Decision Algorithms for Fibonacci-Automatic Words, III: Enumeration and Abelian Properties

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

April 10, 2015

Abstract

We continue our study the class of Fibonacci-automatic words. These are infinite sequences whose nth term is defined in terms of a finite-state function of the Fibonacci representation of n. In this paper, we show how enumeration questions (such as the number of squares) can be decided purely mechanically, using a decision procedure. We also examine abelian properties of these sequences.

1 Introduction

In two previous papers [15, 10], we studied the ramifications of a decision procedure for the Fibonacci-automatic sequences. These are sequences $(a_n)_{n\geq 0}$ generated by finite automata that take, as input the Fibonacci (or Zeckendorf) representation of n and output a_n. In this paper we show that we can also use this decision procedure to solve two other kinds of problems dealing with these sequences: enumeration of the number of factors obeying various properties, and questions involving abelian properties.

Our implementation of the decision procedure is called `Walnut`, and is available for free download at

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

Recall that every integer $n \geq 0$ can be uniquely represented in the form $\sum_{2\leq i\leq j} a_i F_i$, where $a_j = 1$ and $a_i a_{i+1} = 0$ for $2 \leq i < j$. We define $(n)_F$ to be the binary string $a_j a_{j-1} \cdots a_2$ (starting with the most significant digit). Similarly, for a word $w = b_1 b_2 \cdots b_j$ we define its interpretation in "base Fibonacci" $[w]_F = \sum_{1\leq i\leq j} a_i F_{j+2-i}$.

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

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

2 Enumeration

Mimicking the base-k ideas in [6], we can also mechanically enumerate many aspects of Fibonacci-automatic sequences. We do this by encoding the factors having the property in terms of paths of an automaton. This gives the concept of *Fibonacci-regular sequence* as previously studied in [1]. Roughly speaking, a sequence $(a(n))_{n\geq 0}$ taking values in $\mathbb{N}$ is Fibonacci-regular if the set of sequences

$$\{(a([xw]_F)_{w\in\Sigma_2^*} \ : \ x \in \Sigma_2^*\}$$

is finitely generated. Here we assume that $a([xw]_F)$ evaluates to 0 if xw contains the string 11. Every Fibonacci-regular sequence $(a(n))_{n\geq 0}$ has a *linear representation* of the form (u, μ, v) where u and v are row and column vectors, respectively, and $\mu : \Sigma_2 \to \mathbb{N}^{d\times d}$ is a matrix-valued morphism, where $\mu(0) = M_0$ and $\mu(1) = M_1$ are $d \times d$ matrices for some $d \geq 1$, such that

$$a(n) = u \cdot \mu(x) \cdot v$$

whenever $[x]_F = n$. The *rank* of the representation is the integer d. As an example, we exhibit a rank-6 linear representation for the sequence $a(n) = n + 1$:

$$u = [1\ 2\ 2\ 3\ 3\ 2]$$

$$M_0 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$v = [1\ 0\ 0\ 0\ 0\ 0]^T.$$

This can be proved by a simple induction on the claim that

$$u \cdot \mu(x) = [x_F + 1\ (1x)_F + 1\ (10x)_F - x_F\ (100x)_F - x_F\ (101x)_F - (1x)_F\ (1001x)_F - (101x)_F]$$

for strings x.

Recall that if $\mathbf{x}$ is an infinite word, then the subword complexity function $\rho_{\mathbf{x}}(n)$ counts the number of distinct factors of length n. Then, in analogy with [6, Thm. 27], we have

Theorem 1. *If $\mathbf{x}$ is Fibonacci-automatic, then the subword complexity function of $\mathbf{x}$ is Fibonacci-regular.*

Using our implementation, we can obtain a linear representation of the subword complexity function for $\mathbf{f}$. To do so, we use the predicate

$$\{(n, i)_F \ : \ \forall i' < i \ \mathbf{f}[i..i+n-1] \neq \mathbf{f}[i'..i'+n-1]\},$$

which expresses the assertion that the factor of length n beginning at position i has never appeared before. Then, for each n, the number of corresponding i gives $\rho_{\mathbf{f}}(n)$. When we do this for $\mathbf{f}$, we get the following linear representation (u', μ', v') of rank 10:

$$u' = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$M_0' = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$M_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$v' = [1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]^T$$

To show that this computes the function $n+1$, it suffices to compare the values of the linear representations (u, μ, v) and (u', μ', v') for all strings of length $\leq 10 + 6 = 16$ (using [2, Corollary 3.6]). After checking this, we have reproved the following classic theorem of Morse and Hedlund [14]:

Theorem 2. *The subword complexity function of* $\mathbf{f}$ *is* $n+1$.

