forall k::\ \ k\backslash x \wedge k\backslash y\ \ \equiv\ \ k\backslash A \wedge k\backslash B\rangle\ \ .$

To ensure that we are making progress towards the termination condition, we have to define a *bound function*, which is a natural-valued function of the variables $x$ and $y$ that measures the size of the problem to be solved. A guarantee that the value of such a bound function is always decreased at each iteration is a guarantee that the number of times the loop body is executed is at most the initial value of the bound function. The definition of the bound function depends on the assignments we choose for $A$ and $B$.

At this point, we need to exploit properties specific to division. (Refer back to section 2.1 for a discussion of some of the properties.) Inspecting the shape of (9), we see that it is similar to the shape of property (1). This suggests that we can use (1), and in fact, considering this property, we have the corollary:

(10) $[\ \ k\backslash x \wedge k\backslash y\ \ \equiv\ \ k\backslash(x-y) \wedge k\backslash y\ \ ]\ \ .$

The relevance of this corollary is that our invariant is preserved by the assignment $x\ :=\ x-y$ (leaving the value of $y$ unchanged). (Compare (10) with (9).) Note that this also reduces the value of $x$ when $y$ is positive. This suggests that we strengthen the invariant by requiring that $x$ and $y$ remain positive; the assignment $x\ :=\ x-y$ is executed when $x$ is greater than $y$ and, symmetrically, the assignment $y\ :=\ y-x$ is executed when $y$ is greater than $x$. As bound function we can take $x+y$. The algorithm becomes

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

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

$\{$ **Invariant**: $0<x\ \wedge\ 0<y\ \wedge\ \langle\forall k::\ \ k\backslash m \wedge k\backslash n\ \ \equiv\ \ k\backslash x \wedge k\backslash y\rangle$

**Bound function**: $x+y\ \ \}$

$$
\begin{array}{l}
\textsf{do}\ x \neq y \ \rightarrow \\
\qquad \textsf{if}\ \ y < x \rightarrow x := x-y \\
\qquad \Box\ \ x < y \rightarrow y := y-x \\
\qquad \textsf{fi} \\
\textsf{od} \\
\{\ 0<x \ \wedge\ 0<y \ \wedge\ x=y \ \wedge\ \langle\forall k{::}\ k\backslash m \wedge k\backslash n \ \equiv\ k\backslash x \wedge k\backslash y\rangle\ \}\ .
\end{array}
$$

(We leave the reader to perform the standard steps used to verify the correctness of the algorithm.) Finally, since

$$(x < y \vee y < x) \ \equiv\ x \neq y \quad ,$$

we can safely remove the outer guard and simplify the algorithm, as shown below.

$$
\begin{array}{l}
\{\ 0 < m \ \wedge\ 0 < n\ \} \\
x\,,y := m\,,n\ ; \\
\{\ \textbf{Invariant}:\quad 0<x \ \wedge\ 0<y \ \wedge\ \langle\forall k{::}\ k\backslash m \wedge k\backslash n \ \equiv\ k\backslash x \wedge k\backslash y\rangle \\
\quad \textbf{Bound function}: x+y\ \} \\
\textsf{do}\ \ y < x \rightarrow x := x-y \\
\Box\ \ x < y \rightarrow y := y-x \\
\textsf{od} \\
\{\ 0<x \ \wedge\ 0<y \ \wedge\ x=y \ \wedge\ \langle\forall k{::}\ k\backslash m \wedge k\backslash n \ \equiv\ k\backslash x \wedge k\backslash y\rangle\ \}\ .
\end{array}
$$

The algorithm that we have constructed is Euclid's algorithm for computing the greatest common divisor of two positive natural numbers, the oldest non-trivial algorithm that has survived to the present day! (Please note that our formulation of the algorithm differs from Euclid's original version and from most versions found in number-theory books. While they use the property $[\ m\triangledown n \ =\ n\triangledown(m\,\textsf{rem}\,n)\ ]$, we use (10), i.e., $[\ m\triangledown n \ =\ (m-n)\triangledown n\ ]$. For an encyclopedic account of Euclid's algorithm, we recommend [1, p. 334].)

*2.3. Greatest Common Divisor*

In section 2.1.1, we described the problem we were tackling as establishing that the infimum of two natural numbers under the division ordering always exists; it was only at the end of the section that we announced that the algorithm we had derived is an algorithm for determining the greatest common divisor. This was done deliberately in order to avoid the confusion that can —and does— occur when using the words "greatest common divisor". In this section, we clarify the issue in some detail.

Confusion and ambiguity occur when a set can be ordered in two different ways. The natural numbers can be ordered by the usual size ordering (denoted by the symbol $\leq$), but they can also be ordered by the division relation. When the ordering is not made explicit (for instance, when referring to the "least" or "greatest" of a set of numbers), we might normally understand the size ordering, but the division ordering might be meant, depending on the context.

In words, the infimum of two values in a partial ordering —if it exists— is the largest value (with respect to the ordering) that is at most both values (with respect to the ordering). The terminology "greatest lower bound" is often used instead of "infimum". Of course, "greatest" here is with respect to the partial ordering in question. Thus, the infimum (or greatest lower bound) of two numbers with respect to the division ordering —if it exists— is the largest number with respect to the division ordering that divides both of the numbers. Since, *for strictly positive numbers*, "largest with respect to the division ordering" implies "largest with respect to the size ordering" (equally, the division relation, restricted to strictly positive numbers, is a subset of the $\leq$ relation), the "largest number with respect to the *division* ordering that divides both of the numbers" is the same, *for strictly positive numbers*, as the "largest number with respect to the *size* ordering that divides both of the numbers". Both these expressions may thus be abbreviated to the "greatest common divisor" of the numbers, with no problems caused by the ambiguity in the meaning of "greatest" — *when the numbers are strictly positive*. Ambiguity does occur, however, when the number 0 is included, because 0 is the *largest* number with respect to the division ordering, but the *smallest* number with respect to the size ordering. If "greatest" is taken to mean with respect to the division ordering on numbers, the greatest common divisor of 0 and 0 is simply 0. (This is a consequence of the simple fact that, for all $x$ in a partially ordered set, the infimum of $x$ and $x$ exists and is equal to $x$.) If, however, "greatest" is taken to mean with respect to the size ordering, there is no greatest common divisor of 0 and 0. The knock-on effects of the latter are immense — for example the gcd operator is no longer idempotent, since $0\triangledown 0$ is undefined, and it is no longer associative, since, for positive $m$, $(m\nabla 0)\nabla 0$ is well-defined whilst $m\nabla(0\nabla 0)$ is not.

Concrete evidence of the confusion in the standard mathematics literature is easy to find. We looked up the definition of greatest common divisor in three undergraduate mathematics texts, and found three non-equivalent definitions. The texts were chosen simply on the basis that they were owned by a student of mathematics (rcb's son), all having been recommended by mathematics lecturers. The first [2, p. 30] defines "greatest" to mean with respect to the divides relation (as it should be defined); the second [3, p. 21, def. 2.2] defines "greatest" to mean with respect to the $\leq$ relation (and requires that at least one of the numbers be non-zero). The third text [4, p. 78] excludes zero altogether, defining the greatest common divisor of strictly positive numbers as the generator of all linear combinations of the given numbers; the accompanying explanation (in words) of the terminology replaces "greatest" by "largest" but does not clarify with respect to which ordering the "largest" is to be determined.

Now that we know that $\nabla$ is the greatest common divisor, we could change the operator to *gcd*, i.e., replace $m\nabla n$ by $m\ gcd\ n$. However, we stick to the "$\nabla$" notation because it makes the formulae shorter, and, so, easier to read. We also use "$\triangle$" to denote the least common multiple operator. To remember which is which, just remember that infima (*lower* bounds) are indicated by *downward*-pointing symbols (eg. $\downarrow$ for minimum, and $\vee$ for disjunction) and suprema (*upper* bounds) by *upward*-pointing symbols.

## 3. Euclid's Algorithm as a Verification Interface

In this section we show how algorithms and the notion of invariance can be used to prove theorems. In particular, we show that the exploitation of Euclid's algorithm makes proofs related with the greatest common divisor simple and more systematic than the traditional ones.

There is a clear pattern in all our calculations: everytime we need to prove a new theorem involving $\nabla$, we construct an invariant that is valid initially (with $x\,,y\ :=\ m\,,n$) and that corresponds to the theorem to be proved upon termination (with $x=y=m\nabla n$). (Alternatively, we can construct an invariant that is valid on termination (with $x=y=m\nabla n$) and whose initial value corresponds to the theorem to be proved. The invariant in section 3.3 is such an example.) Then, it remains to prove that the chosen invariant is valid after each iteration of the repeatable statement.

We start with a minor change in the invariant that allows us to prove some well-known properties. Then, we explore how the shape of the theorems to be proved determine the shape of the invariant. We also show how to prove a geometrical property of $\nabla$.

