Decision Algorithms for Fibonacci-Automatic Words, II: Related Sequences and Avoidability

Chen Fei Du[1], Hamoon Mousavi[1], Eric Rowland[2],
Luke Schaeffer[3], and Jeffrey Shallit[1]

October 10, 2016

Abstract

We use a decision procedure for the "Fibonacci-automatic" words to solve problems about a number of different sequences. In particular, we prove that there exists an aperiodic infinite binary word avoiding the pattern xxx^R. This is the first avoidability result concerning a nonuniform morphism proven purely mechanically.

Key words and phrases: automatic sequence; decision procedure; avoidability in words; finite automata; Fibonacci representation; palindrome.

1 Introduction

This is the second of three papers on the application of logical methods to solve problems in combinatorics on words. In a previous paper [22], we introduced a decision procedure for the "Fibonacci-automatic" words. These are infinite words $(a_n)_{n \geq 0}$ that are generated by finite automata in the following sense: the input to the automaton is the Fibonacci (or "Zeckendorf") representation w of the number n (see [18, 30]). The output associated with the last state reached upon reading w is then a_n. (If a state has no output associated with it, then by default the output is 1 for an accepting state and 0 for a nonaccepting state.)

Recall that the Fibonacci numbers are defined by $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$. Later, we will also need the associated Lucas numbers, defined by $L_0 = 2$, $L_1 = 1$, and $L_n = L_{n-1} + L_{n-2}$ for $n \geq 2$.

---

[1]School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada; `cfdu@uwaterloo.ca`, `sh2mousa@uwaterloo.ca`, `shallit@uwaterloo.ca` .

[2]Department of Mathematics, Hofstra University, Hempstead, NY 11549-1000, USA; `eric.rowland@hofstra.edu` .

[3]Computer Science and Artificial Intelligence Laboratory, The Stata Center, MIT Building 32, 32 Vassar Street, Cambridge, MA 02139 USA; `lrschaeffer@gmail.com` .

A classical theorem states that every non-negative integer can be represented, in an essentially unique way, as a sum of Fibonacci numbers $(F_i)_{i \geq 2}$, subject to the constraint that no two consecutive Fibonacci numbers are used. See [23, 18, 30, 7, 13]. Such a Fibonacci representation can be written as a binary string $a_1 a_2 \cdots a_n$ representing the integer $\sum_{1 \leq i \leq n} a_i F_{n+2-i}$. For $w = a_1 a_2 \cdots a_n \in \Sigma_2^*$, we define $[a_1 a_2 \cdots a_n]_F := \sum_{1 \leq i \leq n} a_i F_{n+2-i}$, even if $a_1 a_2 \cdots a_n$ has leading zeroes or consecutive 1's. By $(n)_F$ we mean the *canonical* Fibonacci representation for the integer n, having no leading zeroes or consecutive 1's. For example, $(43)_F = 10010001$ and $[0100101]_F = 17$.

The canonical example of a Fibonacci-automatic word is the infinite Fibonacci word

$$\mathbf{f} = (f_i)_{i \geq 0} = 01001010 \cdots$$

where f_i is the last (least significant) digit in the Fibonacci representation of i. In a previous paper [22], we showed that many known and some new properties of $\mathbf{f}$ can be proved by a logical decision procedure involving automata.

In this paper we apply our method to some related Fibonacci-automatic sequences. In particular, we study the aperiodic Rote-Fibonacci word, and show that it avoids the pattern xxx^R. To the best of our knowledge, this is the first nontrivial avoidability property proved purely mechanically, using a decision procedure based on Fibonacci representation. With a more involved argument, we also give an infinite class of periodic words avoiding xxx^R. We also study the Fibonacci-Thue-Morse sequence. Finally, we show how to obtain a result about the avoidance of additive squares, using our technique.

The main technique we use is to write assertions about sequences in the first-order theory of the natural numbers with addition, also allowing indexing into the Fibonacci-automatic sequences under consideration. Given a logical predicate making an assertion about a Fibonacci-automatic sequence, the decision procedure of [22] translates this predicate into an automaton accepting the Fibonacci representation of the free variables making the predicate true. A few details about the implementation, called `Walnut`, are in Section 7; it is freely available for download. We provide `Walnut` commands for many of the examples in this paper, allowing the reader to verify them independently.

2   Basics

We recall some basic definitions of combinatorics on words. By $|x|$ we mean the length of the word x. By ϵ we mean the empty word. A *square* is a nonempty word of the form $x^2 = xx$, like the English word `murmur`. The *order* of a square xx is the length $|x|$. Similarly, a *cube* is a nonempty word of the form $x^3 = xxx$, and so forth. An *overlap* is a word of the form $axaxa$, where a is a single letter and x is a (possibly empty) word, like the English word `alfalfa`. Its *order* is defined to be $|ax|$. A *superoverlap* is a word of the form $abxabxab$, where a, b are single letters and x is a (possibly empty) word, like the English word `tingalingaling` with the first letter removed.

By x^R we mean the reversal of the word x; for example, $(\mathtt{drawer})^R = \mathtt{reward}$. A word x is a *palindrome* if $x = x^R$.

We define the morphism $x \to \overline{x}$ over the alphabet $\{0, 1\}$ by $\overline{0} = 1$ and $\overline{1} = 0$. This is, of course, extended to words (both finite and infinite) by applying the map to each letter. A nonempty binary word w is an *antisquare* if it can be written in the form $w = x\overline{x}$, and it is an *antipalindrome* if it satisfies $x = \overline{x}^R$. Of course, any antipalindrome must be of even length; its *order* is defined to be half its length.

We say that v is a *factor* of a word x if we can write $x = uvw$ for (possibly empty) words u, w. In the same situation, u is a *prefix* of x and w is a *suffix*. These definitions are generalized to infinite words in the obvious way.

For an infinite word $\mathbf{a} = a_0 a_1 a_2 \cdots$ we define $\mathbf{a}[i] = a_i$ and $\mathbf{a}[i..j] = a_i a_{i+1} \cdots a_j$. By x^ω we mean the infinite word $xxx \cdots$.

We say an infinite word $\mathbf{w}$ *avoids squares* if $\mathbf{w}$ contains no factor that is a square. Avoidance for other kinds of patterns is defined analogously. Avoidance in words is a subject of intensive study in combinatorics on words (see, e.g., [25]).

In the next section, we will also need the finite Fibonacci words (X_n), defined as follows:

$$
X_n = \begin{cases}
\epsilon, & \text{if } n = 0; \\
1, & \text{if } n = 1; \\
0, & \text{if } n = 2; \\
X_{n-1} X_{n-2}, & \text{if } n > 2.
\end{cases}
$$

Note that $|X_n| = F_n$ for $n \geq 1$. The word X_n (for $n \neq 1$) is the prefix of length F_n of $\mathbf{f}$.

3 Avoiding the pattern xxx^R and the Rote-Fibonacci word

In this section we show how to apply our decision method to an interesting and novel avoidance property: avoiding the pattern xxx^R. An example matching this pattern in English is a factor of the word `bepepper`, with $x = $ `ep`. In this paper, however, we are concerned only with the binary alphabet $\Sigma_2 = \{0, 1\}$.

Although avoiding patterns with reversal has been considered before in several papers (e.g., [26, 2, 9, 1, 12, 19]), it seems our particular problem is new.

If the goal is just to produce *some* infinite word avoiding xxx^R, then a solution seems easy: namely, the infinite word $(01)^\omega$ clearly avoids xxx^R, since (as the referee observed) an occurrence of xx^R implies two consecutive identical letters. This suggests the problem of determining which periodic infinite words avoid xxx^R. In Theorem 12 we give a partial answer to this question, but for the moment, we change our question to

Are there infinite aperiodic binary words avoiding xxx^R?

To answer this question, we will study a special infinite word, which we call the *Rote-Fibonacci word*. (The name comes from the fact that it is a special case of a class of words discussed in 1994 by Rote [28].) Consider the following transducer T:
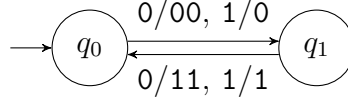
Figure 1: Transducer converting Fibonacci words to Rote-Fibonacci words

This transducer acts on words by following the transitions and outputting the concatenation of the outputs associated with each transition. Thus, for example, the input 01001 gets transduced to the output 00100110.

Before we state our first theorem, we introduce some notation.

- $|x|_l$ denotes the number of occurrences of the letter l in the string x;

- $C(x)$ denotes $\overline{x}$, the complement of x, so that $C(0) = 1$ and $C(1) = 0$;

- For a word $\mathbf{a}$ defined over $\mathbb{Z}$ by $-\mathbf{a}$ we mean the word obtained from x by changing the sign of every letter;

- For a word $\mathbf{a} = a_0 a_1 a_2 \cdots$ defined over $\mathbb{Z}$ by $\Delta(\mathbf{a})$ we mean $(a_1 - a_0)(a_2 - a_1) \cdots$, the first differences of $\mathbf{a}$;

- $s(x)$ denotes the sequence arising from a binary sequence x by changing every second 1 to -1;

- $\Sigma(x)$ denotes the running sum of the sequence x; that is, if $x = a_1 a_2 a_3 \cdots$ then $\Sigma(x)$ is $a_1 (a_1 + a_2)(a_1 + a_2 + a_3) \cdots$;

- $h^\omega(a)$, for a morphism h and a letter a, is the infinite fixed point obtained by iterating h on the letter a.

