Communications in Mathematics **33** (2025), no. 2, Paper no. 3
DOI: https://doi.org/10.46298/cm.12627

Proving Properties of φ-Representations with the `Walnut` Theorem-Prover

Jeffrey Shallit

Abstract. We revisit a classic theorem of Frougny and Sakarovitch concerning automata for φ-representations, and show how to obtain it in a different and more computationally direct way. Using it, we can find simple, induction-free proofs of existing results in the literature about these representations, in a uniform and straightforward manner. In particular, we can easily and "automatically" recover many of the results of recent papers of Dekking and Van Loon. We also obtain a number of new results on φ-representations.

Contents

MSC 2020: 11A63 (Primary); 11A67, 68R15, 03D05, 11B85, 68Q45 (Secondary).
Keywords: β-expansion, Zeckendorf representation, finite automaton, decision procedure.
Contact information:
Affiliation: School of Computer Science, University of Waterloo, Canada.
Email: `shallit@uwaterloo.ca`.

In honor of the 75th birthday of Christiane Frougny

1 Introduction

In 1957, George Bergman [2] observed that one can expand a non-negative real number z in base $\varphi = (1 + \sqrt{5})/2$, in the sense that there exist "digits" $a_i \in \{0, 1\}$ such that

$$z = \sum_{-\infty \leq i < r} a_i \varphi^i. \tag{1}$$

In analogy with base-b representation, we can write such a base-φ representation as a string of digits $a_{r-1} \cdots a_0 \,.\, a_1 a_2 a_3 \cdots$ where the non-negative and negative powers are separated by the analogue of a "decimal point". We call $a_{r-1} \cdots a_0$ the *left part* and $.\, a_1 a_2 a_3 \cdots$ the *right part* of the φ-representation. The left part is always finite, while the right part may be finite or infinite. These parts are analogous to the more familiar integer and fractional parts of decimal representations of the positive reals.

If Equation (1) holds, then we write

$$z = [a_{r-1} \cdots a_0 \,.\, a_1 a_2 a_3 \cdots]_\varphi.$$

For example, $2 = [10.01]_\varphi = [1.11]_\varphi$ and $\frac{1}{2} = [.010010010010 \cdots]_\varphi$.

Furthermore, such a representation $x.y$ as a string of binary digits with a decimal point is essentially unique if one imposes the very natural restriction that $a_i a_{i+1} \neq 1$ for all i (or, equivalently, if the string xy contains no occurrence of the block 11), and furthermore that if the expansion is infinite, it cannot end in $010101 \cdots$. Throughout the paper, we adopt the convention that in expansions of the form $[x.y]_\varphi$, we disregard leading zeros in x and trailing zeros in y. Thus 10.01 and 010.0100 are regarded as "essentially" the same. This convention is extremely useful, as sometimes our automata will need to pad multiple inputs with zeros to ensure they all have the same length.

We call a φ-representation obeying these rules *canonical*, and write it as $(x)_\varphi$. Bergman [2] proved that the canonical φ-representation of a natural number is finite, and Table 1 gives the canonical expansions of the first few positive integers.

n	1	2	3	4	5	6	7
$(n)_\varphi$	1.	10.01	100.01	101.01	1000.1001	1010.0001	10000.0001

Table 1: First few canonical expansions.

The left and right parts of the canonical φ-representation of n can be found in sequences A105424 and A341722 in the *On-Line Encyclopedia of Integer Sequences* (OEIS) [36].

Bergman's ideas were generalized to positive real numbers β by Rényi [30], who called such expansions *β-expansions*. Later, fundamental work characterizing such expansions was done by Parry [29].

Several researchers, including Bertrand-Mathis and Frougny, have found deep and interesting connections between these expansions and finite automata when β is an algebraic number. See, for example, [1, 5, 16, 17, 18, 19, 20, 21, 22].

In a very interesting article, Frougny and Sakarovitch [23] proved that the canonical φ-representation of a non-negative integer can be computed by a finite automaton A, in the following sense: A takes three inputs *in parallel:* n represented in the Zeckendorf numeration system (see below), and binary strings x and y, and accepts if and only if $(n)_\varphi = x.y^R$. Here, y^R denotes the reversal of the string y. This representation is called "folded"; it is useful in part because the coefficients corresponding to small (in absolute value) powers of φ are grouped together when read by an automaton. For example, the entry of Table 1 corresponding to $n = 6$ would correspond to the accepted input $[1, 1, 1][0, 0, 0][0, 1, 0][1, 0, 0]$: the first entries of each triple spell out 1001, which is 6 in Zeckendorf representation; the second entries spell out 1010 and the third entries spell out 1000. However, their article is rather challenging to read and they did not explicitly present this automaton.

The first goal of this paper is to explain how the Frougny-Sakarovitch automaton can be, rather easily, computed explicitly using existing software; namely, the free software `Walnut` initially developed by Hamoon Mousavi [28]. This software can prove or disprove theorems about automatic sequences and their generalizations, using a decision procedure for a certain extension of Presburger arithmetic [34]. It suffices to express the desired assertions in first-order logic. Furthermore, if the particular logical statement has free variables, `Walnut` will compute a finite automaton accepting precisely those values of the free variables making the statement true. Finally, if a formula F has two or more free variables, say n and $x_1, x_2, \ldots x_t$, `Walnut` can compute the number of t-tuples $(x_1, \ldots, x_t)$ such that $F(n, x_1, \ldots, x_t)$ evaluates to `TRUE`.

Once we have this automaton, we can use it to easily reprove existing results from the literature in a straightforward and uniform manner. For example, we reprove some results recently considered by Dekking and Van Loon [13]. They called a finite φ-representation (not necessarily canonical) of an integer n a *Knott expansion* if it does not end in 011, and they developed a rather complicated method for computing the number of different Knott expansions of n. We will see that the Frougny-Sakarovitch automaton allows us to enumerate Knott expansions, and thereby recover the results of Dekking and Van Loon in a purely "automatic" fashion, without any tedious inductions. Furthermore, by simply changing the conditions we impose on the form of the expansion, we can easily and "automatically" enumerate other types of expansions. The second goal of the paper is to illustrate these techniques and obtain a number of new results.

The paper is organized as follows. In Section 2 we construct the Frougny-Sakarovitch automaton. In Sections 3–7 we prove various old and new results about φ-representations; one new result is Theorem 3.1, which proves a 2012 conjecture of Dale Gerdemann. In Section 8 we discuss Knott expansions; in Section 9 we discuss a different type of expansion, called a "natural expansion", introduced by Dekking and Van Loon; and in Section 10 we discuss another kind of expansion, called DVL-expansions. Finally, in Section 11 we discuss more general expansions for algebraic integers.

2 The Frougny-Sakarovitch automata

We will need two different representations of integers: the Zeckendorf system and the negaFibonacci system.

Let the Fibonacci numbers be defined by the recurrence $F_n = F_{n-1} + F_{n-2}$ and initial conditions $F_0 = 0$ and $F_1 = 1$. Note that this uniquely defines F_n for *all* integers n (even negative integers).

In the well-known Zeckendorf system [26, 38], we write a non-negative integer n as a sum $n = \sum_{2 \leq i \leq t} a_i F_i$ with $a_i \in \{0, 1\}$. We abbreviate this by the expression $[x]_F$, where $x = a_t a_{t-1} \cdots a_2$. This representation is unique if we impose the condition $a_i a_{i+1} \neq 1$; we let $(n)_F$ denote the canonical representation obeying this condition. Thus $(43)_F = 10010001$.

It will also be useful to be able to represent negative integers. In the negaFibonacci system [7,35], we write an integer n (positive, negative, or zero) as a sum $n = \sum_{1 \leq i \leq t} a_i F_{-i}$, where again $a_i \in \{0, 1\}$. We abbreviate this by the expression $[x]_{-F}$, where $x = a_t \cdots a_1$. Again, this representation is unique if we impose the condition $a_i a_{i+1} \neq 1$; we let $(n)_{-F}$ denote the canonical representation obeying this condition. Thus $(43)_{-F} = 101001010$.

(An interesting alternative to the negaFibonacci system has recently been proposed by Labbé and Lepšová; see [25].)

We start by constructing the first Frougny-Sakarovitch automaton. It has three inputs over $\{0, 1\}$ that are read in parallel: w, x, and y. Here

- w is the (canonical) Zeckendorf representation of some integer n,
- x is the left part of a φ-representation
- y^R is the right part of a φ-representation

and the automaton accepts if and only if $n = [x.y]_\varphi$. Notice that for this automaton *we impose no other conditions* on x and y; they are simply arbitrary binary strings. However, w is assumed to contain no occurrence of 11. All three inputs are allowed to have any number of leading zeros, which permits the inputs to be of the same length so they can be read simultaneously in parallel.

The basic idea is very simple. As Bergman [2] noted (and easily proved by induction), we have $\varphi^k = F_k\varphi + F_{k-1}$ for all integers k. It now follows that

$$\begin{aligned}
\sum_{0\le i<r} a_i\varphi^i &= \sum_{0\le i<r} a_i(F_i\varphi + F_{i-1}) \\
&= \left(\sum_{0\le i<r} a_iF_i\right)\varphi + \left(\sum_{0\le i<r} a_iF_{i-1}\right) \\
&= \left(a_0F_0 + a_1F_1 + \sum_{2\le i<r} a_iF_i\right)\varphi + a_0F_{-1} + a_1F_0 + a_2F_1 + \sum_{2\le j<r-1} a_{j+1}F_j \\
&= (a_1 + [a_r\cdots a_2]_F)\varphi + [a_r\cdots a_3]_F + a_0 + a_2. \qquad (2)
\end{aligned}$$

Similarly,

$$\begin{aligned}
\sum_{1\le i\le s} b_i\varphi^{-i} &= \sum_{1\le i\le s} b_i(F_{-i}\varphi + F_{-i-1}) \\
&= \left(\sum_{1\le i\le s} b_iF_{-i}\right)\varphi + \sum_{1\le i\le s} b_iF_{-i-1} \\
&= [b_s\cdots b_1]_{-F}\,\varphi + [b_s\cdots b_1 0]_{-F}. \qquad (3)
\end{aligned}$$

Thus, the left part of a φ-representation is easily expressed as a linear combination $c\varphi + d$, in terms of shifted Zeckendorf representations, and the right part of a φ-representation is easily expressed as a linear combination $c'\varphi + d'$, in terms of shifted negaFibonacci representations. Then $n = (c+c')\varphi + d + d'$ implies that $n = [a_{r-1}\cdots a_1a_0.b_1b_2\cdots b_s]$ if and only if

$$\begin{aligned}
c' &= -c \qquad (4) \\
n &= d + d' \qquad (5)
\end{aligned}$$

both hold.