*3.1. Exploring the invariant*

The invariant that we use in section 2.2 rests on the validity of the theorem

$$[\ \ k\backslash m \wedge k\backslash n \ \ \equiv\ \ k\backslash(m{-}n) \wedge k\backslash n\ \ ] \quad .$$

But, as Van Gasteren observed in [5, Chapter 11], we can use the more general and equally valid theorem

$$[\ \ k\backslash(c{\times}m) \wedge k\backslash(c{\times}n) \ \ \equiv\ \ k\backslash(c\times(m{-}n)) \wedge k\backslash(c{\times}n)\ \ ]$$

to conclude that the following property is an invariant of Euclid's algorithm:

$$\langle\forall k,c{::}\ \ k\backslash(c{\times}m) \wedge k\backslash(c{\times}n) \ \ \equiv\ \ k\backslash(c{\times}x) \wedge k\backslash(c{\times}y)\rangle \quad .$$

In particular, the property is true on termination of the algorithm, at which point $x$ and $y$ both equal $m\nabla n$. That is, for all $m$ and $n$, such that $0<m$ and $0<n$,

$$(11)\quad [\ \ k\backslash(c{\times}m) \wedge k\backslash(c{\times}n) \ \ \equiv\ \ k\backslash(c\times(m\nabla n))\ \ ] \quad .$$

In addition, theorem (11) holds when $m<0$, since

$$[\ \ (-m)\nabla n = m\nabla n\ \ ] \ \ \wedge\ \ [\ \ k\backslash(c{\times}(-m)) \ \ \equiv\ \ k\backslash(c{\times}m)\ \ ] \quad ,$$

and it holds when $m$ equals 0, since $[\ k\backslash 0\ ]$. Hence, using the symmetry between $m$ and $n$ we conclude that (11) is indeed valid for all integers $m$ and $n$. (In Van Gasteren's presentation, this theorem only holds for all $(m,n)\neq(0,0)$.)

Theorem (11) can be used to prove a number of properties of the greatest common divisor. If, for instance, we replace $k$ by $m$, we have

$$[\ \ m\backslash(c{\times}n) \ \ \equiv\ \ m\backslash(c\times(m\nabla n))\ \ ] \quad ,$$

and, as a consequence, we also have

$$(12)\quad [\ \ (m\backslash(c{\times}n) \ \ \equiv\ \ m\backslash c) \Leftarrow m\nabla n = 1\ \ ] \quad .$$

More commonly, (12) is formulated as the weaker

$$[\ m\backslash c \ \Leftarrow\ m\nabla n = 1 \ \wedge\ m\backslash(c\times n)\ ] \quad ,$$

and is known as Euclid's Lemma. Another significant property is

(13) $[\ k \setminus (c \times (m\nabla n)) \quad \equiv \quad k \setminus ((c\times m)\nabla(c\times n))\ ] \quad ,$

which can be proved as:

$$\begin{array}{ll} & k \setminus (c \times (m\nabla n)) \\ = & \quad \{ \quad (11) \quad \} \\ & k \setminus (c\times m) \ \wedge\ k \setminus (c\times n) \\ = & \quad \{ \quad (3) \quad \} \\ & k \setminus ((c\times m)\nabla(c\times n)) \quad . \end{array}$$

From (13) and by observing that

$[\ 0 \leq m\nabla n\ ]$ , and

$[\ \langle\forall k::\ k\backslash m \equiv k\backslash n\rangle \ \equiv\ \mathsf{abs}.m = \mathsf{abs}.n\ ] \quad ,$

we conclude

(14) $[\ (c\times m)\nabla(c\times n) = (\mathsf{abs}.c) \times (m\nabla n)\ ] \quad .$

Property (14) states that multiplication by a natural number distributes over $\nabla$. It is an important property that can be used to simplify arguments where both multiplication and the greatest common divisor are involved. An example is Van Gasteren's proof of the theorem

(15) $[\ (m\times p)\nabla n = m\nabla n \ \Leftarrow\ p\nabla n = 1\ ] \quad ,$

which is as follows:

$$\begin{array}{ll} & m\nabla n \\ = & \quad \{ \quad p\nabla n = 1 \text{ and 1 is the unit of multiplication} \quad \} \\ & (m\times(p\nabla n))\nabla n \\ = & \quad \{ \quad (14) \quad \} \\ & (m\times p)\ \nabla\ (m\times n)\ \nabla\ n \\ = & \quad \{ \quad (m\times n)\nabla n = n \quad \} \\ & (m\times p)\nabla n \quad . \end{array}$$

*3.2. $\nabla$ on the left side*

In the previous sections, we have derived a number of properties of the $\nabla$ operator. However, where the divides relation is involved, the operator always occurs on the right side of the relation. (For examples, see (3) and (13).) Now we consider properties where the operator is on the left side of a divides relation. Our goal is to show that

(16) $[\ (m\nabla n)\setminus k \ \equiv\ \langle\exists a,b{::}\ k\ =\ m{\times}a+n{\times}b\rangle\ ]$ ,

where the range of $a$ and $b$ is the integers.

Of course, if (16) is indeed true, then it is also true when $k$ equals $m\nabla n$. That is, a consequence of (16) is

(17) $[\ \langle\exists a,b{::}\ m\nabla n\ =\ m{\times}a+n{\times}b\rangle\ ]$ .

In words, $m\nabla n$ is a linear combination of $m$ and $n$. For example,

$$3\nabla 5\ =\ 1\ =\ 3{\times}2-5{\times}1\ =\ 5{\times}2-3{\times}3\ .$$

Vice-versa, if (17) is indeed true then (16) is a consequence. (The crucial fact is that multiplication distributes through addition.) It thus suffices to prove (17).

We can establish (17) by constructing such a linear combination for given values of $m$ and $n$.

When $n$ is 0, we have

$$m\nabla 0\ =\ m\ =\ m{\times}1+0{\times}1\ .$$

(The multiple of 0 is arbitrarily chosen to be 1.)

When both $m$ and $n$ are non-zero, we need to augment Euclid's algorithm with a computation of the coefficients. The most effective way to establish the property is to establish that $x$ and $y$ are linear combinations of $m$ and $n$ is an invariant of the algorithm; this is best expressed using matrix arithmetic.

In the algorithm below, the assignments to $x$ and $y$ have been replaced by equivalent assignments to the vector $(x\ y)$. Also, an additional variable **C**, whose value is a 2×2 matrix of integers has been introduced into the program. Specifically, **I**, **A** and **B** are 2×2 matrices; **I** is the identity matrix $\begin{pmatrix}1&0\\0&1\end{pmatrix}$ , **A** is the matrix $\begin{pmatrix}1&0\\-1&1\end{pmatrix}$ and **B** is the matrix $\begin{pmatrix}1&-1\\0&1\end{pmatrix}$. (The assignment $(x\ y) := (x\ y){\times}\mathbf{A}$ is equivalent to $x\,,y\ :=\ x{-}y\,,y$, as can be easily checked.)

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

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

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

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

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

$\mathsf{od}$

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

The invariant shows only the relation between the vectors $(x\ y)$ and $(m\ n)$; in words, $(x\ y)$ is a multiple of $(m\ n)$.

It is straightforward to verify that the invariant is established by the initialising assignment, and maintained by the loop body. Crucial to the proof that it is maintained by the loop body is that multiplication (here of matrices) is associative. Had we expressed the assignments to $\mathbf{C}$ in terms of its four elements, verifying that the invariant is maintained by the loop body would have amounted to giving in detail the proof that matrix multiplication is associative. This is a pointless duplication of effort, avoiding which fully justifies the excursion into matrix arithmetic.

(An exercise for the reader is to express the property that $m$ and $n$ are linear combinations of $x$ and $y$. The solution involves observing that $\mathbf{A}$ and $\mathbf{B}$ are invertible. This will be exploited in section 4.2.)

*3.3. A geometrical property*

In this section, we prove that in a Cartesian coordinate system, $m\nabla n$ can be interpreted as the number of points with integral coordinates on the straight line joining the points $(0,0)$ and $(m,n)$, excluding $(0,0)$. Formally, with dummies $s$ and $t$ ranging over integers, we prove:

$$(18)\quad [\ \langle\Sigma s,t\ :\ m{\times}t = n{\times}s\ \wedge\ 0 < s \leq m\ \wedge\ 0 < t \leq n\ :\ 1\rangle\ =\ m\nabla n\ ]\quad .$$

First, we observe that

$$(0 < s \leq m\ \ \equiv\ \ 0 < t \leq n)\ \Leftarrow\ m{\times}t = n{\times}s\quad ,$$

since

$0 < t \leq n$

$$= \quad \{ \quad m\times t = n\times s \quad \}$$

$$0 < n\times s \leq n\times m$$

$$= \quad \{ \quad 0 < n, \text{ cancellation} \quad \}$$

$$0 < s \leq m \quad .$$

This means that we can simplify (18) to

(19) $[\ \langle \Sigma s,t\ :\ m\times t = n\times s\ \wedge\ 0 < t \leq n\ :\ 1\rangle = m\nabla n\ ]$ .