Theorem 1. *The Rote-Fibonacci word*

$$\mathbf{r} = 0010011011011001001101101100100100110110010010011011001 00100 \cdots = r_0 r_1 r_2 \cdots$$

has the following equivalent descriptions:

(a) As the output of the transducer T, starting in state 0, on input $\mathbf{f}$.

(b) As $\tau(h^\omega(a))$, where h and τ are defined by

$$
\begin{aligned}
h(a) &= ab_1 \\
h(b) &= a \\
h(a_0) &= a_2 b \\
h(a_1) &= a_0 b_0 \\
h(a_2) &= a_1 b_2 \\
h(b_0) &= a_0 \\
h(b_1) &= a_1 \\
h(b_2) &= a_2
\end{aligned}
\qquad\qquad
\begin{aligned}
\tau(a) &= 0 \\
\tau(b) &= 1 \\
\tau(a_0) &= 0 \\
\tau(a_1) &= 1 \\
\tau(a_2) &= 1 \\
\tau(b_0) &= 0 \\
\tau(b_1) &= 0 \\
\tau(b_2) &= 1
\end{aligned}
$$

(c) As the binary sequence generated by the following deterministic finite automaton with output (DFAO), with outputs given in the states, and input giving the Fibonacci representation of n.
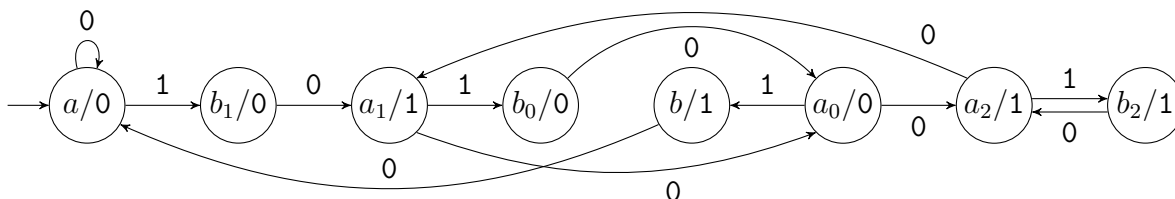

Figure 2: Canonical Fibonacci representation DFAO generating the Rote-Fibonacci word

(d) As the limit, as $n \to \infty$, of the sequence of finite Rote-Fibonacci words $(R_n)_n$ defined as follows: $R_0 = 0$, $R_1 = 00$, and for $n \geq 3$

$$
R_n = \begin{cases} R_{n-1} R_{n-2}, & \text{if } n \equiv 0 \pmod 3; \\ R_{n-1} \overline{R_{n-2}}, & \text{if } n \equiv 1, 2 \pmod 3. \end{cases}
$$

(e) As the sequence obtained from the Fibonacci sequence $\mathbf{f} = f_0 f_1 f_2 \cdots = 0100101001001 \cdots$ as follows: first, change every 0 to 1 and every 1 to 0 in $\mathbf{f}$, obtaining $\overline{\mathbf{f}} = 1011010110110 \cdots$. Next, in $\overline{\mathbf{f}}$ change every second 1 that appears to -1 (which we write as $\overline{1}$ for clarity): $s(\overline{f}) = 10\overline{1}10\overline{1}01\overline{1}01\overline{1}0 \cdots$. Now take the running sum Σ of this sequence, obtaining $\Sigma(s(\overline{f})) = 1101100100100 \cdots$, and finally, complement it to get $\mathbf{r}$.

(f) As $\rho(g^\omega(a))$, where g and ρ are defined as follows

$$
\begin{aligned}
g(a) &= abcab \\
g(b) &= cda \\
g(c) &= cdacd \\
g(d) &= abc
\end{aligned}
\qquad\qquad
\begin{aligned}
\rho(a) &= 0 \\
\rho(b) &= 0 \\
\rho(c) &= 1 \\
\rho(d) &= 1
\end{aligned}
$$

Proof. $(a) \iff (d)$: Let $T_0(x)$ (resp., $T_1(x)$) denote the output of the transducer T starting in state q_0 (resp., q_1) on input x. Then a simple induction on n shows that $T_0(X_{n+1}) = R_n$ and $T_1(X_{n+1}) = \overline{R_n}$. We give only the induction step for the first claim:

$$T_0(X_{n+1}) = T_0(X_n X_{n-1})$$

$$= \begin{cases} T_0(X_n)T_0(X_{n-1}), & \text{if } |X_n| \text{ is even;} \\ T_0(X_n)T_1(X_{n-1}), & \text{if } |X_n| \text{ is odd;} \end{cases}$$

$$= \begin{cases} R_{n-1}R_{n-2}, & \text{if } n \equiv 0 \pmod 3; \\ R_{n-1}\overline{R_{n-2}}, & \text{if } n \not\equiv 0 \pmod 3; \end{cases}$$

$$= R_n.$$

Here we have used the easily-verified fact that $|X_n| = F_n$ is even iff $n \equiv 0 \pmod 3$.

$(b) \iff (d)$: We verify by a tedious induction on n that for $n \geq 0$ we have

$$\tau(h^n(a)) = \tau(h^{n+1}(b)) = R_n$$

$$\tau(h^n(a_i)) = \tau(h^{n+1}(b_i)) = \begin{cases} R_i, & \text{if } n \equiv i \pmod 3; \\ \overline{R_i}, & \text{if } n \not\equiv i \pmod 3. \end{cases}$$

$(c) \iff (b)$: Follows from the well-known transformation from automata to morphisms and vice versa (see, e.g., [17]).

$(d) \iff (e)$: We have

$$\Sigma(s(xy)) = \begin{cases} \Sigma(s(x)) \, \Sigma(s(y)), & \text{if } |x|_1 \text{ even;} \\ \Sigma(s(x)) \, C(\Sigma(s(y))), & \text{if } |x|_1 \text{ odd.} \end{cases}$$

Then we claim that $C(R_n) = \Sigma(s(C(X_{n+2})))$. This can be verified by induction on n. We give only the induction step:

$$\Sigma(s(C(X_{n+2}))) = \Sigma(s(C(X_{n+1})C(X_n)))$$

$$= \begin{cases} \Sigma(s(C(X_{n+1}))) \, \Sigma(s(C(X_n))), & \text{if } |C(X_{n+1})|_1 \text{ even;} \\ \Sigma(s(C(X_{n+1}))) \, C(\Sigma(s(C(X_n)))), & \text{if } |C(X_{n+1})|_1 \text{ odd;} \end{cases}$$

$$= \begin{cases} C(R_{n-1}) \, C(R_{n-2}), & \text{if } n \equiv i \pmod 3; \\ C(R_{n-1}) \, R_{n-2}, & \text{if } n \not\equiv i \pmod 3; \end{cases}$$

$$= C(R_n).$$

$(b) \iff (f)$: Define γ by

$$\gamma(a) = \gamma(a_0) = a$$
$$\gamma(b_0) = \gamma(b_1) = b$$
$$\gamma(a_1) = \gamma(a_2) = c$$
$$\gamma(b) = \gamma(b_2) = d.$$

6

We verify by a tedious induction on n that for $n \geq 0$ we have

$$
\begin{aligned}
g^n(a) &= \gamma(h^{3n}(a)) = \gamma(h^{3n}(a_0)) \\
g^n(b) &= \gamma(h^{3n}(b_0)) = \gamma(h^{3n}(b_1)) \\
g^n(c) &= \gamma(h^{3n}(a_1)) = \gamma(h^{3n}(a_2)) \\
g^n(d) &= \gamma(h^{3n}(b)) = \gamma(h^{3n}(b_2)).
\end{aligned}
$$

$\square$

Corollary 2. *The first differences $\Delta \mathbf{r}$ of the Rote-Fibonacci word $\mathbf{r}$, taken modulo 2, give the complement of the Fibonacci word $\overline{\mathbf{f}}$, with its first symbol omitted.*

Proof. Note that if $\mathbf{x} = a_0 a_1 a_2 \cdots$ is a binary sequence, then $\Delta(C(\mathbf{x})) = -\Delta(\mathbf{x})$. Furthermore $\Delta(\Sigma(\mathbf{x})) = a_1 a_2 \cdots$. Now from the description in part (e), above, we know that $\mathbf{r} = C(\Sigma(s(C(\mathbf{f}))))$. Hence $\Delta(\mathbf{r}) = \Delta(C(\Sigma(s(C(\mathbf{f}))))) = -\Delta(\Sigma(s(C(\mathbf{f})))) = \mathrm{dr}(-s(C(\mathbf{f})))$, where dr drops the first symbol of its argument. Taking the last result modulo 2 gives the result. $\square$

As it turns out, the Rote-Fibonacci word has (essentially) appeared before in several places. For example, in a 2009 preprint of Monnerot-Dumaine [20], the author studies a plane fractal called the "Fibonacci word fractal", specified by certain drawing instructions, which can be coded over the alphabet S, R, L by taking the fixed point $g^\omega(a)$ and applying the coding $\eta(a) = S$, $\eta(b) = R$, $\eta(c) = S$, and $\eta(d) = L$. Here S means "move straight one unit", "R" means "right turn one unit" and "L" means "left turn one unit".