We now turn to a result of Fraenkel and Simpson [11]. They computed the exact number of squares appearing in the finite Fibonacci words X_n; this was previously estimated by [8].

There are two variations: we could count the number of distinct squares in X_n, or what Fraenkel and Simpson called the number of "repeated squares" in X_n (i.e., the total number of *occurrences* of squares in X_n).

To solve this using our approach, we generalize the problem to consider any length-n prefix of X_n, and not simply the prefixes of length F_n.

We can easily write down predicates for these. The first represents the number of distinct squares in $\mathbf{f}[0..n-1]$:

$$L_{\text{ds}} := \{(n,i,j)_F \ : \ (j \geq 1) \text{ and } (i+2j \leq n) \text{ and } \mathbf{f}[i..i+j-1] = \mathbf{f}[i+j..i+2j-1] \\ \text{ and } \forall i' < i \ \mathbf{f}[i'..i'+2j-1] \neq \mathbf{f}[i..i+2j-1]\}.$$

This predicate asserts that $\mathbf{f}[i..i+2j-1]$ is a square occurring in $\mathbf{f}[0..n-1]$ and that furthermore it is the first occurrence of this particular string in $\mathbf{f}[0..n-1]$.

The second represents the total number of occurrences of squares in $\mathbf{f}[0..n-1]$:

$$L_{\text{dos}} := \{(n,i,j)_F \ : \ (j \geq 1) \text{ and } (i+2j \leq n) \text{ and } \mathbf{f}[i..i+j-1] = \mathbf{f}[i+j..i+2j-1]\}.$$

This predicate asserts that $\mathbf{f}[i..i+2j-1]$ is a square occurring in $\mathbf{f}[0..n-1]$.

We apply our method to the second example, leaving the first to the reader. Let $b(n)$ denote the number of occurrences of squares in $\mathbf{f}[0..n-1]$. First, we use our method to find a DFA M accepting L_{dos}. This (incomplete) DFA has 27 states.

Next, we compute matrices M_0 and M_1, indexed by states of M, such that $(M_a)_{k,l}$ counts the number of edges (corresponding to the variables i and j) from state k to state l on the digit a of n. We also compute a vector u corresponding to the initial state of M and a vector v corresponding to the final states of M. This gives us the following linear representation of the sequence $b(n)$: if $x = a_1 a_2 \cdots a_t$ is the Fibonacci representation of n, then

$$b(n) = u M_{a_1} \cdots M_{a_t} v, \tag{1}$$

which, incidentally, gives a fast algorithm for computing $b(n)$ for any n.

Now let $B(n)$ denote the number of square occurrences in the finite Fibonacci word X_n. This corresponds to considering the Fibonacci representation of the form 10^{n-2}; that is,

$B(n+1) = b([10^{n-1}]_F)$. The matrix M_0 is the following 27×27 array

$$\begin{bmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0
\end{bmatrix} \tag{2}$$

and has minimal polynomial

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

It now follows from the theory of linear recurrences that there are constants $c_1, c_2, \ldots, c_8$ such that

$$B(n+1) = (c_1 n + c_2)\alpha^n + (c_3 n + c_4)\beta^n + c_5 n + c_6 + (c_7 n + c_8)(-1)^n$$

for $n \geq 3$, where $\alpha = (1+\sqrt{5})/2$, $\beta = (1-\sqrt{5})/2$ are the roots of $X^2 - X - 1$. We can find these constants by computing $B(4), B(5), \ldots, B(11)$ (using Eq. (1)) and then solving for the values of the constants $c_1, \ldots, c_8$.