In order to use Euclid's algorithm, we need to find an invariant that allows us to conclude (19). If we use as invariant

(20) $\langle \Sigma s,t\ :\ x\times t = y\times s\ \wedge\ 0 < t \leq y\ :\ 1\rangle = x\nabla y$ ,

its initial value is the property that we want to prove:

$$\langle \Sigma s,t\ :\ m\times t = n\times s\ \wedge\ 0 < t \leq n\ :\ 1\rangle = m\nabla n \quad .$$

Its value upon termination is

$$\langle \Sigma s,t\ :\ (m\nabla n)\times t = (m\nabla n)\times s\ \wedge\ 0 < t \leq m\nabla n\ :\ 1\rangle = (m\nabla n)\nabla(m\nabla n) \quad ,$$

which is equivalent (by cancellation of multiplication and idempotence of $\nabla$) to

$$\langle \Sigma s,t\ :\ t = s\ \wedge\ 0 < t \leq m\nabla n\ :\ 1\rangle = m\nabla n \quad .$$

It is easy to see that the invariant reduces to true on termination (because the sum on the left equals $m\nabla n$), making its initial value also true.

It is also easy to see that the righthand side of the invariant is unnecessary as it is the same initially and on termination. This motivates the generalisation of the concept "invariant". "Invariants" in the literature are always boolean-valued functions of the program variables, but we see no reason why "invariants" shouldn't be of any type: for us, an *invariant* of a loop is simply a function of the program variables whose value is unchanged by execution of the loop body[5]. In this case, the value is a natural number. Therefore, we

[5]Some caution is needed here because our more general use of the word "invariant" does not completely coincide with its standard usage for boolean-valued functions. The standard meaning of an invariant of a statement $S$ is a boolean-valued function of the program variables which, in the case that the function evaluates to true, remains true after execution of $S$. Our usage requires that, if the function evaluates to false before execution of $S$, it continues to evaluate to false after executing $S$.

can simplify (20) and use as invariant

(21) $\langle \Sigma s,t \ : \ x\times t = y\times s \ \wedge \ 0 < t \leq y \ : \ 1\rangle$ .

Its value on termination is

$$\langle \Sigma s,t \ : \ (m\nabla n)\times t = (m\nabla n)\times s \wedge 0 < t \leq m\nabla n \ : \ 1\rangle \quad ,$$

which is equivalent to

$$\langle \Sigma s,t \ : \ t = s \wedge 0 < t \leq m\nabla n \ : \ 1\rangle \quad .$$

As said above, this sum equals $m\nabla n$.

Now, since the invariant (21) equals the lefthand side of (19) for the initial values of $x$ and $y$, we only have to check if it remains constant after each iteration. This means that we have to prove (for $y < x \wedge 0 < y$):

$$\begin{array}{ll} & \langle \Sigma s,t \ : \ x\times t = y\times s \wedge 0 < t \leq y \ : \ 1\rangle \\ = & \langle \Sigma s,t \ : \ (x-y)\times t = y\times s \wedge 0 < t \leq y \ : \ 1\rangle \ , \end{array}$$

which can be rewritten, for positive $x$ and $y$, as:

$$\begin{array}{ll} & \langle \Sigma s,t \ : \ (x+y)\times t = y\times s \wedge 0 < t \leq y \ : \ 1\rangle \\ = & \langle \Sigma s,t \ : \ x\times t = y\times s \wedge 0 < t \leq y \ : \ 1\rangle \ . \end{array}$$

The proof is as follows:

$$\begin{array}{ll} & \langle \Sigma s,t \ : \ (x+y)\times t = y\times s \wedge 0 < t \leq y \ : \ 1\rangle \\ = & \qquad \{ \qquad \text{distributivity and cancellation} \quad \} \\ & \langle \Sigma s,t \ : \ x\times t = y\times (s-t) \wedge 0 < t \leq y \ : \ 1\rangle \\ = & \qquad \{ \qquad \text{range translation: } s := s+t \quad \} \\ & \langle \Sigma s,t \ : \ x\times t = y\times s \wedge 0 < t \leq y \ : \ 1\rangle \ . \end{array}$$

Note that the simplification done in (19) allows us to apply the range translation rule in the last step without having to relate the range of variable $s$ with the possible values for variable $t$. Property (18) also holds when $m = 0$ or when $n = 0$. The proof is left to the reader.

## 4. Euclid's Algorithm as a Construction Interface

In this section we show how to use Euclid's algorithm to derive new theorems related with the greatest common divisor. We start by calculating reasonable sufficient conditions for a natural-valued function to distribute over the greatest common divisor. We also derive an efficient algorithm for enumerating the positive rational numbers in two different ways.

### *4.1. Distributivity properties*

In addition to multiplication by a natural number, there are other functions that distribute over $\triangledown$. The goal of this subsection is to determine reasonable sufficient conditions for a natural-valued function $f$ to distribute over $\triangledown$, i.e., for the following property to hold:

(22) $[\ f.(m\triangledown n) = f.m \triangledown f.n\ ]$ .

For simplicity's sake, we restrict all variables to natural numbers. This implies that the domain of $f$ is also restricted to the natural numbers.

We explore (22) by identifying invariants of Euclid's algorithm involving the function $f$. To determine an appropriate loop invariant, we take the right-hand side of (22) and calculate:

$$
\begin{array}{ll}
 & f.m \triangledown f.n \\
= & \qquad \{ \quad \text{the initial values of } x \text{ and } y \text{ are } m \text{ and } n\text{, respectively} \quad \} \\
 & f.x \triangledown f.y \\
= & \qquad \{ \quad \text{suppose that } f.x \triangledown f.y \text{ is invariant;} \\
 & \qquad \quad \ \ \text{on termination: } x = m\triangledown n \wedge y = m\triangledown n \quad \} \\
 & f.(m\triangledown n) \triangledown f.(m\triangledown n) \\
= & \qquad \{ \quad \triangledown \text{ is idempotent} \quad \} \\
 & f.(m\triangledown n) \ .
\end{array}
$$

Property (22) is thus established under the assumption that $f.x\triangledown f.y$ is an invariant of the loop body. (Please note that this invariant is of the more general form introduced in section 3.3.)

The next step is to determine what condition on $f$ guarantees that $f.x\triangledown f.y$ is indeed invariant. Noting the symmetry in the loop body between $x$ and $y$, the condition is easily calculated to be

$$[\ f.(x-y) \bigtriangledown f.y \ = \ f.x \bigtriangledown f.y \ \Leftarrow \ 0 < y < x \ ] \qquad .$$

Equivalently, by the rule of range translation ($x \ := \ x+y$), the condition can be written as

$$(23) \quad [\ f.x \bigtriangledown f.y \ = \ f.(x+y) \bigtriangledown f.y \ \Leftarrow \ 0 < x \wedge 0 < y \ ] \qquad .$$

Formally, this means that

$$\text{“}\, f \text{ distributes over } \bigtriangledown \text{”} \ \Leftarrow \ (23) \quad .$$

Incidentally, the converse of this property is also valid:

$$(23) \ \Leftarrow \ \text{“}\, f \text{ distributes over } \bigtriangledown \text{”} \quad .$$

The simple calculation proceeds as follows:

$f.(x+y) \bigtriangledown f.y$

$=$ $\{$ $f$ distributes over $\bigtriangledown$ $\}$

$f.((x+y) \bigtriangledown y)$

$=$ $\{$ (7) $\}$

$f.(x \bigtriangledown y)$

$=$ $\{$ $f$ distributes over $\bigtriangledown$ $\}$

$f.x \bigtriangledown f.y$ .

By mutual implication we conclude that

$$\text{“}\, f \text{ distributes over } \bigtriangledown \text{”} \ \equiv \ (23) \quad .$$

We have now reached a point where we can determine if a function distributes over $\bigtriangledown$. However, since (23) still has two occurrences of $\bigtriangledown$, we want to refine it into simpler properties. Towards that end we turn our attention to the condition

$$f.x \bigtriangledown f.y \ = \ f.(x+y) \bigtriangledown f.y \quad ,$$

and we explore simple ways of guaranteeing that it is everywhere true. For instance, it is immediately obvious that any function that distributes over addition distributes over $\bigtriangledown$. (Note that multiplication by a natural number is such a function.) The proof is very simple:

$f.(x{+}y) \bigtriangledown f.y$

$= \quad \{ \quad f$ distributes over addition $\}$

$(f.x{+}f.y) \bigtriangledown f.y$

$= \quad \{ \quad (7) \quad \}$

$f.x \bigtriangledown f.y \quad .$

In view of properties (7) and (15), we formulate the following lemma, which is a more general requirement:

**Lemma 24.** All functions $f$ that satisfy

$$\langle \forall x, y{::} \quad \langle \exists a, b \ : \ a \bigtriangledown f.y = 1 \ : \ f.(x{+}y) \ = \ a \times f.x \ + \ b \times f.y \rangle\rangle$$

distribute over $\bigtriangledown$.