More recently, Blondin Massé, Brlek, Labbé, and Mendès France studied a remarkable sequence of finite words closely related to $\mathbf{r}$ [3, 4, 5]. For example, in their paper "Fibonacci snowflakes" [3] they defined a certain sequence $(q_n)_{n \geq 0}$, as follows:

$$
q_n = \begin{cases}
\epsilon, & \text{if } n = 0; \\
\mathtt{R}, & \text{if } n = 1; \\
q_{n-1} q_{n-2}, & \text{if } n \equiv 2 \ (\mathrm{mod}\ 3); \\
q_{n-1} \overline{q_{n-2}}, & \text{if } n > 1 \text{ and } n \equiv 0, 1 \ (\mathrm{mod}\ 3).
\end{cases}
$$

Here $\overline{\mathtt{L}} = \mathtt{R}$ and $\overline{\mathtt{R}} = \mathtt{L}$. This sequence has the following relationship to g: let $\xi(a) = \xi(b) = \mathtt{L}$, $\xi(c) = \xi(d) = \mathtt{R}$. Then

$$
\mathtt{R}\, \xi(g^n(a)) = q_{3n+2}\, \mathtt{L}.
$$

3.1 Avoidability of xxx^R

We are now ready to prove our avoidability result. This is the first main result of the paper.

Theorem 3. *The Rote-Fibonacci word $\mathbf{r}$ is not ultimately periodic, but it avoids the pattern xxx^R.*

Proof. If **r** were ultimately periodic, there would be a period $p \geq 1$ and a position n after which the period begins. We can assert this using the predicate

$$\exists n \geq 0 \; \exists p \geq 1 \; \forall i \geq n \; \mathbf{r}[i] = \mathbf{r}[i + p],$$

or, as expressed in `Walnut`

```
eval rfperiod "?msd_fib En Ep ((p>=1) & Ai (i>=n) => R[i]=R[i+p])":
```

When we now use our decision procedure, we get the answer `false`, so **r** is aperiodic.

Next, we define two useful predicates:

$$\text{RFEQFACT}(i, j, n) := \forall t < n \; (\mathbf{r}[i + t] = \mathbf{r}[j + t])$$
$$\text{RFEQREVFACT}(i, j, n) := \forall t < n \; (\mathbf{r}[i + t] = \mathbf{r}[j - t - 1]),$$

which, in `Walnut`, are expressed as

```
def rfeqfact "?msd_fib At (t<n) => R[i+t]=R[j+t]":
def rfeqrevfact "?msd_fib At (t<n) => R[i+t]=R[j-t-1]":
```

The first predicate expresses the assertion that the length-n factors of **r** beginning at positions i and j are identical, and the second predicate expresses the assertion that the length-n factor beginning at position i is the reverse of the length-n factor ending at position $j - 1$.

Now we use `Walnut` to prove the desired result. A predicate expressing the assertion that there exists a factor of the form xxx^R of length n is as follows:

$$\exists i \; (\text{RFEQFACT}(i, i + n, n) \; \wedge \; \text{RFEQREVFACT}(i, i + 3n, n)),$$

or, in `Walnut`,

```
eval xxxr "?msd_fib Ei ($rfeqfact(i,i+n,n) & $rfeqrevfact(i,i+3*n,n))":
```

When we run this on our program, the only length n accepted is $n = 0$, so the Rote-Fibonacci word **r** contains no occurrences of the pattern xxx^R. □

4   Other properties of the Rote-Fibonacci word

We continue by proving some basic properties of **r**.

Theorem 4.

(a) *All squares in the Rote-Fibonacci word are of order F_{3n+1} for $n \geq 0$, and each such order occurs.*

(b) *All cubes in the Rote-Fibonacci word are of order F_{3n+1} for $n \geq 1$, and each such order occurs.*

(c) *There are palindromes of all lengths ≥ 0 in the Rote-Fibonacci word.*

(d) *There are antipalindromes of all orders ≥ 0 in the Rote-Fibonacci word.*

(e) *All antisquares in the Rote-Fibonacci word are of order F_{3n+2} and F_{3n+3} for $n \geq 0$, and all such orders occur.*

Proof. We use the following predicates:

$$\text{RFSQUARE}(n) := (n \geq 1) \;\wedge\; \exists i \;\; \text{RFEQFACT}(i, i+n, n)$$
$$\text{RFCUBE}(n) := (n \geq 1) \;\wedge\; \exists i \;\; \text{RFEQFACT}(i, i+n, 2n)$$
$$\text{RFPAL}(n) := \exists i \;\; \text{RFEQREVFACT}(i, i+n, n)$$
$$\text{RFANTIPAL}(n) = \exists i \; \forall t < n \;\; \mathbf{r}[i] \neq \mathbf{r}[i + 2n - t - 1]$$
$$\text{RFANTISQUARE}(n) = (n \geq 1) \;\wedge\; \exists i \; \forall t < n \;\; \mathbf{r}[i] \neq \mathbf{r}[i + n + t],$$

or, in `Walnut`,

```
eval rfsquare "?msd_fib (n>=1) & (Ei $rfeqfact(i,i+n,n))":
eval rfcube "?msd_fib (n>=1) & (Ei $rfeqfact(i,i+n,2*n))":
eval rfpal "?msd_fib Ei $rfeqrevfact(i,i+n,n)":
eval rfantipal "?msd_fib Ei At (t<n) => R[i+t] != R[i+2*n-t-1]":
eval rfantisquare "?msd_fib (n>=1) & Ei At (t<n) => R[i+t] != R[i+n+t]":
```

When we evaluate the predicate corresponding to (a), we get the automaton depicted in Figure 3. Clearly the accepted words correspond to F_{3n+1} for $n \geq 0$.
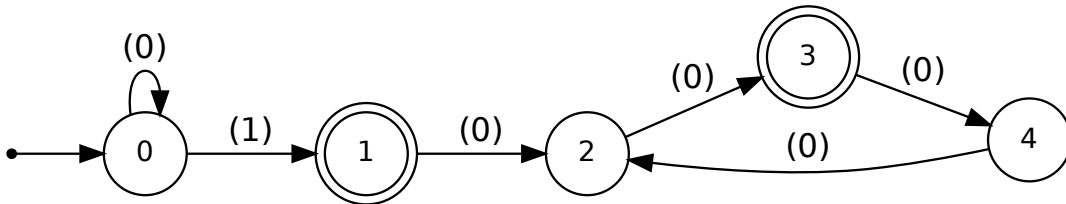

Figure 3: Automaton accepting orders of squares in the Rote-Fibonacci word

The predicate corresponding to (b) is similar. When we evaluate the predicate corresponding to (c), we get all lengths accepted. Similarly, evaluation of the predicate for (d) gives all orders accepted. The predicate corresponding to (e) produces the automaton depicted in Figure 4.
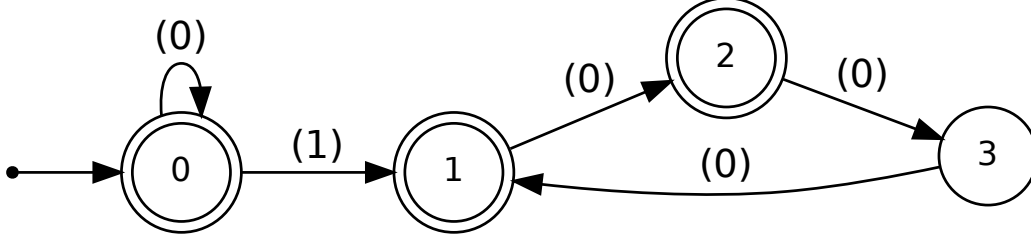
Figure 4: Automaton accepting orders of antisquares in the Rote-Fibonacci word

□

Theorem 5. *The minimum $q(n)$ over all periods of all length-n factors of the Rote-Fibonacci word is as follows:*

$$q(n) = \begin{cases} 1, & \text{if } 1 \leq n \leq 2; \\ 2, & \text{if } n = 3; \\ F_{3j+1}, & \text{if } j \geq 1 \text{ and } L_{3j} \leq n < L_{3j+2}; \\ L_{3j+1}, & \text{if } j \geq 1 \text{ and } L_{3j+2} \leq n < L_{3j+2} + F_{3j-2}; \\ F_{3j+2} + L_{3j}, & \text{if } j \geq 2 \text{ and } L_{3j+2} + F_{3j-2} \leq n < L_{3j+2} + F_{3j-1}; \\ 2F_{3j+2}, & \text{if } L_{3j+2} + F_{3j-1} \leq n < L_{3j+3}. \end{cases}$$

Proof. To prove this, we mimic the proof of Theorem 20 in [22], using the following predicates:

$$\text{RFPER}(i,j,p) := (i \leq j) \wedge (p \geq 1) \wedge (p + i - 1 \leq j) \wedge \text{RFEqFACT}(i, i+p, j+1-p-i)$$
$$\text{RFLEASTPER}(i,j,n) := \text{RFPER}(i,j,n) \wedge \forall n', \ 1 \leq n' < n, \ \neg\text{RFPER}(i,j,n')$$
$$\text{RFLP}(n) := \exists i \ \exists j \ \text{RFLEASTPER}(i,j,n)$$
$$\text{RFLPL}(n,p) := (n \geq 1) \wedge (\exists i \ \text{RFPER}(i, i+n-1, p)) \wedge$$
$$(\forall j \ \forall q \ \text{RFPER}(j, j+n-1, q) \implies (q \geq p)),$$

or, in `Walnut`,

```
def rfper "?msd_fib (i <= j) & (p >= 1) & (p+i <=j+1) & $rfeqfact(i,i+p,j+1-p-i)":
def rfleastper "?msd_fib $rfper(i,j,n) & (Anp ((1<=np)&(np<n)) => (~$rfper(i,j,np)))":
def rflp "?msd_fib Ei Ej $rfleastper(i,j,n)": def rflpl "?msd_fib (n>=1) & (Ei
$rfper(i,i+n-1,p)) & (Aj Aq $rfper(j,j+n-1,q) => (q >= p))":
```