These are conditions that can be expressed in first-order logic, and hence we can build an automaton to check them. In `Walnut` we can build this automaton from smaller, easily-digestible pieces, as we now describe.

It may be helpful to have a short summary of `Walnut` syntax.

- `reg` defines a regular expression
- `def` defines an automaton based on a logical formula
- `eval` evaluates a formula with no free variables and returns `TRUE` or `FALSE`
- `E` represents the existential quantifier $\exists$; `A` represents the universal quantifier $\forall$

- `&` is logical `AND`, `|` is logical `OR`, `=>` is logical implication; `<=>` is logical `IFF`; `~` is logical `NOT`
- `?msd_fib` instructs `Walnut` that numbers should be expressed in Zeckendorf representation; `?msd_neg_fib` does the same thing for negaFibonacci representation.

2.1 Zeckendorf normalizer

We need a "normalizer" for Zeckendorf expansions; it takes x and y as inputs, with x an arbitrary binary string and y a canonical Zeckendorf expansion of some integer n, and accepts if and only if $n = [x]_F$. Such an automaton can be found, for example, in [3, 33]. It has 5 states (or 4 if one omits the dead state). We call it `fibnorm`.

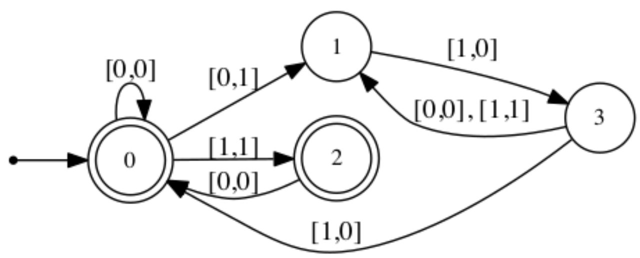

Figure 1: The automaton `fibnorm`.

2.2 NegaFibonacci normalizer

We also need the analogous "normalizer" for negaFibonacci expansions; it takes x and y as inputs, with x an arbitrary binary string and y a canonical negaFibonacci expansion of some integer n, and accepts if and only if $n = [x]_{-F}$. Such an automaton has 5 states (or 4 if one omits the dead state). We call it `negfibnorm`. Correctness is left to the reader.

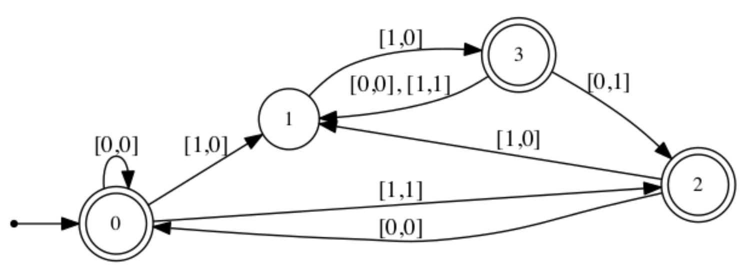

Figure 2: The automaton `negfibnorm`.

2.3 Shifters

As we have seen in Equations (2) and (3), we need to shift some representations left or right. We do this with the following automata:

- `shiftl` takes arguments x and y, and accepts if y is x shifted to the left (and 0 inserted at the right).

- `shiftr` takes arguments x and y, and accepts if y equals the string x shifted to the right (and the least significant digit disappears).

We can easily build these with regular expressions as follows:

```
reg shiftl {0,1} {0,1} "([0,0]|[0,1][1,1]*[1,0])*":
reg shiftr {0,1} {0,1} "([0,0]|[1,0][1,1]*[0,1])*(()|[1,0][1,1]*)":
```

2.4 Last bits

Again, as we have seen in Equations (2) and (3), we need the ability to extract the last, second-to-last, or third-to-last bits of a string. These can easily be specified with regular expressions. We create three: `lstbit1`, `lstbit2`, and `lstbit3`. Here `lstbitn` takes arguments x and y, where $x \in \{0,1\}^*$ and y is the representation of either 0 or 1, and accepts if y is the n'th-last bit of x (or 0 if $|x| < n$).

```
reg lstbit1 {0,1} msd_fib "()|(([0,0]|[1,0])*([0,0]|[1,1]))":
reg lstbit2 {0,1} msd_fib "()|[0,0]|[1,0]|(([0,0]|[1,0])*
(([0,0]([0,0]|[1,0]))|([1,0]([0,1]|[1,1]))))":
reg lstbit3 {0,1} msd_fib "()|[0,0]|[1,0]|(([0,0]|[1,0])
([0,0]|[1,0]))|(([0,0]|[1,0])*(([0,0]([0,0]|[1,0])
([0,0]|[1,0]))|([1,0]([0,0]|[1,0])([0,1]|[1,1]))))":
```

2.5 Converting from negaFibonacci to Zeckendorf

Finally, we will need to be able to convert negaFibonacci representation to Zeckendorf representation. We need two automata:

- `fibnegfib` takes two arguments x and y, where $x = (n)_F$ for some $n \geq 0$ and $y = (m)_{-F}$, and accepts if $n = m$. It has 12 states.

- `fibnegfib2` takes two arguments x and y, where $x = (n)_F$ for some $n \geq 0$ and $y = (m)_{-F}$, and accepts if $n = -m$. It has 9 states.

2.6 Constructing the first Frougny-Sakarovitch automaton

We now have all the pieces we need to construct the first Frougny-Sakarovitch automaton directly from the formulas (2) and (3). We do this with the following `Walnut` code:

```
def phipartleft "?msd_fib Er,s,y,b $shiftr(x,r) & $shiftr(r,s) &
$fibnorm(s,y) & $lstbit2(x,b) & z=y+b":
def intpartleft "?msd_fib Er,s,t,y,b,c $shiftr(x,r) & $shiftr(r,s) &
$shiftr(s,t) & $fibnorm(t,y) & $lstbit1(x,b) & $lstbit3(x,c) &
z=y+b+c":
def phipartright "?msd_neg_fib $negfibnorm(x,z)":
def intpartright "?msd_neg_fib Er $shiftl(x,r) & $negfibnorm(r,z)":

def frougny1 "?msd_fib Et1,t2 $phipartleft(x,t1) &
$phipartright(y,?msd_neg_fib t2) & $fibnegfib2(t1,t2)":
# accepts if the phi part of inputs x and y sums to 0

def frougny2 "?msd_fib Et1, t2, x2 $intpartleft(x,t1)
& $intpartright(y,?msd_neg_fib t2) & (((?msd_neg_fib t2<0) &
$fibnegfib2(x2,?msd_neg_fib t2) & ?msd_fib t1=n+x2)|
(((?msd_neg_fib t2>=0) & $fibnegfib(x2,?msd_neg_fib t2) &
?msd_fib n=t1+x2)))":
# accepts if the integer parts of inputs x and y sum to n

def frougny3 "?msd_fib $frougny1(x,y) & $frougny2(n,x,y)":
fixleadzero frougny frougny3:
```

Here `phipartleft` (resp., `intpartleft`) computes the coefficient of φ (resp., the coefficient of 1) in the expression (2). Similarly, `phipartright` (resp., `intpartright`) does the same thing for (3). The command `fixleadzero` allows one to remove useless leading zeroes from the resulting automaton.

Next, `frougny1` checks the condition (4) and `frougny2` checks the condition (5). Finally, `frougny` checks the conjunction of the two conditions. The resulting automaton `frougny` has 116 states.

If we want a simpler automaton, namely, where $x.y^R$ must be a canonical representation (no occurrences of 11 allowed), we can intersect the automaton `frougny` with a simple automaton imposing this condition on x and y. This gives us a second Frougny-Sakarovitch automaton `saka` with only 39 states.

```
def saka2 "?msd_fib $frougny(n,?msd_fib x,?msd_fib y) & $no11xy(x,y)":
fixleadzero saka saka2:
```

With this automaton we can easily prove the following characterization of the canonical φ-representation of F_n, which is the content of both [23, Corollary 6] and [13, Proposition 3.1].

Proposition 2.1. *We have*

- $(F_{4i})_\varphi = 100(0100)^{i-1}.(0100)^{i-1}1$, $i \geq 1$;

- $(F_{4i+1})_\varphi = 1000(1000)^{i-1}.(1000)^{i-1}1001,\ i \geq 1;$
- $(F_{4i+2})_\varphi = 1(0001)^i.(0001)^i,\ i \geq 0;$
- $(F_{4i+3})_\varphi = 10(0010)^i.(0010)^i 01,\ i \geq 0.$

Proof. We apply the automaton `saka` to the Fibonacci numbers F_n for $n \geq 2$:

```
reg isfib msd_fib "0*10*":
def prop31a "?msd_fib $saka(n,x,y) & $isfib(n)":
fixleadzero prop31 prop31a:
```

The resulting automaton is depicted in Figure 3.

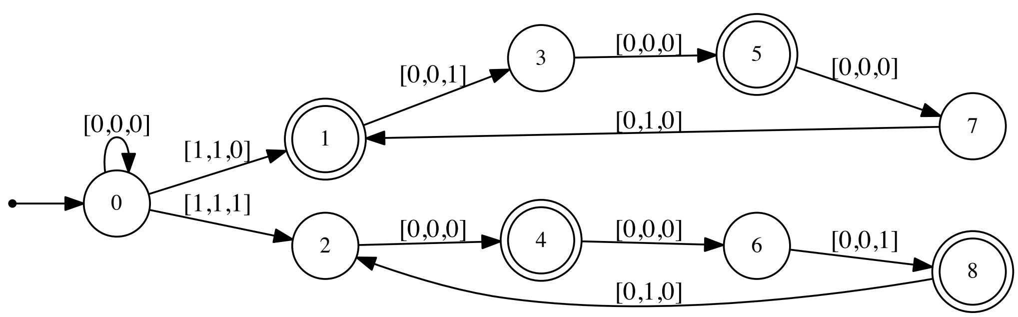

Figure 3: Canonical φ-representation of F_n.