**Proof**

$f.(x{+}y) \bigtriangledown f.y$

$= \quad \{ \quad f.(x{+}y) \ = \ a \times f.x \ + \ b \times f.y \quad \}$

$(a \times f.x \ + \ b \times f.y) \bigtriangledown f.y$

$= \quad \{ \quad (7) \quad \}$

$(a \times f.x) \bigtriangledown f.y$

$= \quad \{ \quad a \bigtriangledown f.y = 1$ and $(15) \quad \}$

$f.x \bigtriangledown f.y \quad .$

□

Note that since the discussion above is based on Euclid's algorithm, it only applies to positive arguments. We now investigate the case where $m$ or $n$ is 0. We have, for $m = 0$ :

$f.(0 \bigtriangledown n) \ = \ f.0 \bigtriangledown f.n$

$= \quad \{ \quad [ \ 0 \bigtriangledown m = m \ ] \quad \}$

$f.n \ = \ f.0 \bigtriangledown f.n$

$= \quad \{ \quad [ \ a \backslash b \equiv a = b \bigtriangledown a \ ] \quad \}$

$$
\begin{array}{ll}
& f.n \setminus f.0 \\
= & \{ \quad \text{definition of divides relation} \quad \} \\
& \langle \exists k \;:\; k \in \mathbb{Z} \;:\; f.0 \;=\; k \times f.n \rangle \\
\Leftarrow & \{ \quad \text{obvious possibilities for } f.0 \text{ or for } f.n \quad \} \\
& f.0 = 0 \;\vee\; f.n = 1 \;\vee\; f.n = f.0 \quad .
\end{array}
$$

Hence, using the symmetry between $m$ and $n$ we have, for $m=0$ or $n=0$:

$$(25) \quad f.(m \triangledown n) \;=\; f.m \triangledown f.n \quad \Leftarrow \quad f.0=0 \;\vee\; f.n=1 \;\vee\; f.n=f.0 \quad .$$

The conclusion is that we can use (25) and lemma 24 to prove that a natural-valued function with domain $\mathbb{N}$ distributes over $\triangledown$.

*Example 0: the Fibonacci function*

In [6], Edsger Dijkstra proves that the Fibonacci function distributes over $\triangledown$ . He does not use lemma 24 explicitly, but he constructs the property

$$(26) \quad \mathsf{fib}.(x{+}y) \;=\; \mathsf{fib}.(y{-}1) \times \mathsf{fib}.x \;+\; \mathsf{fib}.(x{+}1) \times \mathsf{fib}.y \quad ,$$

and then, using the lemma

$$\mathsf{fib}.y \triangledown \mathsf{fib}.(y{-}1) \;=\; 1 \quad ,$$

he concludes the proof. His calculation is the same as that in the proof of lemma 24 but for particular values of $a$ and $b$ and with $f$ replaced by $\mathsf{fib}$. Incidentally, if we don't want to construct property (26) we can easily verify it using induction — more details are given in [7].

An interesting application of this distributivity property is to prove that for any positive $k$, every $k$th number in the Fibonacci sequence is a multiple of the $k$th number in the Fibonacci sequence. More formally, the goal is to prove

$$\mathsf{fib}.(n{\times}k) \text{ is a multiple of } \mathsf{fib}.k \quad ,$$

for positive $k$ and natural $n$. A concise proof is:

$$
\begin{array}{ll}
& \mathsf{fib}.(n{\times}k) \text{ is a multiple of } \mathsf{fib}.k \\
= & \{ \quad \text{definition} \quad \}
\end{array}
$$

$$\begin{array}{ll}
& \mathsf{fib}.k \setminus \mathsf{fib}.(n\times k) \\
= & \{ \quad [\ a\backslash b \equiv a\triangledown b = a\ ]\ , \\
& \quad \text{with } a := \mathsf{fib}.k \text{ and } b := \mathsf{fib}.(n\times k) \quad \} \\
& \mathsf{fib}.k \triangledown \mathsf{fib}.(n\times k)\ =\ \mathsf{fib}.k \\
= & \{ \quad \mathsf{fib} \text{ distributes over } \triangledown \quad \} \\
& \mathsf{fib}.(k\triangledown(n\times k)) = \mathsf{fib}.k \\
= & \{ \quad k\triangledown(n\times k) = k \text{ and reflexivity} \quad \} \\
& \mathsf{true}\ \ .
\end{array}$$

*Example 1: the Mersenne function*

We now prove that, for all integers $k$ and $m$ such that $0 < k^m$, the function $f$ defined as

$$f.m\ =\ k^m - 1$$

distributes over $\triangledown$.

First, we observe that $f.0 = 0$. (Recall the discussion of (25).) Next, we use lemma 24. This means that we need to find integers $a$ and $b$, such that

$$k^{m+n}-1\ =\ a\times(k^m-1) + b\times(k^n-1) \quad \wedge \quad a\triangledown(k^n-1) = 1 \quad .$$

The most obvious instantiations for $a$ are 1, $k^n$ and $k^n-2$. (That two consecutive numbers are coprime follows from (7).) Choosing $a=1$, we calculate $b$:

$$\begin{array}{ll}
& k^{m+n}-1\ =\ (k^m-1) + b\times(k^n-1) \\
= & \{ \quad \text{arithmetic} \quad \} \\
& k^{m+n}-k^m\ =\ b\times(k^n-1) \\
= & \{ \quad \text{multiplication distributes over addition} \quad \} \\
& k^m\times(k^n-1)\ =\ b\times(k^n-1) \\
\Leftarrow & \{ \quad \text{Leibniz} \quad \} \\
& k^m = b\ \ .
\end{array}$$

We thus have

$$k^{m+n}-1\ =\ 1\times(k^m-1) + k^m\times(k^n-1)\ \wedge\ 1\triangledown(k^n-1) = 1 \quad ,$$

and we use lemma 24 to conclude that $f$ distributes over $\triangledown$:

$$[\ \ (k^m{-}1) \triangledown (k^n{-}1) \ \ = \ \ k^{(m \triangledown n)}{-}1 \ \ ] \quad .$$

In particular, the Mersenne function, which maps $m$ to $2^m{-}1$, distributes over $\triangledown$:

$$(27) \quad [\ \ (2^m{-}1) \triangledown (2^n{-}1) \ \ = \ \ 2^{(m \triangledown n)}{-}1 \ \ ] \quad .$$

A corollary of (27) is the property

$$[\ \ (2^m{-}1) \triangledown (2^n{-}1) = 1 \quad \equiv \quad m \triangledown n = 1 \ \ ] \quad .$$

In words, two numbers $2^m{-}1$ and $2^n{-}1$ are coprime is the same as exponents $m$ and $n$ are coprime.

### *4.2. Enumerating the Rationals*

A standard theorem of mathematics is that the rationals are "denumerable", i.e. they can be put in one-to-one correspondence with the natural numbers. Another way of saying this is that it is possible to enumerate the rationals so that each appears exactly once.

Recently, there has been a spate of interest in the construction of bijections between the natural numbers and the (positive) rationals (see [8, 9, 10] and [11, pp. 94–97]). Gibbons *et al* [8] describe as "startling" the observation that the rationals can be efficiently enumerated[6] by "deforesting" the so-called "Calkin-Wilf" [10] tree of rationals. However, they claim that it is "not at all obvious" how to "deforest" the Stern-Brocot tree of rationals.

In this section, we derive an efficient algorithm for enumerating the rationals according to both orderings. 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 (see section 3.2). The enumeration is efficient in the sense that it has the same time and space complexity as the algorithm credited to Moshe Newman in [9], albeit with a constant-fold increase in the number of variables and number of arithmetic operations needed at each iteration.

[6]By an *efficient enumeration* we mean a method of generating each rational without duplication with constant cost per rational in terms of arbitrary-precision simple arithmetic operations.

Note that, in our view, it is misleading to use the name "Calkin-Wilf tree of rationals" because Stern [12] had already documented essentially the same structural characterisation of the rationals almost 150 years earlier than Calkin and Wilf. For more explanation, see the appendix in which we review in some detail the relevant sections of Stern's paper. Stern attributes the structure to Eisenstein, so henceforth we refer to the "Eisenstein-Stern" tree of rationals where recent publications (including our own [13]) would refer to the "Calkin-Wilf tree of rationals". Section 7 includes background information. For a comprehensive account of properties of the Stern-Brocot tree, including further relationships with Euclid's algorithm, see [7, pp. 116–118].

*4.2.1. 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 $\frac{m/(m\triangledown n)}{n/(m\triangledown n)}$. For example, $\frac{2}{3}$ is a rational in lowest form, whereas $\frac{4}{6}$ is the same rational, but not in lowest form.

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.