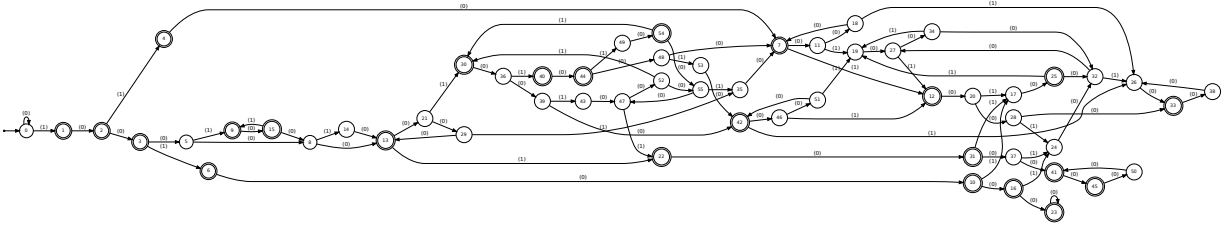The automaton for RFLP is displayed below in Figure 5.

Figure 5: Automaton accepting all least period lengths of factors of **r**
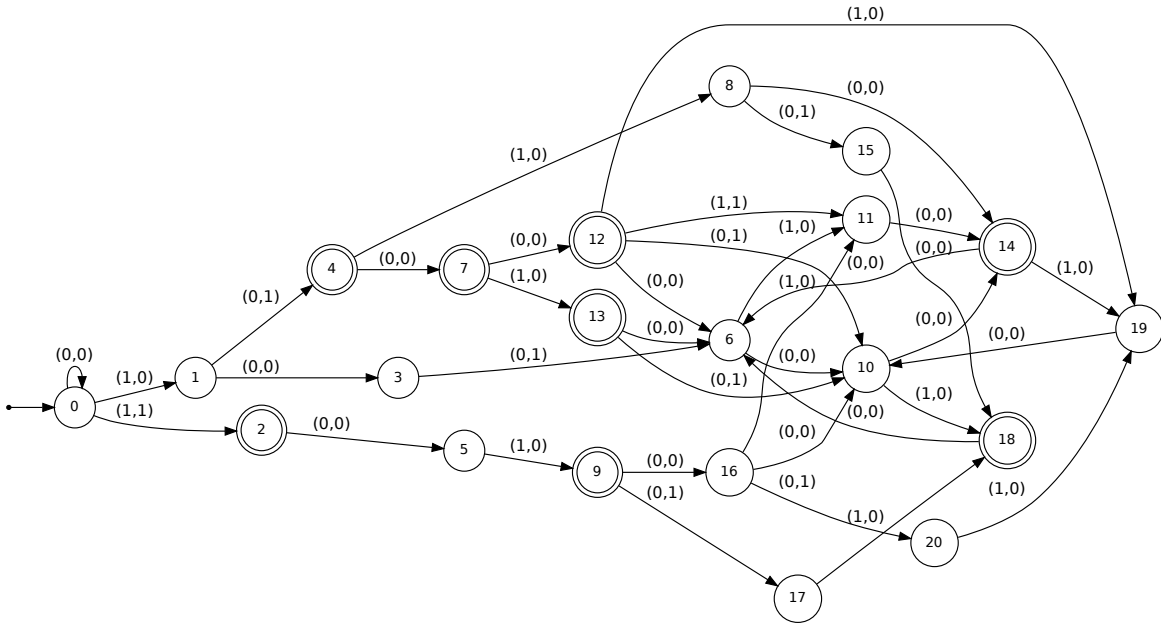
The automaton for RFLPL is displayed below in Figure 6.

Figure 6: Automaton accepting (n, p) where p is least period of length-n factors of **r**

$\square$

Let $\alpha = (1 + \sqrt{5})/2$.

Corollary 6. *The critical exponent of the Rote-Fibonacci word is $2 + \alpha$.*

Proof. An examination of the cases in Theorem 5 shows that the words of maximum exponent are those corresponding to $n = L_{3j+2} - 1$, $p = F_{3j+1}$. As $j \to \infty$, the quantity n/p approaches $2 + \alpha$ from below. $\square$

Recall that the *subword complexity* of a sequence **w** is the function counting number of distinct factors of **w** of length n.

Theorem 7. *The Rote-Fibonacci word has subword complexity $2n$.*

Proof. Follows from Corollary 2 together with [28, Thm. 3]. □

Theorem 8. *The Rote-Fibonacci word is mirror invariant. That is, if z is a factor of* **r** *then so is z^R.*

Proof. We use the predicate

$$\forall i \; \exists j \; \text{RFEQREVFACT}(i, j, n),$$

or, in `Walnut`,

`eval rfmirror "?msd_fib Ai Ej $rfeqrevfact(i,j,n)":`

The resulting automaton accepts the representation of all $n \geq 0$, so the conclusion follows.
□

Corollary 9. *The Rote-Fibonacci word avoids the pattern $xx^R x^R$.*

Proof. Suppose $xx^R x^R$ occurs in **r**. Then by Theorem 8 we know that $(xx^R x^R)^R = xxx^R$ occurs in **f**. But this is impossible, by Theorem 3. □

4.1 Conjectures and open problems about the Rote-Fibonacci word

In this section we collect some conjectures we have not yet been able to prove. We have made some progress and hope to completely resolve them in the future.

Conjecture 10. Every infinite binary word avoiding the pattern xxx^R has critical exponent $\geq 2 + \alpha$.

Conjecture 11. Let z be a finite nonempty primitive binary word. If z^ω avoids xxx^R, then $|z| = 2F_{3n+2}$ for some integer $n \geq 0$. Furthermore, z is a conjugate of the prefix $\mathbf{r}[0..2F_{3n+2} - 1]$, for some $n \geq 0$. Furthermore, for $n \geq 1$ we have that z is a conjugate of $y\overline{y}$, where $y = \tau(h^{3n}(a))$.

We can make some partial progress on Conjecture 11 in the following theorem. Here we give an infinite class of periodic infinite words avoiding the pattern xxx^R. This is the second main result of the paper.

Theorem 12. *Let $k \geq 1$ and define $n = 2F_{3k+2}$. Let $z = \mathbf{r}[0..n-1]$. Then the periodic word z^ω contains no occurrence of the pattern xxx^R.*

12

Proof. Suppose that z^ω does indeed contain an occurrence of xxx^R for some $|x| = \ell > 0$. We consider each possibility for ℓ and eliminate them in turn.

Case I: $\ell \geq n$.
There are two subcases:

Case I.1: $n \nmid \ell$: In this case, by considering the first n symbols of each of the two occurrences of x in xxx^R in z^ω, we see that there are two different cyclic shifts of z that are identical. This can only occur if $\mathbf{r}[0..n-1]$ is a power, and we know from Theorem 4 and Corollary 6 that this implies that $n = 2F_{3k+1}$ or $n = 3F_{3k+1}$ for some $k \geq 0$. But $2F_{3k+1} \neq 2F_{3k'+2}$ and $3F_{3k+1} \neq 2F_{3k'+2}$ provided $k, k' > 0$, so this case cannot occur.

Case I.2: $n \mid \ell$: Then x is a conjugate of z^e, where $e = \ell/n$. By a well-known result, a conjugate of a power is a power of a conjugate; hence there exists a conjugate y of z such that $x = y^e$. Then $x^R = y^e$, so x and hence y is a palindrome. We can now create a predicate that says that some conjugate of $\mathbf{r}[0..n-1]$ is a palindrome:

$$\exists i < n \; (\forall j < n \; \text{CMP}(i+j, n+i-1-j, n))$$

where

$$\begin{aligned}
\text{CMP}(k, k', n) := (((k < n) \;\wedge\; (k' < n)) &\implies (\mathbf{r}[k] = \mathbf{r}[k'])) \;\wedge\; \\
(((k < n) \;\wedge\; (k' \geq n)) &\implies (\mathbf{r}[k] = \mathbf{r}[k' - n])) \;\wedge\; \\
(((k \geq n) \;\wedge\; (k' < n)) &\implies (\mathbf{r}[k - n] = \mathbf{r}[k'])) \;\wedge\; \\
(((k \geq n) \;\wedge\; (k' \geq n)) &\implies (\mathbf{r}[k - n] = \mathbf{r}[k' - n])),
\end{aligned}$$

or, in `Walnut`,

```
def cmp "?msd_fib (((k<n)&(kp<n)) => (R[k]=R[kp])) & (((k<n)&(kp>=n)) =>
(R[k]=R[kp-n])) & (((k>=n)&(kp<n)) => (R[k-n]=R[kp])) & (((k>=n)&(kp>=n))
=> (R[k-n]=R[kp-n]))":
def rfconjpal "?msd_fib Ei (i<n) & (Aj (j<n) => $cmp(i+j,n+i-1-j,n))":
```