Inspection of the automaton shows that, ignoring leading zeros, there are essentially four different acceptance paths:

- $[1,1,0][0,0,1][0,0,0]([0,0,0][0,1,0][0,0,1][0,0,0])^*$.
- $[1,1,1][0,0,0][0,0,0][0,0,1]([0,1,0][0,0,0][0,0,0][0,0,1])^*$;
- $[1,1,0]([0,0,1][0,0,0][0,0,0][0,1,0])^*$;
- $[1,1,1][0,0,0]([0,0,0][0,0,1][0,1,0][0,0,0])^*$;

These correspond to the cases $n \equiv 0, 1, 2, 3 \pmod 4$, respectively, and prove the result. $\square$

We can also see from Figure 3 a partial explanation for the periodicity (mod 4) in the Frougny-Sakarovitch construction.

Exactly the same techniques can be used to prove a similar result for the Lucas numbers, defined by $L_0 = 2$, $L_1 = 1$, and $L_n = L_{n-1} + L_{n-2}$. We omit the details.

Theorem 2.2. *We have*

- $(L_{2i})_\varphi = 10^{2i}.0^{2i-1}1$, $i \geq 1$;
- $(L_{2i+1})_\varphi = (10)^i1.(01)^i$, $i \geq 0$.

One very useful feature of the automaton `saka` is we can use it to completely characterize the folded representations of all natural numbers.

Theorem 2.3. *The string $a_{r-1}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-r}$ is the φ-representation of an integer $n \geq 0$ if and only if*

$$[a_{r-1}, a_{-r}][a_{r-2}, a_{1-r}]\cdots[a_1, a_{-2}][a_0, a_{-1}]$$

is accepted by the automaton in Figure 4.

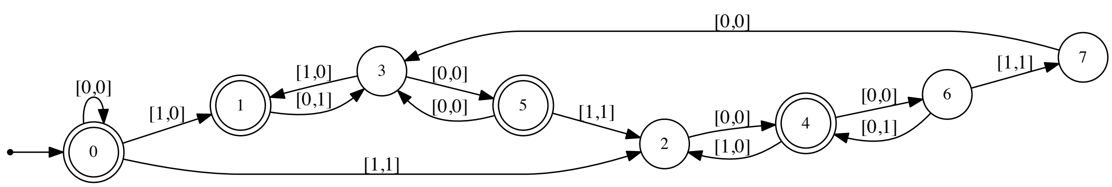

Figure 4: Automaton accepting the folded representations of all $n \geq 0$.

Proof. We use the `Walnut` code

```
def berg "?msd_fib En $saka(n,x,y)":
```

□

Let us now turn to characterizing the right part of canonical φ-representations. Dekking [9, Theorem 7.3] proved the following result:

Theorem 2.4. *If a binary string is of even length, ends in a* 1*, and contains no* 11*, then it appears as the right part of the φ-representation of some natural number n.*

Proof. To prove this, we make a `Walnut` regular expression for strings of even length ending in a 1. Some care is needed here, for two reasons: first, the representation of the right part is “folded”, so we’re really talking about beginning with a 1 instead of ending. Second, our automata may have leading zeros in the (reversed) representation of the right part, for the reasons mentioned previously. Keeping these two points in mind, we use the following `Walnut` code:

```
reg has11 {0,1} "(0|1)*11(0|1)*":
def rightp2 "?msd_fib En,x $saka(n,x,y)":
fixleadzero rightp rightp2:
reg evenl {0,1} "(0*)|(0*1(0|1)((0|1)(0|1))*)":
eval thm73 "?msd_fib Ay $rightp(y) <=> ($evenl(y) & ~$has11(y))":
```

And `Walnut` returns `TRUE`. □

In a similar way, we can prove a companion theorem for the left part of canonical φ-representations. This appears to be new.

Theorem 2.5. *A binary string x appears as the left part of a φ-representation of some n if and only if x has no* 11 *and does not have a suffix of the form* $1(00)^i1$.

Proof. We use the following `Walnut` code:

```
def leftp2 "?msd_fib En,y $saka(n,x,y)":
fixleadzero leftp leftp2:
reg suff {0,1} "(0|1)*1(00)*1":
eval claim "?msd_fib Ax $leftp(x) <=> ((~$suff(x))&(~$has11(x)))":
```

And `Walnut` returns `TRUE`. □

It is easy to see that every positive integer has infinitely many (possibly non-canonical) φ-representations; for example, we may write

$$2 = [10.01]_\varphi = [10.0011]_\varphi = [10.001011]_\varphi = \cdots .$$

However, there can only be finitely many distinct left parts among all of these. How many are there? We can use `Walnut` to "automatically" compute a special kind of formula for this quantity, called a linear representation.

A *linear representation* for a function f from Σ^* to $\mathbb{N}$ is a triple (v, γ, w), where v is a t-element row vector, γ is a $t \times t$ matrix-valued morphism, and w is a t-element column vector, such that $f(x) = v\gamma(x)w$. The *rank* of such a representation is defined to be t. Two linear representations are said to be equivalent if they compute the same function of x. There is an algorithm that, given a linear representation, finds an equivalent minimal one: that is, one of minimal rank [4, Chapter 2]. A nice feature of linear representations is that they allow efficient computation of the corresponding function. For more information about linear representations, see [4].

Let us find a linear representation for $p(n)$, the number of distinct left parts of φ-representations for n. We can do this with the following `Walnut` command:

```
def numleft n "?msd_fib Ey $frougny(n,x,y)":
```

This gives us a linear representation of rank 112 to compute $p(n)$, which can be minimized to a linear representation of rank 28. The first few terms of this sequence are given in Table 2. This is sequence A362970 in the OEIS.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$p(n)$	1	2	2	3	3	3	3	4	5	4	5	4	4	5	6	6	5	4	5	7

Table 2: First few values of $p(n)$.

3 Base-φ sum of digits

Dekking [12] studied the sum-of-digits function for base-φ representations. Denoting by $n = \sum_{L \leq i \leq R} d_i \varphi^i$ the canonical base-φ representation, then we define $s(n) = \sum_{L \leq i \leq R} d_i$. The sequence $s(n)$ is sequence A055778 in the OEIS.

Let us compute a linear representation for $s(n)$. The idea is to define an automaton with two binary inputs x, y that accepts precisely if y has a single 1 in a position matching a 1 in x. Then the number of pairs (x, y) accepted corresponds to the number of 1's in x. We can then find linear representations for the left and right parts of $(n)_\varphi$.

```
reg match1 {0,1} {0,1} "([0,0]|[1,0])*[1,1]([0,0]|[1,0])*":
def countl "?msd_fib Ex,y $saka(n,x,y) & $match1(x,t)":
def countr "?msd_fib Ex,y $saka(n,x,y) & $match1(y,u)":
def sl n "?msd_fib $countl(n,t)":
def sr n "?msd_fib $countr(n,u)":
```

This gives a linear representation of rank 99 for s_L and one of rank 82 for s_R. When minimized, we get a representation of rank 19 for s_L and one of rank 21 for s_R. We can then find a linear representation of rank 40 for s by summing these two linear representations. This can be minimized to a linear representation for s of rank 21.

We now show how to apply our automaton `saka` to solve an open problem due to Dale Gerdemann in 2012, and stated in the text for sequence A055778 in the OEIS. Once again, the proof is more or less purely computational, but here we need some additional computational tools.

Theorem 3.1. *The sum of the digits for n in Zeckendorf representation is always at most the sum of the digits for n in canonical base-φ representation.*

Proof. Let $|x|_a$ denote the number of occurrences of the symbol a in the string x. It is easy, using the techniques above, to obtain a linear representation of rank 23 for the difference $|(n)_\varphi|_1 - |(n)_F|_1$. However, there is no algorithm for determining whether a given linear representation computes a function that is always non-negative, so this linear representation does not seem to be useful to solve the problem. Instead, we use a different technique.

Recall that the automaton `saka` takes three inputs in parallel—n, x, and y—and accepts if $x.y^R$ is the canonical φ-representation for n. The transitions of this automaton are therefore of the form $[a, b, c]$ where $a, b, c \in \{0, 1\}$. Suppose an input $a_1 \cdots a_t$, $b_1 \cdots b_t$, $c_1 \cdots c_t$ is accepted. Then $|(n)_\varphi|_1 - |(n)_F|_1 = \sum_{1 \leq i \leq t} (b_i + c_i) - a_i$.

Therefore, we can take the automaton for `saka` and form a directed graph G from it by replacing the transition on the triple $[a, b, c]$ with a weight labelled $b+c-a$. The assertion that $|(n)_\varphi|_1 - |(n)_F|_1$ is always non-negative then follows from the following stronger claim:

$$\text{Every path from the initial state to any state has a non-negative sum of weights.} \quad (6)$$

But this is precisely the minimum-weight path problem solved by the Bellman-Ford algorithm. We ran this algorithm on G and verified that assertion (6) holds. $\square$

Next, one can consider those n for which $|(n)_F|_1 = |(n)_\varphi|_1$. These correspond to the paths of weight 0 in the graph G just discussed. One can easily verify, by explicit enumeration, that all simple cycles of G (i.e., no repeated vertices except at the beginning and end of the cycle) have weight ≥ 0, and furthermore one can determine all those of weight 0 (there are 26 of them).

Let P be an accepting path of weight 0. If P contains a cycle C, by above the weight of C is non-negative, by removing C, we get an accepting path of P' of weight ≤ 0. But there are no negative-weight accepting paths, so C must also be of weight 0 and hence so is P'. We can then rule out possible cycles by considering the lowest-weight path from the initial state to the first state of C and similarly from the first state of C to a final state; these weights must sum to 0. Of the 26 possible cycles, only three possibilities remain (up to choice of the initial state of the cycle). By considering the possible cycles and how they join up, we get the automaton in Figure 5.