Beginning with an arbitrary pair of positive integers $m$ and $n$, the algorithm presented in section 3.2 calculates an invertible matrix $\mathbf{C}$ such that

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

It follows that

$$(28) \quad (1 \;\; 1) \times \mathbf{C}^{-1} \quad = \quad ({}^{m}\!/\!_{(m\triangledown n)} \;\; {}^{n}\!/\!_{(m\triangledown n)}) \quad .$$

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 (28) 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 binary 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.

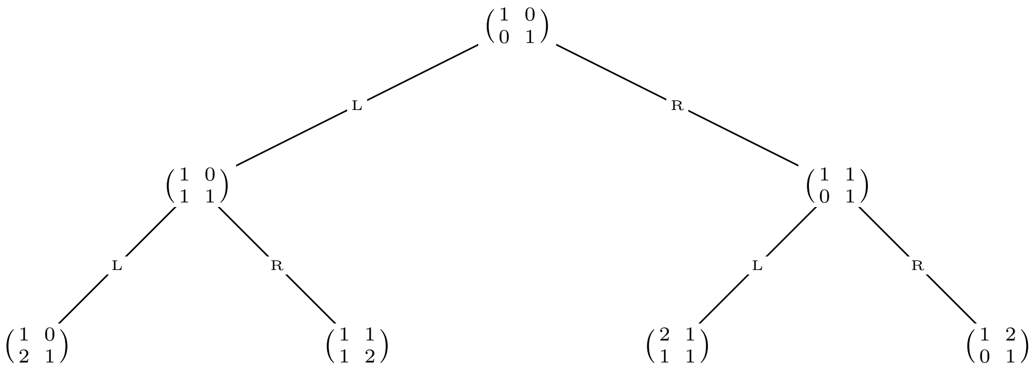


Figure 1: Tree of Products of **L** and **R**

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[7] is that the determinants of **A** and **B** are both equal to 1 and, hence, the determinant of any finite product of **L**s and **R**s is also 1.

Formally, we define the relation $\prec$ on matrices that are finite products of **L**s and **R**s by

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

(Note that the denominator in these fractions is strictly positive; this fact is easily proved by induction.) We prove that, for all such matrices **X**, **Y** and **Z**,

(29) $\mathbf{X}{\times}\mathbf{L}{\times}\mathbf{Y} \;\prec\; \mathbf{X} \;\prec\; \mathbf{X}{\times}\mathbf{R}{\times}\mathbf{Z}$ .

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

[7]The proof is an adaptation of the proof in [7, p. 117] that the rationals in the Stern-Brocot tree are all different. Our use of determinants corresponds to their use of "the fundamental fact" (4.31). Note that the definitions of **L** and **R** are swapped around in [7].)

relation. (Property (29) 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

(30) $\mathbf{X}\times\mathbf{L}\times\mathbf{Y} \prec \mathbf{X}$ ,

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 $\mathbf{L} = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, (30) 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, albeit longer, calculation (which exploits the monotonicity properties of multiplication and addition); 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 each other's transposition, 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

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

becomes

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

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

$$\mathbf{C}^{-1}\times\begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad = \quad \begin{pmatrix} m/(m\triangledown n) \\ n/(m\triangledown n) \end{pmatrix} \quad .$$

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

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

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

$\{\ \textbf{Invariant}:\quad (m\ n)\ =\ (x\ y)\times\mathbf{D}\ \}$

do $\ y<x\ \rightarrow\ (x\ y)\,,\mathbf{D}\ \ :=\ \ (x\ y)\times\mathbf{L}^{-1}\ ,\ \mathbf{L}{\times}\mathbf{D}$

$\Box\ \ x<y\ \rightarrow\ (x\ y)\,,\mathbf{D}\ \ :=\ \ (x\ y)\times\mathbf{R}^{-1}\ ,\ \mathbf{R}{\times}\mathbf{D}$

od

$\{\ (x\ y)\ =\ (m\nabla n\ \ m\nabla n)\ \ \wedge\ \ ({}^{m}\!/\!_{(m\nabla n)}\ \ {}^{n}\!/\!_{(m\nabla n)})\ =\ (1\ 1)\times\mathbf{D}\ \}$

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 the rational $\frac{m}{n}$ to the matrix $\mathbf{D}$ constructed by the revised algorithm together with the function from finite products, $\mathbf{D}$, of $\mathbf{L}$s and $\mathbf{R}$s to $\mathbf{D}\times\binom{1}{1}$.

$\Box$

*4.2.2. 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 sometimes called the Calkin-Wilf tree [8, 11, 10]; we call it the *Eisenstein-Stern tree* of rationals. (See the appendix for an explanation.) The first four levels of the tree are 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.)

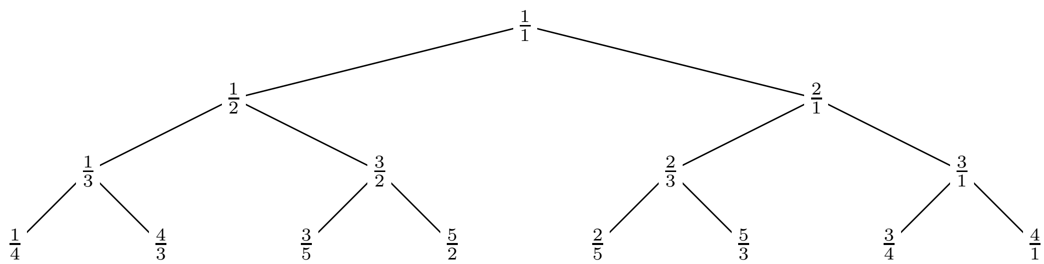


Figure 2: Eisenstein-Stern Tree of Rationals (aka Calkin-Wilf Tree)

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 [7, pp. 116–118]. See fig. 3. In this figure, the vector $\begin{pmatrix} x \\ y \end{pmatrix}$ has been displayed as $\frac{x}{y}$.

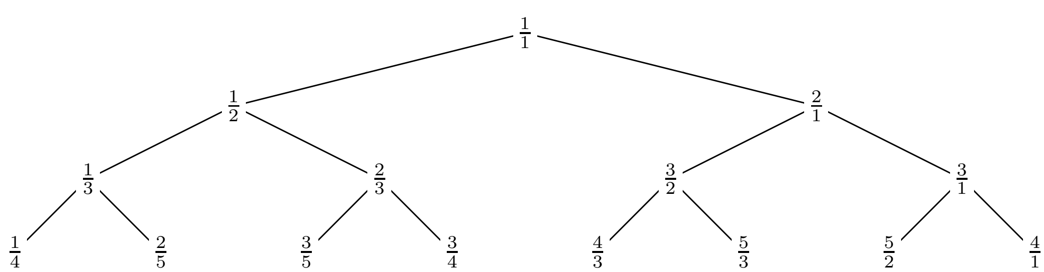


Figure 3: Stern-Brocot Tree of Rationals

Of course, if we can find an efficient way of enumerating the matrices in fig. 1, we immediately get an enumeration of the rationals as displayed in the Eisenstein-Stern 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.

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 $\mathbf{D}$, 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 $\mathbf{D}$ 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 $t\mathrm{LR}^j$, where $\mathrm{R}^j$ is a string of $j$ Rs, its successor is $t\mathrm{RL}^j$.

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 $j$,

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

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

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

(We omit the details. Briefly, by induction, $\mathbf{L}^j$ equals $\begin{pmatrix} 1 & 0 \\ j & 1 \end{pmatrix}$. Also, $\mathbf{R}$ is the transpose of $\mathbf{L}$.)

The final task is to determine, given a matrix $\mathbf{D}$, 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 $\mathbf{D} = \mathbf{T} \times \mathbf{L} \times \mathbf{R}^j$ for some $\mathbf{T}$. This can be determined by examining Euclid's algorithm once more.

The matrix form of Euclid's algorithm discussed in theorem 31 computes a matrix $\mathbf{D}$ given a pair of positive numbers $m$ and $n$; it maintains the invariant

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

$\mathbf{D}$ is initially the identity matrix and $x$ and $y$ are initialised to $m$ and $n$, respectively; immediately following the initialisation process, $\mathbf{D}$ 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 $\mathbf{D}$ is premultiplied by $\mathbf{R}$ is thus the greatest number $j$ such that $j{\times}m$ is less than $n$, which is $\lfloor \frac{n-1}{m} \rfloor$. Now suppose the input values $m$ and $n$ are coprime. Then, on termination of the algorithm, $(1\ 1) \times \mathbf{D}$ equals $(\, m\ n \,)$. That is, if

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

then,

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

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 $\mathbf{D}$ is a power of $\mathbf{R}$ exactly when the rationals in the Eisenstein-Stern, or Stern-Brocot, tree are integers, and this integer is the number of the next level in the tree (where the root is on level 0). So, it is easy to test whether the last matrix on the current level has been reached. Equally, the first matrix on the next level is easily calculated. For reasons we discuss in the next section, we choose to test whether the rational in the Eisenstein-Stern tree is an integer; that is, we evaluate the boolean $\mathbf{D}_{00}+\mathbf{D}_{10} = 1$. In this way, we get the following (non-terminating) program which computes the successive values of $\mathbf{D}$.