The predicate CMP has three arguments k, k', n: assuming $0 \leq k, k' < 2n$, it is true iff $\mathbf{r}[k \bmod n] = \mathbf{r}[k' \bmod n]$. The predicate RFCONJPAL is true iff some conjugate of $\mathbf{r}[0..n-1]$ is a palindrome. When we run this on `Walnut`, we discover that the only n with Fibonacci representation of the form 10010^i accepted are those with $i \equiv 0, 2 \pmod 3$, which means that $2F_{3k+2}$ is not among them. So this case cannot occur.
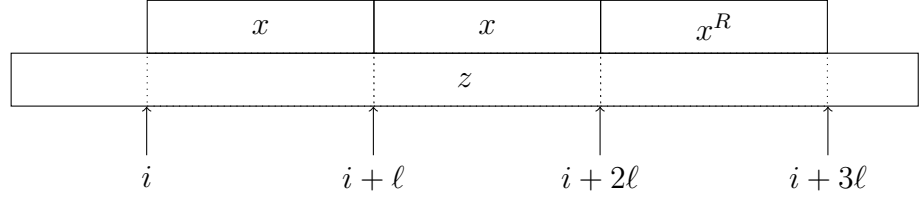
Case II: $\ell < n$.
There are now four subcases to consider, depending on the number of copies of z needed to "cover" our occurrence of xxx^R. In Case II.j, for $1 \leq j \leq 4$, we consider j copies of z and the possible positions of xxx^R inside that copy.
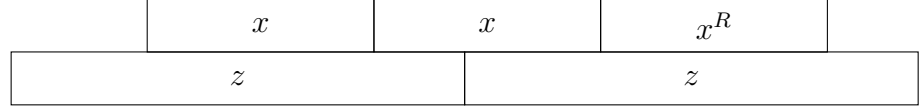
Because of the complicated nature of comparing one copy of x to itself in the case that one or both overlaps a boundary between different copies of z, it would be very helpful to be able
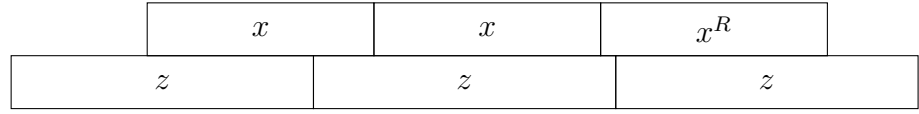
Case II.1:

$i + 3\ell < n$

Case II.2:

$n \le i + 3\ell < 2n$

Case II.3:

$2n \le i + 3\ell < 3n$
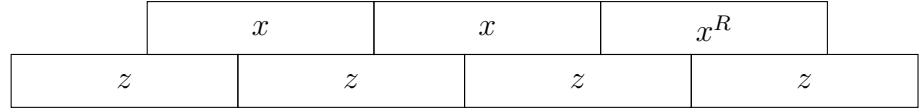

Case II.4:

$3n \le i + 3\ell < 4n$

Figure 7: Cases of the argument

to encode statements like $\mathbf{r}[k \bmod n] = \mathbf{r}[\ell \bmod n]$ in our logical language. Unfortunately, we cannot do this if n is arbitrary. So instead, we use a trick: assuming that the indices k, k' satisfy $0 \le k, k' < 2n$, we can use the $\mathrm{CMP}(k, k')$ predicate introduced above to simulate the assertion $\mathbf{r}[k \bmod n] = \mathbf{r}[k' \bmod n]$. Of course, for this to work we must ensure that $0 \le k, k' < 2n$ holds.

Cases II.1 through II.4 are illustrated in Figure 7. We assume that $|x| = \ell$ and xxx^R begins at position i of z^ω. We have the inequalities $i < n$ and $\ell < n$ which apply to each case. Our predicates are designed to compare the first copy of x to the second copy of x, and the first copy of x to the x^R.

Case II.1: If xxx^R lies entirely within one copy of z, it also lies in $\mathbf{r}$, which we have already seen cannot happen, in Theorem 3. This case therefore cannot occur.

Case II.2: We use the predicate

$$\exists i \, \exists \ell \, (i{+}3\ell \ge n) \wedge (i{+}3\ell < 2n) \wedge (\forall j < \ell \; \mathrm{CMP}(i{+}j, i{+}\ell{+}j)) \wedge (\forall k < \ell \; \mathrm{CMP}(i{+}k, i{+}3\ell{-}1{-}k))$$

to assert that there is a repetition of the form xxx^R.

Case II.3: We use the predicate

$$\exists i \, \exists \ell \, (i{+}3\ell \geq 2n) \wedge (i{+}3\ell < 3n) \wedge (\forall j < \ell \ \text{CMP}(i{+}j, i{+}\ell{+}j{-}n)) \wedge (\forall k < \ell \ \text{CMP}(i{+}k, i{+}3\ell{-}1{-}k{-}n))).$$

Case II.4: We use the predicate

$$\exists i \, \exists \ell \, (i{+}3\ell \geq 3n) \wedge (i{+}3\ell < 4n) \wedge (\forall j < \ell \ \text{CMP}(i{+}j, i{+}\ell{+}j{-}n)) \wedge (\forall k < \ell \ \text{CMP}(i{+}k, i{+}3\ell{-}1{-}k{-}2n)).$$

When we checked each of the cases II.2 through II.4 with our program, we discovered that $n = 2F_{3k+2}$ is never accepted. Actually, for these cases we had to employ one additional trick, because the computation for the predicates as stated required more space than was available on our machine. Here is the additional trick: instead of attempting to run the predicate for all n, we ran it only for n whose Fibonacci representation was of the form 10010^*. This significantly restricted the size of the automata we created and allowed the computation to terminate. In fact, we propagated this condition throughout the predicate.

We therefore eliminated all possibilities for the occurrence of xxx^R in z^ω and so it follows that no xxx^R occurs in z^ω. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Very recently, Currie and Rampersad [10] have considered the problem of enumerating the number of binary words of length n avoiding the pattern xxx^R. In a very surprising result, they proved that it is $n^{\Theta(\log n)}$. This is the first pattern-avoidance problem having this form of growth rate.

Open Problem 13. How many binary words of length n avoid both the pattern xxx^R and $(2 + \alpha)$-powers?

Call x *minimal* if xxx^R has no proper factor of the form www^R.

Conjecture 14. For $n = F_{3k+1}$ there are 4 minimal x of length n. For $n = F_{3k+1} \pm F_{3k-2}$ there are 2 minimal x. Otherwise there are none.

For $k \geq 3$ the 4 minimal words of length $n = F_{3k+1}$ are given by $\mathbf{r}[p_i..p_i + n - 1]$, $i = 1, 2, 3, 4$, where

$$(p_1)_F = 1000(010)^{k-3}001$$
$$(p_2)_F = 10(010)^{k-2}001$$
$$(p_3)_F = 1001000(010)^{k-3}001$$
$$(p_4)_F = 1010(010)^{k-2}001$$

For $k \geq 3$ the 2 minimal words of length $n = F_{3k+1} - F_{3k-2}$ are given by $\mathbf{r}[q_i..q_i + n - 1]$, $i = 1, 2$, where

$$(q_1)_F = 10(010)^{k-3}001$$
$$(q_2)_F = 10000(010)^{k-3}001$$

For $k \geq 3$ the 2 minimal words of length $n = F_{3k+1} + F_{3k-2}$ are given by $\mathbf{r}[s_i..s_i + n - 1]$, $i = 1, 2$, where

$$(s_1)_F = 10(010)^{k-3}001$$
$$(s_2)_F = 1000(01)^{k-2}001$$

5   The Fibonacci-Thue-Morse sequence

In this section we briefly apply our method to another Fibonacci-automatic sequence, obtaining several new results.

Consider a Fibonacci analogue of the Thue-Morse sequence

$$\mathbf{v} = (v_n)_{n \geq 0} = 0111010010001100010111000101 \cdots$$

where v_n is the sum of the bits, taken modulo 2, of the Fibonacci representation of n. We call this the *Fibonacci-Thue-Morse sequence*; it was introduced in [29, Example 2, pp. 12–13]. It is generated by the following automaton:
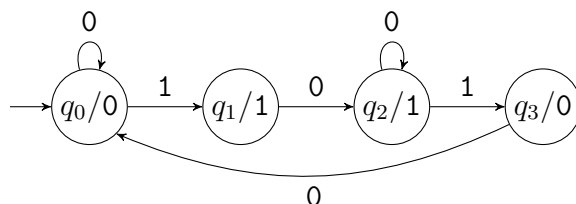

Figure 8: Automaton for the Fibonacci-Thue-Morse sequence $\mathbf{v}$

Theorem 15.