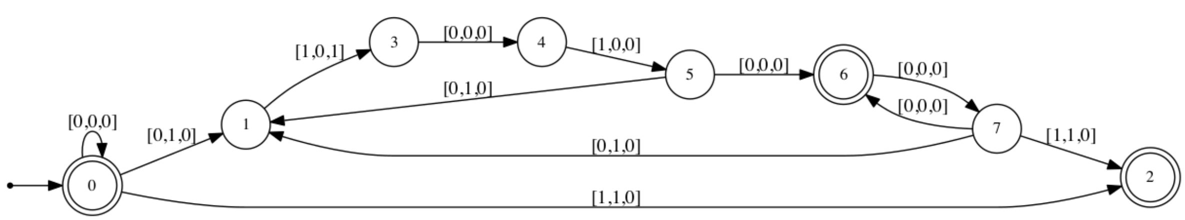

Figure 5: Automaton for weight-0 paths.

To get the accepted set of n, we project to the first coordinate and determinize the resulting automaton. Thus we have proved the following result:

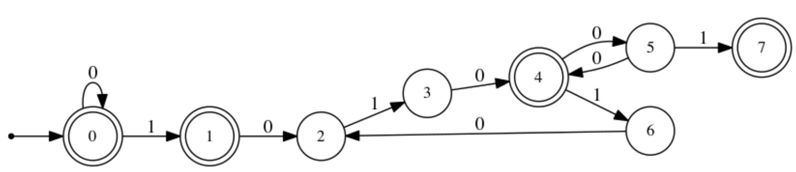

Figure 6: Automaton for weight-0 paths.

Theorem 3.2. *The automaton in Figure 6 accepts precisely those n in Fibonacci representation for which $|(n)_F|_1 = |(n)_\varphi|_1$.*

Similarly, one can study the sum of all the bits of the base-φ representation, taken modulo 2.

```
reg sum2 {0,1} {0,1} "([0,0]|[1,1])*(([0,1]|[1,0])([0,0]|[1,1])*
([0,1]|[1,0])([0,0]|[1,1])*)*([0,1]|[1,0])([0,0]|[1,1])*":
# sum of all the bits of x and y, mod 2

def sdpr2 "?msd_fib Ex,y $saka(n,x,y) & $sum2(x,y)":
# sum of digits of phi-rep mod 2 is 1
```

This gives a 73-state automaton for this binary sequence. It is sequence A330037 in the OEIS.

The automata `frougny` and `saka` that we have constructed make it very easy to compute other quantities related to φ-representations. For example, let us define $s_L(n)$ to be the sum of the bits of the left part of the φ-representation of n. This is apparently a new sequence, never before studied; it is sequence A362716 in the OEIS. (Dekking [11, 12] studied the sum $s(n)$ of *all* the bits of the φ-representation.)

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$s_L(n)$	0	1	1	1	2	1	2	1	2	2	2	3	1	2	2	3	2	3	1	2

Table 3: First few values of $s_L(n)$.

We can easily find a linear representation for $s_L(n)$ as follows:

```
reg match1 {0,1} {0,1} "([0,0]|[1,0])*[1,1]([0,0]|[1,0])*":
def count1 "?msd_fib Ex,y $saka(n,x,y) & $match1(x,t)":
def sl n "?msd_fib $count1(n,t)":
```

This gives us a linear representation of rank 99, which can be minimized to the following linear representation (v, γ, w) of rank 19:

$$v = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\gamma(0) = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&-1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&-1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&-1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&-1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&-1&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&-1&0&0&0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&-1&0&0&0&0&0&0&0&0&2&0&0
\end{bmatrix}
\quad
\gamma(1) = \begin{bmatrix}
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&-1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&-1&0&0&0&0&-1&0&0&0&1&0&0&1\\
0&1&0&0&0&0&-1&0&0&0&0&0&0&0&0&1&0&0&0
\end{bmatrix}
\quad
w = \begin{bmatrix} 0\\1\\1\\1\\2\\1\\2\\1\\2\\2\\2\\3\\1\\2\\3\\3\\1\\3\\4 \end{bmatrix}.$$

Similarly, one can study the sum-of-digits function $s_R(n)$ for the right part of the canonical φ-representation. We prove a new theorem about it:

Theorem 3.3. *The difference $d(n) := s_R(n+1) - s_R(n)$ is a Fibonacci-automatic sequence taking values in $\{-1, 0, 1\}$ only.*

Proof. We use the following `Walnut` code.

```
reg match1 {0,1} {0,1} "([0,0]|[1,0])*[1,1]([0,0]|[1,0])*":
def countr "?msd_fib Ex,y $saka(n,x,y) & $match1(y,u)":
def sr n "?msd_fib $countr(n,t)":
def sr1 n "?msd_fib $countr(n+1,t)":
```

This produces linear representations for $s_R(n)$ (rank 82; minimizes to rank 21) and $s_R(n+1)$ (rank 64; minimizes to rank 17). From this we can produce a linear representation for $d(n) = s_R(n+1) - s_R(n)$ of rank 38, which minimizes to one of rank 17. We can then do the "semigroup trick" [34, §4.11] to show that the resulting sequence is Fibonacci automatic. The automaton is depicted in Figure 7. ∎

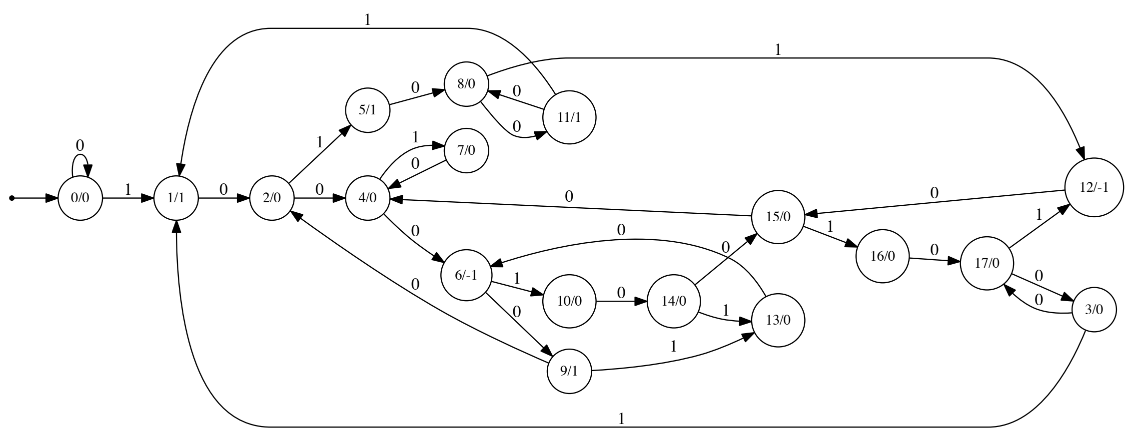

Figure 7: Fibonacci automaton for $d(n)$.

4 Length of base-φ representations

Similarly, one can study the length $\ell(n)$ of the left part of $(n)_\varphi$ (or, alternatively, one more than the exponent of the highest power of φ appearing in $(n)_\varphi$).

This is sequence A362692 in the OEIS.

Here the idea is to create a simple automaton, `onepos`, that accepts binary strings x and t if and only if t contains a single 1 and it occurs at or to the right of the first 1 in x. Then we simply count the number of such t:

```
def intlength "?msd_fib Ex,y $saka(n,x,y) & $onepos(x,t)":
def intl n "?msd_fib $intlength(n,t)":
```

The resulting linear representation can be minimized to the following linear representation of rank 9:

$$v^T = \begin{bmatrix} 1\\0\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix} \qquad \gamma(0) = \begin{bmatrix} 1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&-1&0&2&0&0&0\\ 0&0&0&-1&0&2&0&0&0\\ 0&0&0&-2&1&3&-1&0&0\\ 0&0&0&-2&0&3&0&0&0 \end{bmatrix} \qquad \gamma(1) = \begin{bmatrix} 0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&-1&0&2\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \qquad w = \begin{bmatrix} 0\\1\\2\\3\\3\\4\\4\\5\\5 \end{bmatrix}.$$

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$\ell(n)$	0	1	2	3	3	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7

Table 4: First few values of $\ell(n)$.

Inspection suggests that $\ell(n)$ is an increasing sequence and the gaps between successive terms are only 0 and 1.

We can prove this by finding the linear representation for $\ell(n) - \ell(n-1)$ and then using the "semigroup trick". This gives a Fibonacci DFAO, depicted in Figure 8, computing $\ell(n) - \ell(n-1)$.

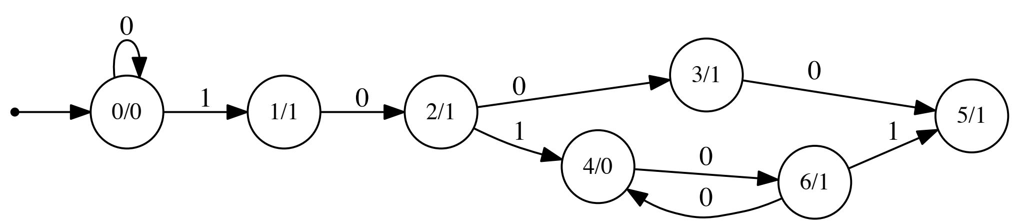

Figure 8: Fibonacci DFAO for first difference of $\ell(n)$.

Inspection of this automaton and observing that the Fibonacci representation of the Lucas numbers are of the form $1010*$, gives the following result:

Theorem 4.1. *We have $\ell(n) - \ell(n-1) = 1$ if and only if $n = 1$ or $n = L_{2i}$ or $n = L_{2i-1} + 1$ for $i \geq 1$.*

This is implied by a result of Sanchis and Sanchis [32, Thm. 2.1].

5 Individual bits

We can also prove theorems about individual bits of the φ-representation. Let

$$n = \sum_{L \le i \le R} d_i(n)\varphi^i$$

be the canonical φ-representation of the natural number n and let

$$\alpha = (3-\varphi)/5 = (5-\sqrt{5})/10.$$

The Sturmian sequence based on α is defined to be $w(n) = \lfloor (n+1)\alpha \rfloor - \lfloor n\alpha \rfloor$; this is sequence A221150 in the OEIS. For more about Sturmian sequences, see, e.g., [27, Chap. 2].