$\mathbf{D} := \mathbf{I}\,;$

do $\mathbf{D}_{00}+\mathbf{D}_{10} = 1 \;\rightarrow\; \mathbf{D} \;:=\; \begin{pmatrix} 1 & 0 \\ \mathbf{D}_{01}+\mathbf{D}_{11} & 1 \end{pmatrix}$

$\Box \quad \mathbf{D}_{00}+\mathbf{D}_{10} \neq 1 \;\rightarrow\; j := \left\lfloor \frac{\mathbf{D}_{01}+\mathbf{D}_{11}-1}{\mathbf{D}_{00}+\mathbf{D}_{10}} \right\rfloor \;;\; \mathbf{D} := \mathbf{D} \times \begin{pmatrix} 2j+1 & 1 \\ -1 & 0 \end{pmatrix}$

od

A minor simplification of this algorithm is that the "$-\,1$" in the assignment to $j$ can be omitted. This is because $\lfloor \frac{n-1}{m} \rfloor$ and $\lfloor \frac{n}{m} \rfloor$ are equal when $m$ and $n$ are coprime and $m$ is different from 1. We return to this shortly.

*4.2.3. The Enumerations*

As remarked earlier, we immediately get an enumeration of the rationals as displayed in the Eisenstein-Stern 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.

In the case of enumerating the Eisenstein-Stern tree, several optimisations are possible. First, it is immediate from our derivation that the value assigned to the local variable $j$ is a function of $(1\ 1)\times\mathbf{D}$. In turn, the matrix $\begin{pmatrix}2j+1 & 1\\ -1 & 0\end{pmatrix}$ is also a function of $(1\ 1)\times\mathbf{D}$. Let us name the function $J$, so that the assignment becomes

$$\mathbf{D}\ :=\ \mathbf{D}\times J.((1\ 1)\times\mathbf{D})\ \ .$$

Then, the Eisenstein-Stern enumeration iteratively evaluates

$$(1\ 1)\times(\mathbf{D}\times J.((1\ 1)\times\mathbf{D}))\ \ .$$

Matrix multiplication is associative; so this is

$$((1\ 1)\times\mathbf{D})\times J.((1\ 1)\times\mathbf{D})\ \ ,$$

which is also a function of $(1\ 1)\times\mathbf{D}$. Moreover —in anticipation of the current discussion— we have been careful to ensure that the test for a change in the level in the tree is also a function of $(1\ 1)\times\mathbf{D}$. Combined together, this means that, in order to enumerate the rationals in Eisenstein-Stern order, it is not necessary to compute $\mathbf{D}$ at each iteration, but only $(1\ 1)\times\mathbf{D}$. Naming the two components of this vector $m$ and $n$, and simplifying the matrix multiplications, we get[8]

$$\begin{array}{llllll}
\multicolumn{6}{l}{m,n\ :=\ 1,1\ ;}\\
\mathsf{do} & m=1 & \rightarrow & m,n & := & n{+}1\,,\,m\\
\Box & m\neq 1 & \rightarrow & m,n & := & (2\left\lfloor\dfrac{n-1}{m}\right\rfloor+1)\times m-n\ \ ,\ \ m\\
\mathsf{od} &&&&&
\end{array}$$

At this point, a further simplification is also possible. We remarked earlier that $\lfloor\frac{n-1}{m}\rfloor$ equals $\lfloor\frac{n}{m}\rfloor$ when $m$ and $n$ are coprime and $m$ is different from

[8]Recall that, to comply with existing literature, the enumerated rational is $\frac{n}{m}$ and not $\frac{m}{n}$.

1. By good fortune, it is also the case that $(2\lfloor\frac{n}{m}\rfloor+1)\times m-n$ simplifies to $n{+}1$ when $m$ is equal to 1. That is, the elimination of "$-$ 1" in the evaluation of the floor function leads to the elimination of the entire case analysis! This is the algorithm attributed to Newman in [9].

$$
\begin{array}{l}
m{,}n \ := \ 1{,}1 \ ; \\
\textsf{do} \quad m{,}n \ := \ (2\left\lfloor\dfrac{n}{m}\right\rfloor+1)\times m-n \ , \ m \\
\textsf{od}
\end{array}
$$

*4.2.4. Discussion*

Our construction of an algorithm for enumerating the rationals in Stern-Brocot order was motivated by reading two publications, [7, pp. 116–118] and [8]. Gibbons, Lester and Bird [8] show how to enumerate the elements of the Eisenstein-Stern tree, but claim that "it is not at all obvious how to do this for the Stern-Brocot tree". Specifically, they say[9]:

> 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.

In this section, we have shown that it is possible to enumerate the rationals in Stern-Brocot order without incurring "additional state of increasing size". More importantly, we have presented *one* enumeration algorithm with *two* specialisations, one being the "Calkin-Wilf" enumeration they present, and the other being the Stern-Brocot enumeration that they described as being "not at all obvious".

The optimisation of Eisenstein-Stern enumeration which leads to Newman's algorithm is not possible for Stern-Brocot enumeration. Nevertheless, the complexity of Stern-Brocot enumeration is the same as the complexity

[9] Recall that they attribute the tree to Calkin and Wilf rather than Eisenstein and Stern.

of Newman's algorithm, both in time and space. The only disadvantage of Stern-Brocot enumeration is that four variables are needed in place of two; the advantage is the (well-known) advantage of the Stern-Brocot tree over the Eisenstein-Stern tree — the rationals on a given level are in ascending order.

Gibbons, Lester and Bird's goal seems to have been to show how the functional programming language Haskell implements the various constructions – the construction of the tree structures and Newman's algorithm. In doing so, they repeat the existing mathematical presentations of the algorithms as given in [7, 10, 9]. The ingredients for an efficient enumeration of the Stern-Brocot tree are all present in these publications, but the recipe is missing!

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.

## 5. Conclusion

In our view, much of mathematics is inherently algorithmic; it is also clear that, in the modern age, algorithmic problem solving is just as important, if not much more so, than in the 19th century. Somehow, however, mathematical education in the 20th century lost sight of its algorithmic roots. We hope to have exemplified in this paper how a fresh approach to introductory number theory that focuses on the algorithmic content of the theory can combine practicality with mathematical elegance. By continuing this endeavour we believe that the teaching of mathematics can be enriched and given new vigour.

## References


[1] D. E. Knuth, The Art of Computer Programming, volume 2 (3rd ed.): Seminumerical Algorithms, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[2] K. E. Hirst, Numbers, Sequences and Series, Edward Arnold, 1995.

[3] D. M. Burton, Elementary Number Theory, 6th Edition, McGraw-Hill Higher Education, 2005.

[4] J. B. Fraleigh, A First Course in Abstract Algebra, 6th Edition, Addison Wesley Longman Inc., 1998.

[5] A. van Gasteren, On the Shape of Mathematical Arguments, no. 445 in LNCS, Springer-Verlag, 1990.

[6] E. W. Dijkstra, Fibonacci and the greatest common divisor (April 1990). URL `http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1077.PDF`

[7] R. L. Graham, D. E. Knuth, O. Patashnik, Concrete Mathematics : a Foundation for Computer Science, 2nd Edition, Addison-Wesley Publishing Company, 1994.

[8] J. Gibbons, D. Lester, R. Bird, Enumerating the rationals, Journal of Functional Programming 16 (3) (2006) 281–291.

[9] D. E. Knuth, C. Rupert, A. Smith, R. Stong, Recounting the rationals, continued, American Mathematical Monthly 110 (7) (2003) 642–643.

[10] N. Calkin, H. S. Wilf, Recounting the rationals, The American Mathematical Monthly 107 (4) (2000) 360–363.

[11] M. Aigner, G. Ziegler, Proofs From The Book, 3rd Edition, Springer-Verlag, 2004.

[12] M. A. Stern, Üeber eine zahlentheoretische Funktion, Journal für die reine und angewandte Mathematik 55 (1858) 193–220.

[13] R. Backhouse, J. F. Ferreira, Recounting the rationals: Twice!, in: Mathematics of Program Construction, Vol. 5133 of LNCS, 2008, pp. 79–91.
URL `http://joaoff.com/publications/2008/rationals`

[14] A. Brocot, Calcul des rouages par approximation, nouvelle méthode, Revue Chronométrique 3 (1861) 186–194, available via http://joaoff.com/publications/2008/rationals/.

[15] B. Hayes, On the teeth of wheels, American Scientist 88 (4) (2000) 296–300.
URL http://www.americanscientist.org/issues/pub/2000/4/on-t%he-teeth-of-wheels

[16] N. J. A. Sloane, The On-Line Encyclopedia of Integer Sequences, http://www.research.att.com/~njas/sequences/.