(a) *The only squares in $\mathbf{v}$ are of order 4 and F_n for $n \geq 2$, and a square of each such order occurs.*

(b) *The only cubes in $\mathbf{v}$ are the strings 000 and 111.*

(c) *The only overlaps in $\mathbf{v}$ are of order F_{2n} for $n \geq 1$, and an overlap of each such order occurs.*

(d) *There are no super-overlaps in $\mathbf{v}$.*

(e) *There are only 31 distinct palindromes in $\mathbf{v}$ (including the empty string). The longest is of length 12.*

16

Proof. We use the following predicates:

$$\text{FTMSQUARE}(n) := (n \geq 1) \ \wedge \ \exists i \ \forall t < n \ \mathbf{v}[i+t] = \mathbf{v}[i+n+t]$$
$$\text{FTMCUBE}(n) := (n \geq 1) \ \wedge \ \exists i \ \forall t < 2n \ \mathbf{v}[i+t] = \mathbf{v}[i+n+t]$$
$$\text{FTMOVERLAP}(n) = (n \geq 1) \ \wedge \ \exists i \ \forall t \leq n \ \mathbf{v}[i+t] = \mathbf{v}[i+n+t]$$
$$\text{FTMSUPEROVERLAP}(n) = (n \geq 2) \ \wedge \ \exists i \ \forall t \leq n+1 \ \mathbf{v}[i+t] = \mathbf{v}[i+n+t]$$
$$\text{FTMPAL}(n) := \exists i \ \forall t < n \ \mathbf{v}[i+t] = \mathbf{v}[i+n-t-1]$$

The equivalent statements in `Walnut` are

```
eval ftmsquare "?msd_fib (n>=1) & Ei At (t<n) => V[i+t] = V[i+n+t]":
eval ftmcube "?msd_fib (n>=1) & Ei At (t<2*n) => V[i+t] = V[i+n+t]":
eval ftmoverlap "?msd_fib (n>=1) & Ei At (t<=n) => V[i+t] = V[i+n+t]":
eval ftmsuperoverlap "?msd_fib (n>=2) & Ei At (t<=n+1) => V[i+t] = V[i+n+t]":
eval ftmpal "?msd_fib Ei At (t<n) => V[i+t] = V[i+n-t-1]":
```

When we run FTMSQUARE on `Walnut` we get the automaton depicted in Figure 9.
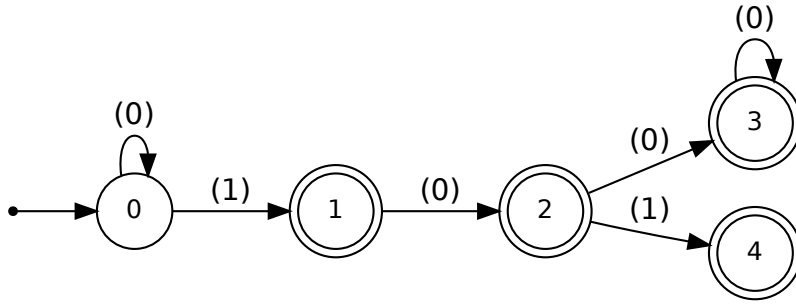

Figure 9: Orders of squares in the Fibonacci-Thue-Morse sequence

When we run FTMCUBE on `Walnut` we get an automaton that just accepts the order $n = 1$, corresponding to the length-3 factors 000 and 111. When we run FTMOVERLAP on `Walnut` we get an automaton that accepts the Fibonacci representations $1(00)^*$, which correspond to F_{2n} for $n \geq 1$. When we run FTMSUPEROVERLAP we get an automaton that accepts nothing. Finally, when we run FTMPAL we get the automaton depicted in Figure 10.
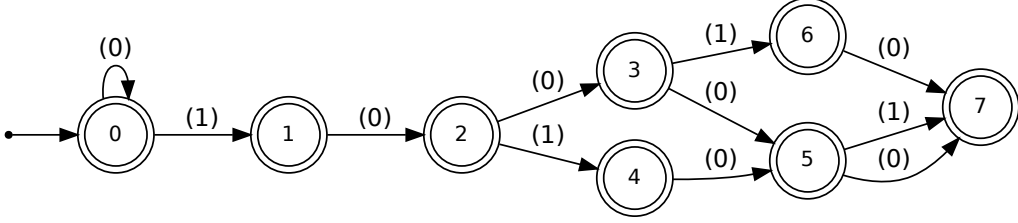
Figure 10: Lengths of palindromes in the Fibonacci-Thue-Morse sequence

□

We might also like to show that **v** is recurrent. The obvious predicate for this property holding for all words of length n is

$$\forall i \; \exists j > i \; (\forall t \; ((t < n) \implies (\mathbf{v}[i+t] = \mathbf{v}[j+t]))).$$

Unfortunately, when we attempt to run this with our prover, we get an intermediate NFA of 1159 states that we cannot determinize within the available space.

Instead, we rewrite the predicate, setting $k := j - i$ and $u := i + t$. This gives

$$\forall i \; \exists j > i \; \forall k \geq 1 \; \forall u, i \leq u < n + i \; (i = j + k) \implies \mathbf{v}[u] = \mathbf{v}[u + k],$$

or, in `Walnut`,

```
eval ftmrecur "?msd_fib Ai Ej ((j>i)&(Ak Av (((k>=1)&(i=j+k)&(v>=i)&(v<n+i)) =>
(V[v]=V[v+k])))))":
```

When we run this we discover that **v** is indeed recurrent.

Theorem 16. *The word* **v** *is recurrent.*

Another quantity of interest for the Thue-Morse-Fibonacci word **v** is its subword complexity $\rho_{\mathbf{v}}(n)$. It is not hard to see that it is linear. To obtain a deeper understanding of it, let us compute the first difference sequence $d(n) = \rho_{\mathbf{v}}(n+1) - \rho_{\mathbf{v}}(n)$. It is easy to see that $d(n)$ is the number of words w of length n with the property that both $w0$ and $w1$ appear in **v**. The natural way to count this is to count those i such that $t := \mathbf{v}[i..i+n-1]$ is the first appearance of that factor in **v**, and there exists a factor $\mathbf{v}[k..k+n]$ of length $n+1$ whose length-n-prefix equals t and whose last letter $\mathbf{v}[k+n]$ differs from $\mathbf{v}[i+n]$.

$$(\forall j < i \; \exists t < n \; \mathbf{v}[i+t] \neq \mathbf{v}[j+t]) \wedge (\exists k \; (\forall u < n \; \mathbf{v}[i+u] = \mathbf{v}[k+u]) \wedge \mathbf{v}[i+n] \neq \mathbf{v}[k+n]).$$

18

Unfortunately the same blowup appears as in the recurrence predicate, so once again we need to substitute, resulting in the predicate

$$(\forall j < i \; \exists k \geq 1 \; \exists v \; (i = j + k) \; \wedge \; (v \geq j) \; \wedge \; (v < n + j) \; \wedge \; \mathbf{v}[u] \neq \mathbf{v}[u + k]) \; \wedge$$
$$(\exists l > i \; \mathbf{v}[i + n] \neq \mathbf{v}[l + n]) \; \wedge$$
$$(\forall k' \; \forall u' \; (k' \geq 1) \; \wedge \; (l = i + k') \; \wedge \; (u' \geq i) \; \wedge \; (v' < n + i) \implies \mathbf{v}[k' + u'] = \mathbf{v}[u']),$$

or, in `Walnut`,

```
eval ftmsc "?msd_fib (Aj (j<i) => (Ek Ev (k>=1)&(i=j+k)&(v>=j)&(v<n+j)&(V[v]!=V[v+k])))
& (El ( (l>i) & (V[i+n] != V[l+n]) & (Akp Avp ((kp>=1)&(l=i+kp)&(vp>=i)&(vp<n+i))
=> (V[kp+vp]=V[vp])))))":
```

From the resulting 46-state automaton we can, as in [11], obtain a linear representation of rank 46, of the form $u\mu((n)_F)v$, where μ is a certain matrix-valued morphism and u, v are fixed vectors. We now consider all vectors of the form $u\{\mu(0), \mu(1)\}^*$. There are only finitely many distinct vectors of this form, and from them we can construct an automaton that generates the sequence $(d(n))_{n \geq 0}$. We omit the details and just state the result.

Theorem 17. *The first difference sequence $(d(n))_{n \geq 0}$ of the subword complexity of $\mathbf{v}$ is Fibonacci-automatic, and is generated by the following automaton.*
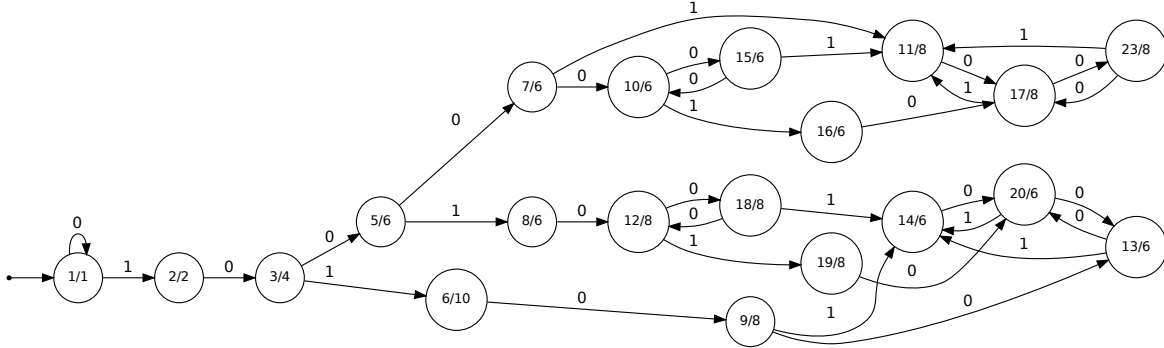

Figure 11: Automaton computing $d(n)$

Note that states are not numbered consecutively in Figure 17.

6   Combining two representations and avoidability

In this section we show how our decidability method can be used to handle an avoidability question where two different representations arise.