The following result characterizes $d_0(n)$ and $d_1(n)$. Some previous papers have characterized these two sequences (see [10, 24]), but it appears that the specific characterization we present here is new.

Theorem 5.1. *We have*

(a) $d_0(n+1) = w(n)$ *for* $n \ge 1$*;*

(b) $d_1(n) = w(n+1)$ *for* $n \ge 0$.

Proof.

(a) We can compute $w(n)$ with `Walnut` using the existing code `phin`, which computes $\lfloor \varphi n \rfloor$ (see [34, p. 278]). The resulting automaton computes w in a "synchronized" fashion; that is, the inputs are n and x, both in Zeckendorf representation, and the automaton accepts if and only if $x = w(n)$. Now

$$\lfloor \alpha n \rfloor = \left\lfloor \frac{3-\varphi}{5} n \right\rfloor = \left\lfloor \frac{\lfloor (3-\varphi)n \rfloor}{5} \right\rfloor = \left\lfloor \frac{3n - 1 - \lfloor \varphi n \rfloor}{5} \right\rfloor$$

for $n \ge 1$. It follows that a synchronized automaton for $w(n)$ can be computed by the following `Walnut` code.

```
reg shift {0,1} {0,1} "([0,0]|[0,1][1,1]*[1,0])*":
def phin "?msd_fib (s=0 & n=0) | Ex $shift(n-1,x) & s=x+1":
def alphan "?msd_fib Ex $phin(n,x) & z=(3*n-(x+1))/5":
def w "?msd_fib Ez $alphan(n+1,z+1) & $alphan(n,z)":
```

Similarly, we can construct a synchronized automaton for d_0 and verify claim (a) as follows:

```
def d0 "?msd_fib Ex,y $saka(n,x,y) & $lstbit1(x,1)":
eval checka "?msd_fib An (n>=1) => ($w(n) <=> $d0(n+1))":
```

and `Walnut` returns `TRUE`.

(b) We use the following `Walnut` code, to compute a synchronized automaton for d_1 and check the assertion:

```
def d1 "?msd_fib Ex,y $saka(n,x,y) & $lstbit2(x,1)":
eval checkb "?msd_fib An ($w(n+1) <=> $d1(n))":
```

and `Walnut` returns `TRUE`.

$\square$

Now let us turn to $d_{-1}(n)$. Of course, this is a Fibonacci-automatic sequence (and can be computed by a DFAO of 13 states). We can find a surprising connection for $d_{-1}(n)$ by consulting the OEIS, as follows.

Instead of representations involving sums of Fibonacci numbers, we can consider sums of Lucas numbers L_n, defined previously. The following result was proved by Brown [6]: every natural number can be written as a sum $\sum_{0 \leq i \leq t} e_i L_i$, where $e_i \in \{0, 1\}$. We use the shorthand notation $[e_t \cdots e_1 e_0]_L$ for this sum. However, this representation is not always unique, even if we impose the natural condition that

$$e_i e_{i+1} \neq 1 \text{ for all } i. \tag{7}$$

For example, we have $[1010]_L = L_3 + L_1 = 5 = L_2 + L_0 = [101]_L$. Recently, Chu, Luo, and Miller [8] proved that if we impose Condition (7), then there are at most two distinct Lucas representations for each n. The set of n with two distinct Lucas representations forms sequence A342089 in the OEIS.

Theorem 5.2. *For all $n \geq 0$ we have $d_{-1}(n) = 1$ if and only if n has exactly two distinct Lucas representations.*

Proof. We start by constructing an automaton to evaluate a Lucas representation. It takes a number n in Zeckendorf representation and a binary string x as inputs and accepts if $[x]_L = n$.

To do so, we use the classic identity $L_n = F_{n+1} + F_{n-1}$ for all n. It follows that $[xabc]_L = [xab]_F + [x]_F + a + 2c$ for for binary strings x and $a, b, c \in \{0, 1\}$. We can therefore compute it as follows:

```
def luc2fib "?msd_fib Eu,v,w,a,c,y,z $shiftr(x,u) & $shiftr(u,v) &
$shiftr(v,w) & $lstbit3(x,a) & $lstbit1(x,c) & $fibnorm(u,y) &
$fibnorm(w,z) & n=y+z+a+2*c":
```

Next, we create an automaton to accept the Fibonacci representation of those n with two different Lucas representations.

```
reg has11 {0,1} "(0|1)*11(0|1)*":
reg equal {0,1} {0,1} "([0,0]|[1,1])*":
def tworep "?msd_fib Ex,y (~$has11(x)) & (~$has11(y)) &
(~$equal(x,y)) & $luc2fib(n,x) & $luc2fib(n,y)":
```

Finally, we create a synchronized automaton for $d_{-1}(n)$ and check its equality with sequence A342089:

```
def dm1 "?msd_fib Ex,y $saka(n,x,y) & $lstbit1(y,1)":
eval checkseq "?msd_fib An ($dm1(n) <=> $tworep(n))":
```

and `Walnut` returns `TRUE`. $\square$

Remark 5.3. Hart and Sanchis [24] proved that the limiting frequency of 1's in $(d_{-1}(n))_{n\geq 0}$, namely $\lim_{n\to\infty} \frac{1}{n} \sum_{0\leq i<n} d_{-1}(n)$, is $\gamma := 1/(3\varphi + 1) \doteq 0.17082$. We can easily prove this with `Walnut` as follows: first, define $d'(n) = \sum_{0\leq i\leq n} d_{-1}(i)$. Then we can "guess" a synchronized automaton for $d'(n)$ using the method outlined in [34, §10.15]. Next, we verify that our guess is correct using the automaton `dm1` computed above. Finally, we show with `Walnut` that $|d'(n) - w'(n)| \leq 1$, where $w'(n)$ is the Sturmian sequence corresponding to γ. We omit the details.

We now turn to considering what Dekking and van Loon [14] called "vertical runs of 1's". Given $i \in \mathbb{Z}$, we saw above that $d_i(n)$ is the coefficient of α^i in the canonical φ-representation of n. A *vertical run of 1's* takes place starting at position n if

$$d_i(n-1) = 0, \quad d_i(n) = 1, \quad d_i(n+1) = 1, \ldots, d_i(n+t-1) = 1, \quad d_i(n+t) = 0.$$

and we say the length of the run is t. (If $n = 0$ we assume $d_i(-1) = 0$ for all i.) The name comes from thinking of the base-φ representation of the numbers $d_i(n-1), \ldots, d_i(n+t)$ written vertically, with the i'th position lined up. For example, for $i = 4$ and $n = 7$ we have a vertical run of length 5, as illustrated in Table 5.

n	$(n)_\varphi$
6	0**0**1010.0001
7	0**1**0000.0001
8	0**1**10001.0001
9	0**1**0010.0101
10	0**1**0100.0101
11	0**1**0101.0101
12	1**0**0000.101001

Table 5: A vertical run of 5 1's.

For each i we can determine the possible lengths of vertical runs.

Theorem 5.4. *For the canonical φ-representation the vertical runs of 1's have length $v(i)$, as follows:*

$$v(i) = \begin{cases} L_{i-1} + (-1)^i, & \text{if } i \geq 1; \\ 1, & \text{if } i = 0; \\ L_{-i}, & \text{if } i < 0. \end{cases}$$

Proof. We can create a `Walnut` formula that determines the possible lengths of vertical runs for each $i \geq 0$, as follows:

```
def matchfib "?msd_fib Ex,y $saka(n,x,y) & $isfib(t) & $match1(x,t)":
def verticalr "?msd_fib (i>0) & En (~$matchfib(n-1,t)) &
(~$matchfib(n+i,t)) & Aj (j<i) => $matchfib(n+j,t)":
```

The resulting automaton accepts the Zeckendorf representation of $v(i)$ in parallel with a binary string with exactly one 1, occurring i bits from the right end. It is displayed in Figure 9.

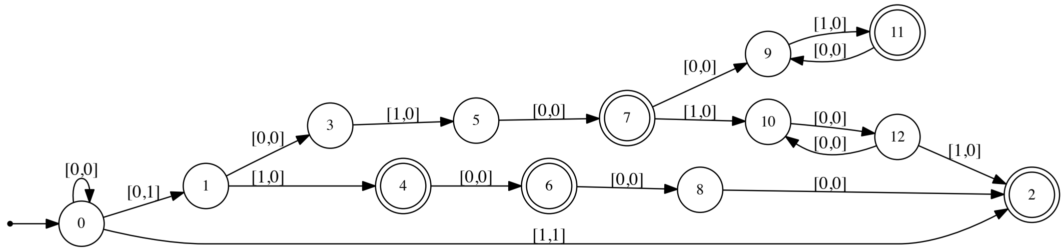

Figure 9: Automaton for $v(i)$, $i > 0$.

As one can see by inspecting the automaton in Figure 9, the acceptance paths, ignoring leading 0's, are of the following types:

(a) $[1,1]$

(b) $[0,1][1,0]$

(c) $[0,1][1,0][0,0]$

(d) $[0,1][0,0][1,0][0,0]$

(e) $[0,1][1,0][0,0][0,0][0,0]$

(f) $[0,1][0,0][1,0][0,0]([0,0][1,0])^*$

(g) $[0,1][0,0][1,0][0,0][1,0]([0,0][0,0])^*[0,0][1,0]$

These correspond to the following accepted inputs:

(a) $i = 0$, $v(0) = 1$;

(b) $i = 1$, $v(1) = 1$;

(c) $i = 2$, $v(2) = 2$;

(d) $i = 3$, $v(3) = 2$;

(e) $i = 4$, $v(5) = 5$;

(f) $i = 2k + 1$ odd, $v(i) = L_{i-1} - 1$;

(g) $i = 2k$ even, $v(i) = L_{i-1} + 1$.

The case of $i < 0$ can be handled similarly; we omit the details. □

Remark 5.5. Dekking and Van Loon [14] claimed that "there is no such regularity", but Theorem 5.4 would seem to contradict that.