[] R. Backhouse, J. F. Ferreira, On Euclid's algorithm and elementary number theory (2009).
URL http://joaoff.com/publications/2009/euclid-alg

# Appendix: Historical Remarks

The primary novel result of our paper is the construction given in section 4.2 of an algorithm to enumerate the rationals in Stern-Brocot order. Apart from minor differences, this section of our paper was submitted in April 2007 to the American Mathematical Monthly; it was rejected in November 2007 on the grounds that it was not of sufficient interest to readers of the Monthly. One (of two referees) did, however, recommend publication. The referee made the following general comment.

> Each of the two trees of rationals—the Stern-Brocot tree and the Calkin-Wilf tree—has some history. Since this paper now gives the definitive link between these trees, I encourage the authors, perhaps in their Discussion section, to also give the definitive histories of these trees, something in the same spirit as the Remarks at the end of the Calkin and Wilf paper.

Since the publication of [13], we have succeeded in obtaining copies of the original papers and it is indeed interesting to briefly review the papers. But we do not claim to provide "definitive histories of these trees" — that is a task for a historian of mathematics.

Section 6 is about the paper [12] published in 1858 by Stern. The surprising fact that emerges from the review is that the so-called "Calkin-Wilf" tree of rationals, and not just the "Stern-Brocot" tree, is studied in detail in his paper. Moreover, of the two structures, the "Calkin-Wilf" tree is more readily recognised; the "Stern-Brocot" tree requires rather more understanding to identify. Brocot's paper [14], which we review in section 7, is interesting because it illustrates how 19th century mathematics was driven by practical, algorithmic problems. (For additional historical remarks, see also [15].)

## 6. Stern's Paper

Earlier we have commented that the structure that has recently been referred to as the "Calkin-Wilf" tree was documented by Stern [12] in 1858. In this section we review those sections of Stern's paper that are relevant to our own.

### *6.1. The Eisenstein Array*

Stern's paper is a detailed study of what has now become known as the "Eisenstein array" of numbers (see, for example, [16, sequence A064881]). (Stern's paper cites two papers written by the more famous mathematician Gotthold Eisenstein; we have not read these papers.) Given two natural numbers $m$ and $n$, Stern describes a process (which he attributes to Eisenstein) of generating an infinite sequence of rows of numbers. The *zeroth* row in the sequence ("nullte Entwickelungsreihe") is the given pair of numbers:

$$m \quad n \quad .$$

Subsequent rows are obtained by inserting between every pair of numbers the sum of the numbers. Thus the *first* row is

$$m \quad m{+}n \quad n$$

and the *second* row is

$$m \quad 2{\times}m+n \quad m{+}n \quad m+2{\times}n \quad n \quad .$$

The process of constructing such rows is repeated indefinitely. The sequence of numbers obtained by concatenating the individual rows in order is what is now called the *Eisenstein array* and denoted by $Ei(m,n)$ (see, for example, [16, sequence A064881]) . Stern refers to each occurrence of a number in rows other than the zeroth row as either a *sum element* ("Summenglied") or a *source element* ("Stammglied"). The sum elements are the newly added numbers. For example, in the first row the number $m+n$ is a sum element; in the second row the number $m+n$ is a source element.

### *6.2. The Eisenstein-Stern Tree of Rationals*

A central element of Stern's analysis of the Eisenstein array is the consideration of subsequences of numbers in individual rows. He calls these *groups* ("Gruppen") and he records the properties of pairs of consecutive numbers (groups of size two — "zweigliedrige Gruppen") and triples of consecutive numbers (groups of size three — "dreigliedrige Gruppen").

In sections 5 thru 8 of his paper, Stern studies $Ei(1,1)$, the Eisenstein array that begins with the pair $(1,1)$. He proves that all pairs of consecutive numbers in a given row are coprime and every pair of coprime numbers appears exactly once as such a pair of consecutive numbers. He does not use the word "tree" —tree structures are most probably an invention of modern computing science— and he does not refer to "rational numbers" —he refers instead to relatively prime numbers ("relatieve Primzahlen")— but there is no doubt that, apart from the change in terminology, he describes the tree of rationals that in recent years has been referred to as the "Calkin-Wilf" tree of rationals. It is for this reason that we believe it is misleading to use the name "Calkin-Wilf tree" and prefer to use the name "Eisenstein-Stern tree".

Other sections of Stern's paper record additional properties of the tree, which we do not discuss here. For example, Stern discusses how often each number appears as a sum number.

### *6.3. The Stern-Brocot Tree of Rationals*

Identification of the so-called Stern-Brocot tree of rationals in Stern's paper is more demanding. Recall the process of constructing a sequence of rows of numbers from a given pair of numbers $m$ and $n$. It is clear that every number is a linear combination of $m$ and $n$. Stern studies the *coefficients* ("Coefficienten"), i.e. the pair of multiplicative factors of $m$ and $n$, defined by the linear combination. Fig. 4 displays the coefficients in a way that allows direct comparison with the Stern-Brocot tree of rationals (fig. 3). (The reader may also wish to compare fig. 4 with Graham, Knuth and Patashnik's depiction of the tree [7, p. 117].)

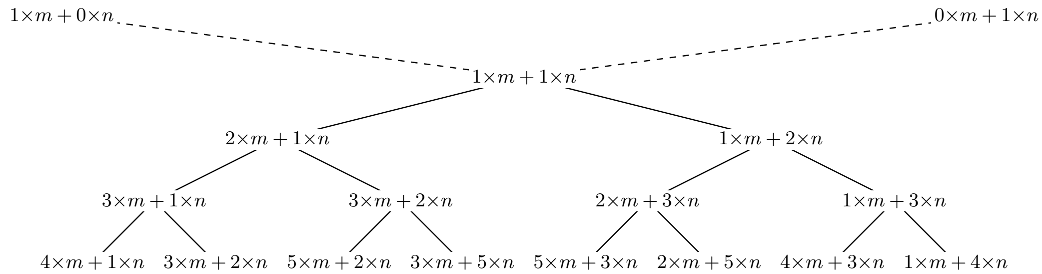


Figure 4: Tree of “coefficients” of $Ei(m,n)$

The numbers at the top-left and top-right of fig. 4 are the numbers $m$ and $n$ written as $1{\times}m+0{\times}n$ and $0{\times}m+1{\times}n$, respectively, in order to make the coefficients clear. This, we recall, is the zeroth row in Stern's structure.

In the subsequent levels of the tree, only the sum elements are displayed. The correspondence between figs. 4 and 3 should be easy to see; the number $k{\times}m+l{\times}n$ in fig. 4 is displayed as the rational $\frac{l}{k}$ in fig. 3. The “fundamental fact” (4.31) in [7] is observed by Stern [12, equation (8), p.207] and used immediately to infer that coefficients are relatively prime. In section 15 of his paper, Stern uses the (already proven) fact that the Eisenstein-Stern tree is a tree of (all) rationals to deduce that the Stern-Brocot tree is also a tree of rationals.

### *6.4. Newman's Algorithm*

An interesting question is whether Stern also documents the algorithm currently attributed to Moshe Newman for enumerating the elements of the Eisenstein array. This is a question we found difficult to answer because of our limited understanding of German. However, the answer would appear to be: almost, but not quite!

As remarked earlier, Stern documents a number of properties of groups of numbers in rows of the Eisenstein array, in particular groups of size three. Of course, a group of size three comprises two groups of size two. Since groups of size two in the Eisenstein array correspond to rationals in the Eisenstein-Stern tree, by studying groups of size three Stern is effectively studying consecutive rationals in the Eisenstein-Stern tree of rationals.

It is important to note that Stern's focus is the sequence of *rows* of numbers (in modern terminology, the tree of numbers) as opposed to the (flattened) sequence of numbers defined by $Ei(m,n)$ — significantly, the last number in one row and the first number in the next row do not form a "group" according to Stern's definition. This means that, so far as we have been able to determine, he nowhere considers a triple of numbers that crosses a row boundary.

Newman's algorithm (in the form we use in section 4.2.3) predicts that each triple of numbers in a given row of $Ei(1{,}1)$ has the form

$$a \quad b \quad (2\left\lfloor \frac{a}{b} \right\rfloor + 1) \times b - a$$

(Variable names have been chosen to facilitate comparison with Stern's paper.) It follows immediately that the sum of the two outer elements of the triple is divisible by the middle element (that is, $a + ((2\left\lfloor \frac{a}{b} \right\rfloor + 1) \times b - a)$ is divisible by $b$); this fact is observed by Stern (for triples in a given row) in section 4 of his paper. Importantly for what follows, Stern observes that the property holds for $Ei(m{,}n)$ for arbitrary natural numbers $m$ and $n$, and not just $Ei(1{,}1)$. Stern observes further [12, (4) p.198] that each triple in $Ei(m{,}n)$ has the form

$$(32) \quad a \quad b \quad (2t + 1) \times b - a$$

for some number $t$. Stern identifies $t$ as the number of rows preceding the current row in which the number $b$ occurs as a sum element. (In particular, if $b$ is a sum element then $t$ equals 0.) Stern shows how to calculate $t$ from the position of $b$ in the row — effectively by expressing the position as a binary numeral. (Note that "$t$" is the variable name used in Stern's paper; it has the same role as the variable "$j$" in our derivation of the algorithm in section 4.2.3.)