Let x be a finite word over the alphabet $\mathbb{N}^* = \{1, 2, 3 \ldots\}$. We say that x is an *additive square* if $x = x_1 x_2$ with $|x_1| = |x_2|$ and $\sum x_1 = \sum x_2$. For example, with the usual association of $\mathtt{a} = 1$, $\mathtt{b} = 2$, and so forth, up to $\mathtt{z} = 26$, we have that the English word $\mathtt{baseball}$ is an additive square, as $\mathtt{base}$ and $\mathtt{ball}$ both sum to 27.

An infinite word $\mathbf{x}$ over $\mathbb{N}^*$ is said to *avoid additive squares* if no factor is an additive square. It is currently unknown, and a relatively famous open problem, whether there exists an infinite word over a *finite* subset of $\mathbb{N}^*$ that avoids additive squares [6, 24, 16, 14], although it is known that additive cubes can be avoided over an alphabet of size 4 [8]. (Recently this last result was improved to alphabet size 3; see [27].)

However, it is easy to avoid additive squares over an *infinite* subset of $\mathbb{N}^*$; for example, any sequence that grows sufficiently quickly will have the desired property. Hence it is reasonable to ask about the *lexicographically least* sequence over $\mathbb{N}^*$ that avoids additive squares. Such a sequence begins

$$12131214212521312134121721 \cdots ,$$

but we do not currently know if this sequence is unbounded.

Here we consider the following variation on this problem. Instead of considering arbitrary sequences, we only consider sequences $\mathbf{a} = (a_i)_{i \geq 1}$ satisfying $\mathbf{a} = S(\mathbf{b})$ for some sequence $\mathbf{b} = (b_i)_{i \geq 0}$, where S is the sequence transformation defined by

$$\mathbf{a}[i] = \mathbf{b}[\nu_2(i)]$$

for all $i \geq 1$ and $\nu_2(i)$ is the exponent of the largest power of 2 dividing i. (Note that $\mathbf{a}$ and $\mathbf{b}$ are indexed differently.) For example, if $\mathbf{b} = 123 \cdots$, then $\mathbf{a} = 1213121412131215 \cdots$, the so-called "ruler sequence". It is known that $\mathbf{a}$ is squarefree and is, in fact, the lexicographically least sequence over $\mathbb{N}^*$ avoiding squares [15].

We then ask: what is the lexicographically least sequence avoiding additive squares that is of the form $S(\mathbf{b})$? The following theorem gives the answer. This is our third and final main result.

Theorem 18. *The lexicographically least sequence over $\mathbb{N}^*$ of the form $S(\mathbf{b})$ that avoids additive squares is*

$$\mathbf{a} = 1213121512131218 \cdots ,$$

where $\mathbf{b}[i] := F_{i+2}$.

Proof. First, we show that $\mathbf{a} := S(\mathbf{b}) = \prod_{k=1}^{\infty} \mathbf{b}[\nu_2(k)] = \prod_{k=1}^{\infty} F_{\nu_2(k)+2}$ avoids additive squares. For $m, n, j \in \mathbb{N}$, let $A(m, n, j)$ denote the number of occurrences of j in $\nu_2(m + 1), \ldots, \nu_2(m + n)$.

Consider two consecutive blocks of the same size, say $a_{i+1} \cdots a_{i+n}$ and $a_{i+n+1} \cdots a_{i+2n}$. Our goal is to compare the sums $\sum_{i < j \leq i+n} a_j$ and $\sum_{i+n < j \leq i+2n} a_j$.

First we prove

Lemma 19. *Let $m, j \geq 0$ and $n \geq 1$ be integers. Let $A(m, n, j)$ denote the number of occurrences of j in $\nu_2(m + 1), \ldots, \nu_2(m + n)$. Then for all $m, m' \geq 0$ we have*

$$|A(m', n, j) - A(m, n, j)| \leq 1.$$

Proof. We start by observing that the number of positive integers $\leq n$ that are divisible by t is exactly $\lfloor n/t \rfloor$. It follows that the number $B(n, j)$ of positive integers $\leq n$ that are divisible by 2^j but not by 2^{j+1} is

$$B(n, j) = \left\lfloor \frac{n}{2^j} \right\rfloor - \left\lfloor \frac{n}{2^{j+1}} \right\rfloor. \tag{1}$$

Now from the well-known identity

$$\lfloor x \rfloor + \left\lfloor x + \frac{1}{2} \right\rfloor = \lfloor 2x \rfloor,$$

valid for all real numbers x, substitute $x = n/2^{j+1}$ to get

$$\left\lfloor \frac{n}{2^{j+1}} \right\rfloor + \left\lfloor \frac{n}{2^{j+1}} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{n}{2^j} \right\rfloor,$$

which, combined with (1), shows that

$$B(n, j) = \left\lfloor \frac{n}{2^{j+1}} + \frac{1}{2} \right\rfloor.$$

Hence

$$\frac{n}{2^{j+1}} - \frac{1}{2} \leq B(n, j) < \frac{n}{2^{j+1}} + \frac{1}{2}. \tag{2}$$

Now the number of occurrences of j in $\nu_2(m+1), \ldots, \nu_2(m+n)$ is $A(m, n, j) = B(m + n, j) - B(m, j)$. From (2) we get

$$\frac{n}{2^{j+1}} - 1 < A(m, n, j) < \frac{n}{2^{j+1}} + 1 \tag{3}$$

for all $m \geq 0$. Since $A(m, n, j)$ is an integer, the inequality (3) implies that

$$|A(m', n, j) - A(m, n, j)| \leq 1$$

for all m, m'. $\qquad\square$

Note that for all $i, n \in \mathbb{N}$, we have $\sum_{k=i}^{i+n-1} \mathbf{a}[k] = \sum_{j=0}^{\lfloor \log_2(i+n) \rfloor} A(i, n, j) F_{j+2}$, so for adjacent blocks of length n we have

$$\sum_{k=i+n}^{i+2n-1} \mathbf{a}[k] - \sum_{k=i}^{i+n-1} \mathbf{a}[k] = \sum_{j=0}^{\lfloor \log_2(i+2n) \rfloor} (A(i+n, n, j) - A(i, n, j)) F_{j+2}.$$

Hence, $\mathbf{a}[i \mathinner{.\,.} i + 2n - 1]$ is an additive square iff

$$\sum_{j=0}^{\lfloor \log_2(i+2n) \rfloor} (A(i+n, n, j) - A(i, n, j)) F_{j+2} = 0,$$

and by above, each $A(i + n, n, j) - A(i, n, j) \in \{-1, 0, 1\}$.

The above suggests that we can take advantage of "unnormalized" Fibonacci representation in our computations. For $\Sigma \subseteq \mathbb{Z}$ and $w \in \Sigma^*$, we let the unnormalized Fibonacci representation $\langle w \rangle_{uF}$ be defined in the same way as $\langle w \rangle_F$, except over the alphabet Σ.

In order to use our decision procedure, we need two auxiliary DFAs: one that, given $i, n \in \mathbb{N}$ (represented in base 2), computes $\langle A(i + n, n, -) - A(i, n, -) \rangle_{uF}$, and another that, given $w \in \{-1, 0, 1\}^*$, decides whether $\langle w \rangle_{uF} = 0$. The first task can be done by a 6-state (incomplete) DFA M_{add22F} that accepts the language

$$\{z \in (\Sigma_2^2 \times \{-1, 0, 1\})^* \ : \ \forall j (\pi_3(z)[j] = A(\langle \pi_1(z) \rangle_2 + \langle \pi_2(z) \rangle_2, \langle \pi_2(z) \rangle_2, j) - A(\langle \pi_1(z) \rangle_2, \langle \pi_2(z) \rangle_2, j))\}.$$

Here π_i denotes projection onto the i'th coordinate. The second task can be done by a 5-state (incomplete) DFA $M_{\text{1uFisZero}}$ that accepts the language

$$\{w \in \{-1, 0, 1\}^* \ : \ \langle w \rangle_{uF} = 0\}.$$

We applied a modified decision procedure to the predicate $n \geq 1 \ \wedge \ \exists w (\text{add22F}(i, n, w) \ \wedge \ \text{1uFisZero}(w))$ and obtained as output a DFA that accepts nothing, so $\mathbf{a}$ avoids additive squares.

Next, we show that $\mathbf{a}$ is the lexicographically least sequence over $\mathbb{N}^*$ of the form $S(\mathbf{b})$ that avoids additive squares.

Note that for all $\mathbf{x}, \mathbf{y} \in \mathbb{N}^*$, $S(\mathbf{x}) < S(\mathbf{y})$ iff $\mathbf{x} < \mathbf{y}$ in the lexicographic ordering. Thus, we show that if any entry $\mathbf{b}[s]$ with $\mathbf{b}[s] > 1$ is changed to some $t \in [1, \mathbf{b}[s] - 1]$, then $\mathbf{a} = S(\mathbf{b})$ contains an additive square using only the first occurrence of the change at $\mathbf{a}[2^s - 1]$. More precisely, we show that for all $s, t \in \mathbb{N}$ with $t \in [1, F_{s+2} - 1]$, there exist $i, n \in \mathbb{N}$ with $n \geq 1$ and $i + 2n < 2^{s+1}$ such that either

$$2^s - 1 \in [i, i + n - 1] \text{ and } \sum_{k=i+n}^{i+2n-1} \mathbf{a}[k] - \sum_{k=i}^{i+n-1} \mathbf{a}[k] + t = 0$$

or

$$2^s - 1 \in [i + n, i + 2n - 1] \text{ and } \sum_{k=i+n}^{i+2n-1} \mathbf{a}[k] - \sum_{k=i}^{i+n-1} \mathbf{a}[k] - t = 0.$$

Setting up for a modified decision procedure, we use the following predicate, which says "r is a power of 2 and changing $\mathbf{a}[r - 1]$ to any smaller number results in an additive square in the first $2r$ positions", and six auxiliary DFAs. Note that all arithmetic and comparisons are in base 2.