6 Palindromic and anti-palindromic φ-representations

Let us consider natural numbers with palindromic φ-representations, that is, where $n = [x.x^R]_\varphi$. There are two variations: one where we demand that the expansion be canonical, and one where we do not make this assumption.

Let us start with the canonical case. Since the representation that our automaton `saka` uses is "folded", we can find the Zeckendorf representation of those n with palindromic φ-representations with the following `Walnut` code:

```
def palcanon "?msd_fib Ex,y $saka(n,x,y) & $equal(x,y)":
```

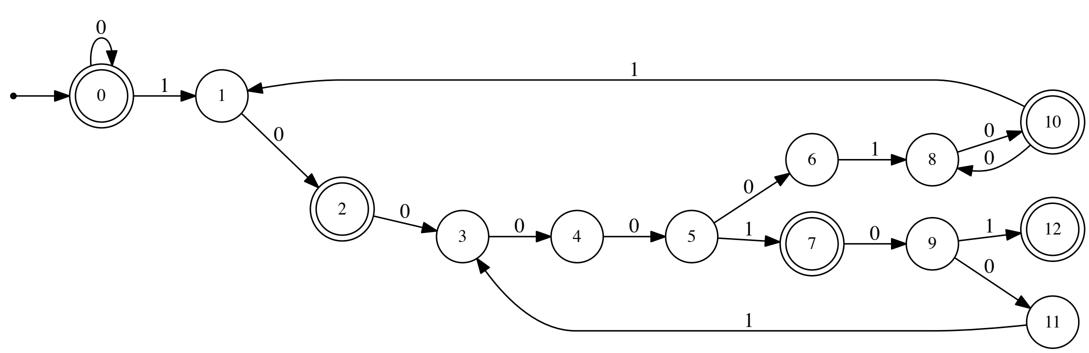

Figure 10: Fibonacci automaton for those n with palindromic φ-representations.

Hence we have shown the following

Theorem 6.1. *The natural number n has a palindromic canonical φ-representation if and only if the 12-state automaton depicted in Figure 10 accepts $(n)_F$.*

The resulting sequence of n accepted by this automaton is

$$2, 14, 36, 38, 94, 96, 246, 248, 260, \ldots$$

and forms sequence A362780 in the OEIS.

Now we turn to the case of allowing non-canonical expansions. Here there are additional examples such as $6 = [1001.1001]$, which is palindromic, non-canonical due to the presence of 11. We can construct an automaton for these n as follows:

```
def pal "?msd_fib Ex,y $frougny(n,x,y) & $equal(x,y)":
```

This proves the following result:

Theorem 6.2. *The natural number n has some palindromic (possibly non-canonical) φ-representation if and only if the 16-state automaton depicted in Figure 11 accepts $(n)_F$.*

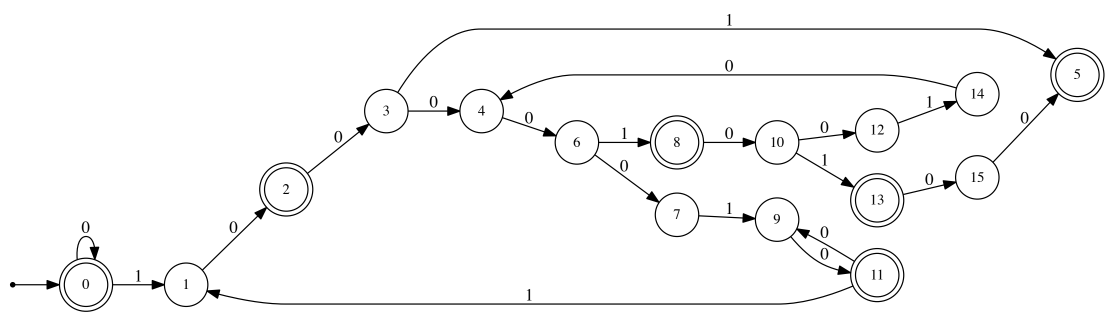

Figure 11: Fibonacci automaton for those n having some palindromic (possibly non-canonical) φ-representations.

The resulting sequence of n accepted by this automaton is

$$2, 6, 14, 36, 38, 94, 96, 100, 246, 248, 252, 260, \ldots$$

and forms sequence A330672 in the OEIS. We remark that the only new examples are those where the only 11 occurs as the 1 at the end of x (and hence at the beginning of x^R), as can be verified with the following `Walnut` code:

```
def pal11 "?msd_fib Ex,y $frougny(n,x,y) & $equal(x,y) & $has11(x)":
```

which accepts nothing.

Next we turn to Shevelev's so-called "φ-antipalindromic numbers"; these are the n for which the canonical base-φ representation of n is of the form $xa.x^R$, where $a \in \{0,1\}$. (The name comes from A178482 and is rather confusing, but we are keeping it.)

```
def shevanti "?msd_fib Ex,y $shiftr(x,y) & $saka(n,x,y)":
```

Theorem 6.3. *Shevelev's φ-antipalindromic numbers n are precisely those for which $(n)_F$ is accepted by the automaton in Figure 12.*

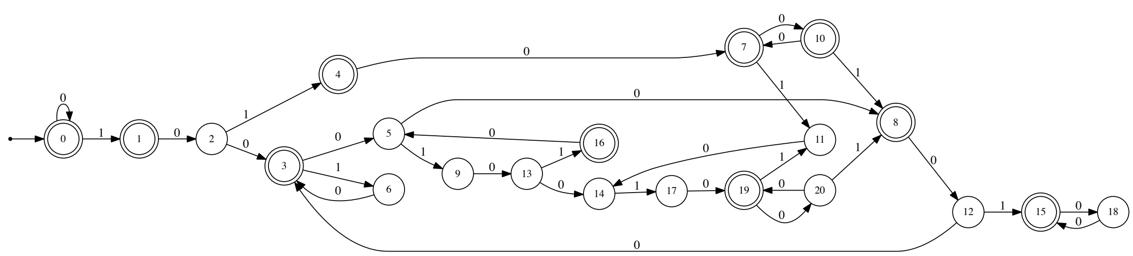

Figure 12: Fibonacci automaton accepting Shevelev's φ-antipalindromic numbers.

The resulting sequence of accepted n,

$$0, 1, 3, 4, 7, 8, 10, 11, 18, 19, 21, 22, 25, 26, 28, 29, 47, \ldots$$

forms sequence A178482 in the OEIS.

Finally, we consider antipalindromic expansions, i.e., expansions of the form $n = [x.\overline{x^R}]$, where the overline denotes a bitwise complement $0 \to 1, 1 \to 0$. Here we have to allow leading zeros in the left part and trailing zeros in the right part.

Theorem 6.4. *There is a 193-state automaton that accepts precisely those $(n)_F$ for which n has a (possibly non-canonical) antipalindromic base-φ expansion.*

Proof. We use the following `Walnut` code to produce the automaton:

```
reg compl {0,1} {0,1} "[0,0]*([0,1]|[1,0])*":
def antip "?msd_fib $frougny(n,x,y) & $compl(x,y)":
```

□

The first few terms of this sequence are

$$1, 3, 4, 5, 6, 8, 11, 13, 14, 15, 16, 21, 23, 29, 31, 33, 35, 37, 39, 41, 43, 45, \ldots.$$

This is apparently a new sequence, never studied before, and is sequence A362781 in the OEIS.

Table 6 gives the first few antipalindromic expansions for these numbers.

n	$[x.y]$
1	1.0
3	0011.0011
4	101.010
5	0110.1001
6	001001.011011
8	001100.110011
11	10101.01010
13	00100001.01111011
14	011011.001001
15	00100100.11011011

Table 6: Antipalindromic expansions.

7 Canonical expansions with fixed number of 1's

As further examples of what can be done, let us consider those n for which $(n)_\varphi$ contains a given fixed number t of 1's, for $2 \le t \le 5$. We use the following code:

```
reg haszero1 {0,1}  "0*":
reg hasone1 {0,1}   "0*10*":
reg hastwo1 {0,1}   "0*10*10*":
reg hasthree1 {0,1} "0*10*10*10*":
reg hasfour1 {0,1}  "0*10*10*10*10*":
reg hasfive1 {0,1}  "0*10*10*10*10*10*":
def canon2 "?msd_fib Ex,y $saka(n,x,y) & (($haszero1(x)&$hastwo1(y))|
($hasone1(x)&$hasone1(y))|($hastwo1(x)&$haszero1(y)))":
def canon3 "?msd_fib Ex,y $saka(n,x,y) & (($haszero1(x)&$hasthree1(y))|
($hasone1(x)&$hastwo1(y))|($hastwo1(x)&$hasone1(y))|
($hasthree1(x)&$haszero1(y)))":
def canon4 "?msd_fib Ex,y $saka(n,x,y) & (($haszero1(x)&$hasfour1(y))|
($hasone1(x)&$hasthree1(y))|($hastwo1(x)&$hastwo1(y))|
($hasthree1(x)&$hasone1(y))|($hasfour1(x)&$haszero1(y)))":
def canon5 "?msd_fib Ex,y $saka(n,x,y) & (($haszero1(x)&$hasfive1(y))|
($hasone1(x)&$hasfour1(y))|($hastwo1(x)&$hasthree1(y))|
($hasthree1(x)&$hastwo1(y))|($hasfour1(x)&$hasone1(y))|
($hasfive1(x)&$haszero1(y)))":
```

We summarize our results below.

Theorem 7.1. *For all $t \ge 0$ there is an automaton recognizing those $(n)_F$ such that $(n)_\varphi$ contains exactly t 1's. For $2 \le t \le 5$ the number of states is given in Table 7.*

t	number of states	sequence in OEIS enumerating $(n)_\varphi$ with t 1's
2	6	A005248
3	9	A104626
4	24	A104627
5	46	A104628

Table 7: Number of states for automaton recognizing $(n)_F$ such that $(n)_\varphi$ contains exactly t 1's.