So far as we have been able to determine, Stern does not explicitly remark that $t$ equals $\left\lfloor \frac{a}{b} \right\rfloor$ in the case of $Ei(1{,}1)$, but he does so implicitly in section 10 where he relates the the continued fraction representation of $\frac{a}{b}$ to the row number in which the pair $(a, b)$ occurs. He does not appear to suggest a similar method for computing $t$ in the general case of enumerating $Ei(M{,}N)$. However, it is straightforward to combine our derivation of Newman's algorithm with Stern's theorems to obtain an algorithm to enumerate the elements of $Ei(M{,}N)$ for arbitrary natural numbers $M$ and $N$. Interested readers may consult our website ??? where several implementations are discussed.

As stated at the beginning of this section, the conclusion is that Stern almost derives Newman's algorithm, but not quite. On the other hand, because his analysis is of the general case $Ei(m,n)$ as opposed to $Ei(1,1)$, his results are more general.

### *6.5. Stern-Brocot Enumeration*

We now turn to the question whether Stern also gives an algorithm for enumerating the rationals in Stern-Brocot order.

To this end, we observe that the form (32) extends to the coefficients of each element of $Ei(M,N)$, and hence to the elements of the Stern-Brocot tree. Specifically, triples in $Ei(M,N)$ have the form

$$n_0M{+}m_0N \quad n_1M{+}m_1N \quad ((2k+1)n_1-n_0)M+((2k+1)m_1-m_0)N$$

It is easy to exploit this formula directly to get an enumeration of the rationals in Stern-Brocot order, just as we did above to obtain an enumeration of $Ei(M,N)$. Just recall that the Stern-Brocot rationals are given by the coefficients of the sum elements, and the sum elements are the odd-numbered elements in the rows of $Ei(M,N)$ (where numbering starts from zero). The algorithm so obtained is the one we derived in section 4.2.3.

In this sense, Stern does indeed provide an algorithm for enumerating the rationals in Stern-Brocot order, albeit implicitly. However, as with Newman's algorithm, he fails to observe the concise formula for the value of the variable $k$. Also, a major methodological difference is our exploitation of the concision and precision afforded by matrix algebra. Given the state of development of matrix algebra in 1858, Stern cannot be criticised for not doing the same; this cannot be said, however, for mathematical texts still in use today!

Finally, we remark that Stern returns to the properties of triples in section 19 of his paper. Unfortunately, we have been unable to fully understand this section. Our understanding is that the section is also relevant to enumerating the rationals in Stern-Brocot order but is restricted to determining modulo-values rather than exact values.

## 7. Brocot, the Watchmaker

Achille Brocot was a famous French watchmaker who, some years before the publication of his paper [14], had to fix some pendulums used for astronomical measurements. However, the device was incomplete and he did not know how to compute the number of teeth of cogs that were missing. He was unable to find any literature helpful to the solution of the problem, so, after some experiments, he devised a method to compute the numbers. In his paper, Brocot illustrates his method with the following example:

> A shaft turns once in 23 minutes. We want suitable cogs so that another shaft completes a revolution in 3 hours and 11 minutes, that is 191 minutes.

The ratio between both speeds is $\frac{191}{23}$, so we can clearly choose a cog with 191 teeth, and another one with 23 teeth. But, as Brocot wrote, it was not possible, at that time, to create cogs with so many teeth. And because 191 and 23 are coprime, cogs with fewer teeth can only approximate the true ratio.

Brocot's contribution was a method to compute approximations to the true ratios (hence the title of his paper, "Calculus of cogs by approximation"). He begins by observing that $\frac{191}{23}$ must be between the ratios $\frac{8}{1}$ and $\frac{9}{1}$. If we choose the ratio $\frac{8}{1}$, the error is $-7$ since $8\times 23 = 1\times 191 - 7$. This means that if we choose this ratio, the slower cog completes its revolution seven minutes early, i.e., after $8\times 23$ minutes. On the other hand, if we choose the ratio $\frac{9}{1}$, the error is 16 since $9\times 23 = 1\times 191 + 16$, meaning that the slower cog completes its revolution sixteen minutes late, i.e., after $9\times 23$ minutes.

Accordingly, Brocot writes two rows:

| | | |
|---|---|---|
| 8 | 1 | -7 |
| 9 | 1 | +16 |

His method consists in iteratively forming a new row, by adding the numbers in all three columns of the rows that produce the smallest error. Initially, we only have two rows, so we add the numbers in the three columns and we write the row of sums in the middle.

| | | |
|---|---|---|
| 8 | 1 | -7 |
| 17 | 2 | +9 |
| 9 | 1 | +16 |

(If we choose the ratio $\frac{17}{2}$, the slower cog completes its revolution $\frac{9}{2}$ minutes later, since $\frac{17}{2} = \frac{191+\frac{9}{2}}{23}$.) Further approximations are constructed by adding a row adjacent to the row that minimises the error term. The process ends once we reach the error 0, which refers to the true ratio. The final state of the table is:

| | | |
|---|---|---|
| 8 | 1 | -7 |
| 33 | 4 | -5 |
| 58 | 7 | -3 |
| 83 | 10 | -1 |
| 191 | 23 | 0 |
| 108 | 13 | +1 |
| 25 | 3 | +2 |
| 17 | 2 | +9 |
| 9 | 1 | +16 |

The conclusion is that the two closest approximations to $\frac{191}{23}$ are ratios of $\frac{83}{10}$ (which runs $\frac{1}{10}$ minutes faster) and $\frac{108}{13}$ (which runs $\frac{1}{13}$ minutes slower). We could continue this process, getting at each stage a closer approximation to $\frac{191}{23}$. In fact, Brocot refines the table shown above, in order to construct a multistage cog train (see [14, p. 191]).

At each step in Brocot's process we add a new ratio $\frac{m+m'}{n+n'}$, which is usually called the mediant of $\frac{m}{n}$ and $\frac{m'}{n'}$. Similarly, each node in the Stern-Brocot tree is of the form $\frac{m+m'}{n+n'}$, where $\frac{m}{n}$ is the nearest ancestor above and to the left, and $\frac{m'}{n'}$ is the nearest ancestor above and to the right. (Consider, for example, the rational $\frac{4}{3}$ in figure 3. Its nearest ancestor above and to the left is $\frac{1}{1}$ and its nearest ancestor above and to the right is $\frac{3}{2}$.) Brocot's process can be used to construct the Stern-Brocot tree: first, create an array that contains initially the rationals $\frac{0}{1}$ and $\frac{1}{0}$; then, insert the rational $\frac{m+m'}{n+n'}$ between two adjacent fractions $\frac{m}{n}$ and $\frac{m'}{n'}$. In the first step we add only one rational to the array

$$\frac{0}{1}, \frac{1}{1}, \frac{1}{0} \quad ,$$

but in the second step we add two new rationals:

$$\frac{0}{1}, \frac{1}{2}, \frac{1}{1}, \frac{2}{1}, \frac{1}{0} \quad .$$

Generally, in the $n^{th}$ step we add $2^{n-1}$ new rationals. Clearly, this array can be represented as an infinite binary tree, whose first four levels are represented in figure 3 (we omit the fractions $\frac{0}{1}$ and $\frac{1}{0}$).

The most interesting aspect to us of Brocot's paper is that it solves an *algorithmic* problem. Brocot was faced with the practical problem of how to approximate rational numbers in order to construct clocks of satisfactory accuracy and his solution is undisputably an *algorithm.* Stern's paper is closer to a traditional mathematical paper but, even so, it is an in-depth study of an algorithm for generating rows of numbers of increasing length.

## 8. Conclusion

There can be no doubt that what has been dubbed in recent years the "Calkin-Wilf" tree of rationals is, in fact, a central topic in Stern's 1858 paper. Calkin and Wilf [10] admit that in Stern's paper "there is a structure that is essentially our tree of fractions" but add "in a different garb" and do not clarify what is meant by "a different garb". It is unfortunate that the misleading name has now become prevalent; in order to avoid further misinterpretations of historical fact, it would be desirable for Stern's paper to be translated into English.

We have not attempted to determine how the name "Stern-Brocot" tree came into existence. It has been very surprising to us how much easier it is to identify the Eisenstein-Stern tree in Stern's paper in comparison to identifying the Stern-Brocot tree.

*Acknowledgements.* Thanks go to Christian Wuthrich for help in translating the most relevant parts of Stern's paper, and 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, and to Arjan Mooij and Jeremy Weissmann for their comments on section 4.1.