$\text{powOf2}(r) \ \wedge \ \forall t \ ((t \geq 1 \ \wedge \ t < r \ \wedge \ \text{canonFib}(t)) \rightarrow \exists i \exists n (n \geq 1 \ \wedge \ i + 2n < 2r \ \wedge$

$\quad ((i < r \ \wedge \ r \leq i + n \ \wedge \ \forall w \ (\text{add22F}(i, n, w) \rightarrow \forall x \ (\text{bitAdd}(t, w, x) \rightarrow \text{2uFisZero}(x)))) \ \vee$

$\quad (i + n < r \ \wedge \ r \leq i + 2n \ \wedge \ \forall w \ (\text{add22F}(i, n, w) \rightarrow \forall x \ (\text{bitSub}(t, w, x) \rightarrow \text{2uFisZero}(x))))))).$

$$L(M_{\mathrm{powOf2}}) = \{w \in \Sigma_2^* \ : \ \exists n (w = (2^n)_2)\}.$$
$$L(M_{\mathrm{canonFib}}) = \{w \in \Sigma_2^* \ : \ \exists n (w = (n)_F)\}.$$
$$L(M_{\mathrm{bit(Add/Sub)}}) = \{z \in (\Sigma_2 \times \{-1, 0, 1\} \times \{-1, 0, 1, 2\})^* \ : \ \forall i \ (\pi_1(z)[i] \pm \pi_2(z)[i] = \pi_3(z)[i])\}.$$
$$L(M_{\mathrm{2uFisZero}}) = \{w \in \{-1, 0, 1, 2\}^* \ : \ \langle w \rangle_{uF} = 0\}.$$

We applied a modified decision procedure to the above predicate and auxiliary DFAs and obtained as output M_{powOf2}, so **a** is the lexicographically least sequence over $\mathbb{N}^*$ of the form $S(\mathbf{b})$ that avoids additive squares. $\qquad \square$

Remark 20. We explain how to construct automata handling unnormalized representations with `Walnut`. Suppose we are given just a normalized adder and the automaton M_{canonFib} that checks if a representation is canonical (no two consecutive 1's). Relabel transitions in the adder from (x, y, z) to $(x + y, z)$, obtaining and NFA, and determinize. Then intersect with $(0, 1^2)^*$. Note that we can split any string in $\{0, 1\}^*$ into the bitwise sum of two canonical Fibonacci representations (e.g., split into even position digits and odd position digits). Therefore the constructed automaton accepts all of $\{(n_{uF}, n_F) \ : \ n \in \mathbb{N}\}$, where the first component represents an unnormalized representation. On the other hand, the automaton accepts nothing more because, since Fibonacci representation is a positional system, the bitwise sum of x and y gives us the sum of x and y.

Now we have an automaton (call it M) that compares unnormalized representations with normalized representations for equality. We can use M and the original adder to construct an unnormalized adder:

$$\exists x' \ \exists y' \ \exists z' \ M(x, x') \ \wedge \ M(y, y') \ \wedge \ M(z, z') \ \wedge \ A(x', y', z').$$

Now we play the same game with the unnormalized adder. That is, replace (x, y, z) with $(x + y, z)$ to get an automaton that compares unnormalized representations over $\{0, 1, 2\}$ to an unnormalized representations over $\{0, 1\}$ (which can then be compared to normalized representations). Similarly, if we relabel (x, y, z) to $(x - z, y)$ then the first input is over $\{-1, 0, 1\}$, and the automaton compares it to a representation over $\{0, 1\}$. Repeat until the unnormalized representations cover the desired range of inputs.

7   Comments on our implementation

Our software `Walnut` is available for free download at

<div align="center">

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

</div>

To run the examples in this paper, the reader will have to define the automata for `R` and `V` in the `Word Automata Library` directory. The appropriate files can be downloaded from the URL above. For more information about `Walnut`, see [21].

We used the command

<div align="center">

`java -Xmx11000M -d64 Main.prover`

</div>

to run the examples in this paper.

For more details about the implementation, see [22].

8 Acknowledgments

References

[1] B. Bischoff, J. D. Currie, and D. Nowotka. Unary patterns with involution. *Internat. J. Found. Comp. Sci.* **23** (2012), 1641–1652.

[2] B. Bischoff and D. Nowotka. Pattern avoidability with involution. In *WORDS 2011*, pp. 65–70, 2011. Available at `http://rvg.web.cse.unsw.edu.au/eptcs/content.cgi?WORDS2011`.

[3] A. Blondin Massé, S. Brlek, A. Garon, and S. Labbé. Two infinite families of polyominoes that tile the plane by translation in two distinct ways. *Theoret. Comput. Sci.* **412** (2011), 4778–4786.

[4] A. Blondin Massé, S. Brlek, S. Labbé, and M. Mendès France. Fibonacci snowflakes. *Ann. Sci. Math. Québec* **35** (2011), 141–152.

[5] A. Blondin Massé, S. Brlek, S. Labbé, and M. Mendès France. Complexity of the Fibonacci snowflake. *Fractals* **20** (2012), 257–260.

[6] T. C. Brown and A. R. Freedman. Arithmetic progressions in lacunary sets. *Rocky Mountain J. Math.* **17** (1987), 587–596.

[7] L. Carlitz. Fibonacci representations. *Fibonacci Quart.* **6** (1968), 193–220.

[8] J. Cassaigne, J. Currie, L. Schaeffer, and J. Shallit. Avoiding three consecutive blocks of the same size and same sum. *J. Assoc. Comput. Mach.* **61**(2) (2014), Paper 10.

[9] J. D. Currie. Pattern avoidance with involution. Available at `https://arxiv.org/abs/1105.2849`, 2011.

[10] J. D. Currie and N. Rampersad. Growth rate of binary words avoiding xxx^R. *Theoret. Comput. Sci.* **609** (2016), 456–468.

[11] C. F. Du, H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, III: Enumeration and abelian properties. To appear, *Int. J. Found. Comput. Sci.* Preliminary version available at `https://cs.uwaterloo.ca/~shallit/Papers/part3.pdf`, 2015.

[12] G. Fici and L. Q. Zamboni. On the least number of palindromes contained in an infinite word. *Theoret. Comput. Sci.* **481** (2013), 1–8.

[13] A. S. Fraenkel. Systems of numeration. *Amer. Math. Monthly* **92** (1985), 105–114.

[14] A. R. Freedman and T. C. Brown. Sequences on sets of four numbers. *INTEGERS: Elect. J. of Combin. Number Theory* **16** (2016), #A33 (electronic).

[15] M. Guay-Paquet and J. Shallit. Avoiding squares and overlaps over the natural numbers. *Discrete Math.* **309** (2009), 6245–6254.

[16] L. Halbeisen and N. Hungerbühler. An application of Van der Waerden's theorem in additive number theory. *INTEGERS: Elect. J. of Combin. Number Theory* **0** (2000), #A7. http://www.integers-ejcnt.org/vol0.html.

[17] C. Holton and L. Q. Zamboni. Directed graphs and substitutions. *Theory Comput. Systems* **34** (2001), 545–564.

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

[19] R. Mercaş. A note on the avoidability of binary patterns with variables and reversals. Preprint, available at https://arxiv.org/pdf/1508.04571.pdf, 2015.

[20] A. Monnerot-Dumaine. The Fibonacci word fractal. Published electronically at http://hal.archives-ouvertes.fr/hal-00367972/fr/, 2009.

[21] H. Mousavi. Automatic theorem proving in Walnut. Preprint, available at https://arxiv.org/abs/1603.06017, 2016.

[22] H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, I: Basic results. *RAIRO Inform. Théor. App.* **50** (2016), 39–66.

[23] A. Ostrowski. Bemerkungen zur Theorie der Diophantischen Approximationen. *Abh. Math. Sem. Hamburg* **1** (1922), 77–98,250–251. Reprinted in *Collected Mathematical Papers*, Vol. 3, pp. 57–80.

[24] G. Pirillo and S. Varricchio. On uniformly repetitive semigroups. *Semigroup Forum* **49** (1994), 125–129.

[25] N. Rampersad. Infinite sequences and pattern avoidance. Master's thesis, University of Waterloo, 2004. Available at https://cs.uwaterloo.ca/~shallit/narad_masters.pdf.

[26] N. Rampersad and J. Shallit. Words avoiding reversed subwords. *J. Combin. Math. Combin. Comput.* **54** (2005), 157–164.

[27] M. Rao. On some generalizations of abelian power avoidability. *Theoret. Comput. Sci.* **601** (2015), 39–46.

[28] G. Rote. Sequences with subword complexity 2n. *J. Number Theory* **46** (1994), 196–213.

[29] J. O. Shallit. A generalization of automatic sequences. *Theoret. Comput. Sci.* **61** (1988), 1–16.

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