8 Knott expansions

Recently Dekking and Van Loon [13] studied Knott representations of the positive integers, which are those φ-representation of n that do not end in 011 (followed, perhaps, by an arbitrary number of 0's). For example, 1.11 and 10.01 are Knott representations of 2, but 1.1011 and 10.0011 are not Knott. They gave a method to compute $\mathrm{Tot}^\kappa(n)$, the number of Knott representations of n in terms of the Zeckendorf representation of n, but finding it and expressing it is rather complicated. Here we show that, starting with the first Frougny-Sakarovitch automaton, one can easily find, with `Walnut`, a linear representation for $\mathrm{Tot}^\kappa(n)$ that permits efficient computation. All we have to do is find an automaton to check the Knott condition and ask `Walnut` to compute the appropriate matrices, as follows:

```
def dekking "?msd_fib $frougny(n,x,y) & $knott(x,y)":
def dek n "?msd_fib $dekking(n,x,y)":
```

These give a linear representation (v, γ, w) for $\mathrm{Tot}^\kappa(n)$ of rank 122. This is sequence A289749 in the OEIS.

This linear representation gives us a $O(\log n)$ method to compute $\mathrm{Tot}^\kappa(n)$, and much more. It also allows us to compute closed forms for various kinds of n. For example, we can easily recover the following two results of [13]:

Theorem 8.1. *We have*

(a) $\mathrm{Tot}^\kappa(F_k) = F_k$ *for* $k \geq 1$*;*

(b) $\mathrm{Tot}^\kappa(L_k) = \begin{cases} k, & \text{if } k \text{ odd;} \\ k+1, & \text{if } k \text{ even.} \end{cases}$

Proof. The basic idea has already been explored in detail in a number of works; see [34] for example. The basic idea is that if the Zeckendorf representation of n looks like rs^kt for strings r, s, t, then the linear representation for f evaluates to $v\gamma(r)\gamma(s)^k\gamma(t)w$, and therefore is dependent on the entries of the kth power of the matrix $\gamma(s)$. This, in turn, is governed by the zeros of the minimal polynomial of $\gamma(s)$.

For our Theorem, we have $(F_k)_F = 10^{k-2}$ and $(L_k)_F = 1010^{k-3}$. Therefore, the subsequences $\mathrm{Tot}^\kappa(F_k)$ and $\mathrm{Tot}^\kappa(L_k)$ can be expressed as a linear combination of the powers of the zeros of the minimal polynomial of $\gamma(0)$.

We can ask `Maple` (or any symbolic algebra system) to compute the minimal polynomial of $\gamma(0)$; it is

$$X^3(X^2+1)(X^4+3X^2+1)(X^2+X-1)(X^2-X-1)(X-1)^3(X+1)^3.$$

The zeros other than $0, 1, -1$ are therefore

$$-\varphi i,\ -(1/\varphi)i,\ (1/\varphi)i,\ \varphi i,\ -\varphi,\ 1/\varphi,\ -1/\varphi,\ \varphi.$$

It follows that $\mathrm{Tot}^\kappa(F_k)$ for $k \geq 3$ is a linear combination of the k'th powers of these zeros, together with $k^2, k, 1, k^2(-1)^k, k(-1)^k, (-1)^k$. We can then solve for the coefficients with linear algebra and hence prove both of the desired results. □

Furthermore, we can (almost trivially) obtain new results at will. As an example, consider the following result:

Proposition 8.2. *We have* $\mathrm{Tot}^\kappa(3F_n) = F_{n+2} - F_{n-4}$ *for* $n \geq 4$.

Proof. We use the fact that $(3F_n)_F = 10^n 10$, and the technique above. □

As another example of the power of these techniques, let us compute the average number of Knott expansions for n in the interval $F_i \leq n < F_{i+1}$. The idea was already explained in [34, §9.10]: the canonical Zeckendorf representations for n in the interval $[F_i, F_{i+1})$ consist of those $i-1$ bit numbers that start with 1 and have no occurrence of 11. Suppose (v', γ', w') is a linear representation for Tot^κ that has been modified so that $v'\gamma'(x)w' = 0$ if x has an occurrence of 11; then

$$s_\kappa(i) := \sum_{F_i \leq n < F_{i+1}} \mathrm{Tot}^\kappa(n) = \sum_{\substack{x \text{ contains no } 11 \\ \text{and starts with } 1 \\ |x| = i-1}} v'\gamma'(x)w'$$
$$= v'\gamma'(1)(\gamma'(0) + \gamma'(1))^{i-2} w',$$

and hence $s_\kappa(n)$ is representable as a linear combination of the $i-2$'th powers of the zeros of the minimal polynomial for $\gamma'(0) + \gamma'(1)$. We can then determine the coefficients by solving a linear system. When we do this, we get the following result:

Theorem 8.3. *The average number of Knott expansions for n in the interval $[F_i, F_{i+1})$ is asymptotically $\Theta(\rho^i)$, where $\rho = \zeta/\varphi \doteq 1.5334624126$, and $\zeta \doteq 2.4811943040920156$ is the dominant zero of $X^3 - 2X^2 - 2X + 2$.*

Proof. We use this `Walnut` code to compute the linear representation:

```
def dek2 n "?msd_fib (~$has11(n)) & $dekking(n,x,y)":
```

The result is a linear representation of rank 122, with minimal polynomial of degree 38. We can use the technique of [15, Theorem 6] to refine this minimal polynomial for $s_\kappa(i)$ to

$$(X^3 - 2X^2 - 2X + 2)(X + 1)(X^4 + 3X^2 + 1)(X^2 - X - 1).$$

The dominant zero here is $2.4811943040920156\cdots$ of $X^3 - 2X^2 - 2X + 2$. Dividing by F_{i-1}, the number of summands, gives the result for the average value. $\square$

9 Natural expansions

Dekking and Van Loon also enumerated what they called "natural" φ-representations, which are those expansions $n = [x.y]_\varphi$ where the length of y (not including trailing zeros, of course) is the same as the length of y', where $(n)_\varphi = x'.y'$.

We can do this in the same manner as we constructed the automaton saka above: namely, we simply intersect the first Frougny-Sakarovitch automaton with another automaton that imposes the length condition.

```
reg first1match {0,1} {0,1} "[0,0]*[1,1]([0,0]|[0,1]|[1,0]|[1,1])*":
def natural "?msd_fib Ew,x $saka(n,w,x) & $frougny(n,y,z) &
$first1match(x,z)":
def nat n "?msd_fib $natural(n,y,z)":
```

which produces a linear representation of rank 123 for $\mathrm{Tot}^\nu(n)$, the number of such expansions of length n. Using this linear representation, we can easily recover Theorem 4.3 of Dekking and Van Loon, as follows:

Theorem 9.1. *We have* $\mathrm{Tot}^\nu(F_{2n+1}) = \mathrm{Tot}^\nu(F_{2n+2}) = F_{2n+1}$ *for* $n \geq 0$.

Using the same technique, we can also easily prove an additional result:

Theorem 9.2. *We have* $\mathrm{Tot}^\nu(L_{2n+1}) = 1$ *and* $\mathrm{Tot}^\nu(L_{2n}) = 2n$ *for* $n \geq 1$.

10 Dekking–Van Loon "canonical" expansions

Dekking and Van Loon [14] introduced a different kind of base-φ representation, which they called "canonical". However, to avoid confusion, in this paper, we call them DVL-expansions.

In a DVL expansion, we allow 11, but only as the digits $d_1 d_0$, and then only if $d_{-1} = 0$. We can create an automaton dvl that accepts DVL expansions by adding a restriction dvlcond to frougny, as depicted in Figure 13.

The input is x and y in parallel, corresponding to the φ-representation $x.y^R$, and accepts if it obeys the DVL condition.

We can then accept DVL expansions, as follows:

```
def dvl "?msd_fib $frougny(n,x,y) & $dvlcond(x,y)":
```

The resulting automaton has 48 states and accepts n, x, y iff the DVL expansion of n is $[x.y^R]$.

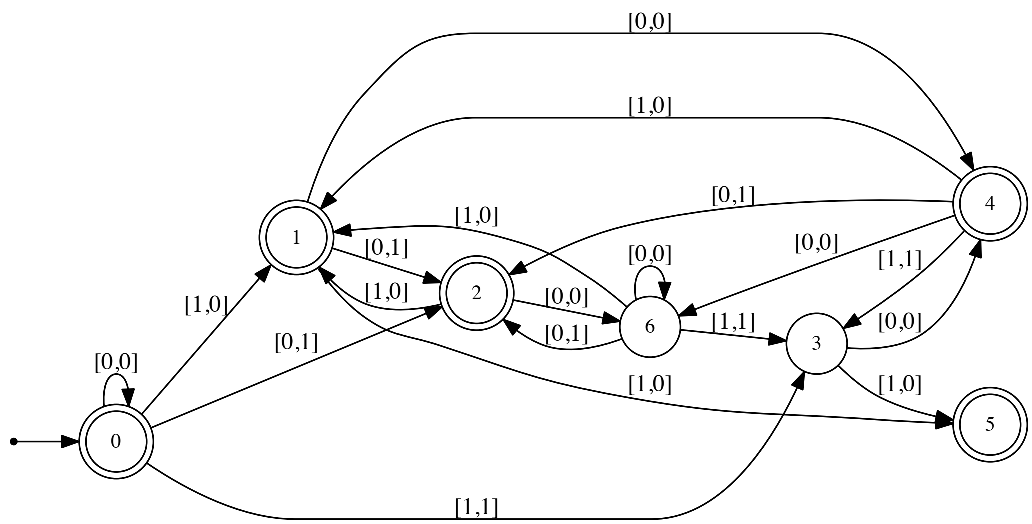

Figure 13: Automaton checking the DVL condition.

We can then use this automaton to "automatically" obtain many of the results in [14]; for example, their Proposition 3.3:

Proposition 10.1. *The canonical φ-representation of an integer n differs from the DVL expansion of n if and only if there exists $m \geq 1$ such that $n = \lfloor(\varphi + 2)m\rfloor$.*

Proof. We use the `Walnut` code

```
reg equal {0,1} {0,1} "([0,0]|[1,1])*":
def differ "?msd_fib Ex,y,w,z $saka(n,x,y) & $dvalcanon(n,w,z) &
((~$equal(x,w))|(~$equal(y,z)))":
eval prop33 "?msd_fib An $differ(n) <=> (Em,t m>=1 &  $phin(m,t) &
n=t+2*m)":
```

and `Walnut` returns `TRUE`. □

Theorem 6.2 of Dekking and Van Loon [14] characterized the lengths of vertical runs in DVL expansions; they used a case-based proof and 3 pages. We can obtain their results directly using exactly the same ideas as for canonical expansions above in Theorem 5.4. For example, for $i \geq 1$, we use the code

```
def matchdvl "?msd_fib Ex,y $dvl(n,x,y) & $isfib(t) & $match1(x,t)":
def verticaldvl "?msd_fib (i>0) & En (~$matchdvl(n-1,t)) &
(~$matchdvl(n+i,t)) & Aj (j<i) => $matchdvl(n+j,t)":
```

which gives an 11-state automaton from which we can easily read off their results.

11 Algebraic integers

As is well-known [37, §2.4], the algebraic integers of $\mathbb{Q}(\sqrt{5})$ are given by $\mathbb{Z}[\varphi]$. The non-negative real members of $\mathbb{Z}[\varphi]$ are precisely those with finite φ-representations [31].

Given a real number of the form $z = m\varphi + n \geq 0$ with $m, n \in \mathbb{Z}$, we would like to compute its φ-representation with an automaton. Since m, n could each possibly be negative, it makes sense to represent them in negaFibonacci representation. We can then compute the appropriate automaton, using Eqsuations (2) and (3) by a small variation of what we did in Section 2. We give the `Walnut code`:

```
def intpartleft "?msd_fib Er,s,t,y,b,c $shiftr(x,r) & $shiftr(r,s) &
$shiftr(s,t) & $fibnorm(t,y) & $lstbit1(x,b) & $lstbit3(x,c) &
z=y+b+c":

def phipartleft "?msd_fib Er,s,y,b $shiftr(x,r) & $shiftr(r,s) &
$fibnorm(s,y) & $lstbit2(x,b) & z=y+b":

def intpartright "?msd_neg_fib Er $shiftl(x,r) & $negfibnorm(r,z)":

def phipartright "?msd_neg_fib $negfibnorm(x,z)":

def genfrou "?msd_neg_fib Ez,t,u,q,r,s $intpartleft(x,?msd_fib z) &
$fibnegfib((?msd_fib z),t) & $intpartright(y,u) & n=t+u &
$phipartleft(x,?msd_fib q) & $phipartright(y,r) &
$fibnegfib((?msd_fib q),s) & m=r+s":
# 536 states

def canfrou "?msd_neg_fib $genfrou(m,n,x,y) & $no11xy(x,y)":
# 259 states
```

Here `genfrou` does not assume that the expansion given by $[x.y^R]$ is canonical, but `canfrou` does.

As an application, let us find the algebraic numbers $m\varphi + n$ such that the left and right parts of their base-φ representation have the same length.

Theorem 11.1. *There is a 96-state automaton that accepts those pairs m, n, in negaFibonacci representation, such that $(m\varphi + n)_\varphi$ has left and right parts of the same length.*

Proof. We use the following `Walnut` code:

```
def samelen "?msd_neg_fib Ex,y $canfrou(m,n,x,y) & $first1match(x,y)":
```

□

12 A final word

`Walnut` is available for free download at

https://cs.uwaterloo.ca/~shallit/walnut.html.

For the definitions of `Walnut` automata that we did not give explicitly, see

https://cs.uwaterloo.ca/~shallit/papers.html.

Most of the other results of [13] can be proved using the analogous techniques.

In principle, everything we have said in this paper can be applied to β-expansions, where β is a quadratic Pisot number. The choice $\beta = \varphi$ was particularly easy in the current version of `Walnut`, because it already has Fibonacci and negaFibonacci numbers implemented, and adders for integers represented in these forms.

Acknowledgments

I thank Anna Lubiw, George Bergman, Jacques Sakarovitch, Michel Dekking, and Daniel Berend for helpful comments and insights.

I also thank the two anonymous referees, who read the paper carefully and provided many useful suggestions.

Finally, I thank both George Bergman (happy 80th birthday, George!) and Christiane Frougny (Joyeux anniversaire, Christiane !) for their inspiring work over many decades.

References

[1] D. Berend and C. Frougny. Computability by finite automata and Pisot bases. *Math. Systems Theory*, 27:275–282, 1994.

[2] G. Bergman. A number system with an irrational base. *Math. Mag.*, 31:98–110, 1957.

[3] J. Berstel. An exercise on Fibonacci representations. *RAIRO Inform. Théor. App.*, 35:491–498, 2001.

[4] J. Berstel and C. Reutenauer. *Noncommutative Rational Series With Applications*, volume 137 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2011.

[5] A. Bertrand-Mathis. Comment écrire les nombres entiers dans une base qui n'est pas entière. *Acta Math. Hung.*, 54:237–241, 1989.

[6] J. L. Brown, Jr. Unique representation of integers as sums of distinct Lucas numbers. *Fibonacci Quart.*, 7:243–252, 1969.

[7] M. W. Bunder. Zeckendorf representations using negative Fibonacci numbers. *Fibonacci Quart.*, 30:111–115, 1992.

[8] H. V. Chu, D. C. Luo, and S. J. Miller. On Zeckendorf related partitions using the Lucas sequence. *Fibonacci Quart.*, 60:111–119, 2022.

[9] F. M. Dekking. The structure of base phi expansions. Arxiv preprint arXiv:2305.08349 [math.NT], May 15 2023. Available at https://arxiv.org/abs/2305.08349.

[10] F. M. Dekking. Base phi representations and golden mean beta-expansions. *Fibonacci Quart.*, 58:38–48, 2020.

[11] F. M. Dekking. The sum of digits function of the base phi expansion of the natural numbers. *INTEGERS*, 20:#A45, 2020.

[12] F. M. Dekking. The sum of digits functions of the Zeckendorf and the base phi expansions. *Theoret. Comput. Sci.*, 859:70–79, 2021.

[13] M. Dekking and A. Van Loon. Counting base phi representations. Arxiv preprint arXiv:2304.11387 [math.NT], April 22 2023. Available at https://arxiv.org/abs/2304.11387.

[14] M. Dekking and A. Van Loon. On the representation of the natural numbers by powers of the golden mean. Arxiv preprint arXiv:2111.07544 [math.NT], November 15 2021. Available at https://arxiv.org/abs/2111.07544.

[15] L. Fleischer and J. O. Shallit. Automata, palindromes, and reversed subwords. *J. Automata, Languages, and Combinatorics*, 26:221–253, 2021.

[16] C. Frougny. Linear numeration systems, θ-developments and finite automata. In B. Monien and R. Cori, editors, *STACS 89*, volume 349 of *Lecture Notes in Computer Science*, pages 144–155. Springer-Verlag, 1989.

[17] C. Frougny. Representation of numbers in non-classical numeration systems. In P. Kornerup and D. W. Matula, editors, *Proc. 10th IEEE Symp. Computer Arithmetic*, pages 17–21. IEEE Press, 1991.

[18] C. Frougny. Confluent linear numeration systems. *Theoret. Comput. Sci.*, 106:183–219, 1992.

[19] C. Frougny. How to write integers in non-integer base. In I. Simon, editor, *1st Latin American Symposium on Theoretical Informatics (LATIN '92)*, Lecture Notes in Computer Science, pages 154–164. Springer-Verlag, 1992.

[20] C. Frougny. Representations of numbers and finite automata. *Math. Systems Theory*, 25:37–60, 1992.

[21] C. Frougny. Systèmes de numération linéaires et θ-représentations. *Theoret. Comput. Sci.*, 94:223–236, 1992.

[22] C. Frougny. On-line digit set conversion in real base. *Theoret. Comput. Sci.*, 292:221–235, 2003.

[23] C. Frougny and J. Sakarovitch. Automatic conversion from Fibonacci representation to representation in base φ, and a generalization. *Internat. J. Algebra Comput.*, 9:351–384, 1999.

[24] E. Hart and L. Sanchis. On the occurrence of F_n in the Zeckendorf decomposition of nF_n. *Fibonacci Quart.*, 37:21–33, 1999.

[25] S. Labbé and J. Lepšová. A Fibonacci analogue of the two's complement numeration system. *RAIRO Inform. Théor. App.*, 57:Paper 12, 2023.

[26] C. G. Lekkerkerker. Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci. *Simon Stevin*, 29:190–195, 1952.

[27] M. Lothaire. *Algebraic Combinatorics on Words*, volume 90 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2002.

[28] H. Mousavi. Automatic theorem proving in `Walnut`. Arxiv preprint, arXiv:1603.06017 [cs.FL], May 25 2021. Available at http://arxiv.org/abs/1603.06017.

[29] W. Parry. On the β-expansions of real numbers. *Acta Math. Acad. Sci. Hung.*, 11:401–416, 1960.

[30] A. Rényi. Representations for real numbers and their ergodic properties. *Acta Math. Acad. Sci. Hung.*, 8:477–493, 1957.

[31] C. Rousseau. The phi number system revisited. *Math. Mag.*, 68:283–284, 1995.

[32] G. R. Sanchis and L. A. Sanchis. On the frequency of occurrence of α^i in the α-expansions of the positive integers. *Fibonacci Quart.*, 39:123–137, 2001.

[33] J. Shallit. Robbins and Ardila meet Berstel. *Inform. Process. Lett.*, 167:106081, 2021.

[34] J. Shallit. *The Logical Approach To Automatic Sequences: Exploring Combinatorics on Words with* `Walnut`, volume 482 of *London Math. Soc. Lecture Note Series*. Cambridge University Press, 2023.

[35] J. Shallit, S. L. Shan, and K.-H. Yang. Automatic sequences in negative bases and proofs of some conjectures of Shevelev. *RAIRO Inform. Théor. App.*, 57:4, 2023.

[36] N. J. A. Sloane et al. The on-line encyclopedia of integer sequences. Electronic resource, available at https://oeis.org, 2023.

[37] I. N. Stewart and D. O. Tall. *Algebraic Number Theory.* Chapman and Hall, 1979.

[38] E. Zeckendorf. Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. *Bull. Soc. Roy. Liège*, 41:179–182, 1972.

Received: December 1, 2023
Accepted for publication: April 25, 2024
Communicated by: Rigo Michel, Emilie Charlier and Julien Leroy