When we do so, we find

$$
\begin{aligned}
c_1 &= \frac{2}{5} & c_2 &= -\frac{2}{25}\sqrt{5} - 2 \\
c_3 &= \frac{2}{5} & c_4 &= \frac{2}{25}\sqrt{5} - 2 \\
c_5 &= 1 & c_6 &= 1 \\
c_7 &= 0 & c_8 &= 0
\end{aligned}
$$

A little simplification, using the fact that $F_n = (\alpha^n - \beta^n)/(\alpha - \beta)$, leads to

Theorem 3. *Let $B(n)$ denote the number of square occurrences in X_n. Then*

$$B(n+1) = \frac{4}{5}nF_{n+1} - \frac{2}{5}(n+6)F_n - 4F_{n-1} + n + 1$$

for $n \geq 3$.

This statement corrects a small error in Theorem 2 in [11] (the coefficient of F_{n-1} was wrong; note that their F and their Fibonacci words are indexed differently from ours), which was first pointed out to us by Kalle Saari.

In a similar way, we can count the cube occurrences in X_n. Using analysis exactly like the square case, we easily find

Theorem 4. *Let $C(n)$ denote the number of cube occurrences in the Fibonacci word X_n. Then for $n \geq 3$ we have*

$$C(n) = (d_1 n + d_2)\alpha^n + (d_3 n + d_4)\beta^n + d_5 n + d_6$$

where

$$
\begin{aligned}
d_1 &= \frac{3-\sqrt{5}}{10} & d_2 &= \frac{17}{50}\sqrt{5} - \frac{3}{2} \\
d_3 &= \frac{3+\sqrt{5}}{10} & d_4 &= -\frac{17}{50}\sqrt{5} - \frac{3}{2} \\
d_5 &= 1 & d_6 &= -1.
\end{aligned}
$$

We now turn to a question of Chuan and Droubay. Let us consider the prefixes of $\mathbf{f}$. For each prefix of length n, form all of its n shifts, and let us count the number of these shifts that are palindromes; call this number $d(n)$. (Note that in the case where a prefix is a power, two different shifts could be identical; we count these with multiplicity.)

Chuan [7, Thm. 7, p. 254] proved

Theorem 5. *For $i > 2$ we have $d(F_i) = 0$ iff $i \equiv 0 \pmod 3$.*

Proof. Along the way we actually prove a lot more, characterizing $d(n)$ for all n, not just those n equal to a Fibonacci number.

We start by showing that $d(n)$ takes only three values: 0, 1, and 2. To do this, we construct an automaton accepting the language

$$\{(n,i)_F \ : \ (0 \leq i < n) \ \wedge \ \mathbf{f}[i..n-1]\mathbf{f}[0..i-1] \text{ is a palindrome } \}.$$

From this we construct the linear representation (u, M_0, M_1, v) of $d(n)$ as discussed above; it has rank 27.

The range of c is finite if the monoid $\mathcal{M} = \langle M_0, M_1 \rangle$ is finite. This can be checked with a simple queue-based algorithm, and $\mathcal{M}$ turns out to have cardinality 151. From these a simple computation proves

$$\{uMv \ : \ M \in \mathcal{M}\} = \{0, 1, 2\},$$

and so our claim about the range of c follows.

Now that we know the range of c we can create predicates $P_0(n), P_1(n), P_2(n)$ asserting that (a) there are no length-n shifts that are palindromes (b) there is exactly one shift that is a palindrome and (c) more than one shift is a palindrome, as follows:

$$P_0 : \neg\exists i, (0 \leq i < n), \mathbf{f}[i..n-1]\mathbf{f}[0..i-1] \text{ is a palindrome}$$

$$P_1 : \exists i, (0 \leq i < n), \mathbf{f}[i..n-1]\mathbf{f}[0..i-1] \text{ is a palindrome and } \neg\exists j \neq i (0 \leq j < n), \mathbf{f}[j..n-1]\mathbf{f}[0..j-1]$$

$$P_2 : \exists i, j, 0 \leq i < j < n \mathbf{f}[i..n-1]\mathbf{f}[0..i-1] \text{ and } \mathbf{f}[j..n-1]\mathbf{f}[0..j-1] \text{ are both palindromes}$$

For each one, we can compute a finite automaton characterizing the Fibonacci representations of those n for which $d(n)$ equals, respectively, 0, 1, and 2.

For example, we computed the automaton corresponding to P_0, and it is displayed in Figure 1 below.

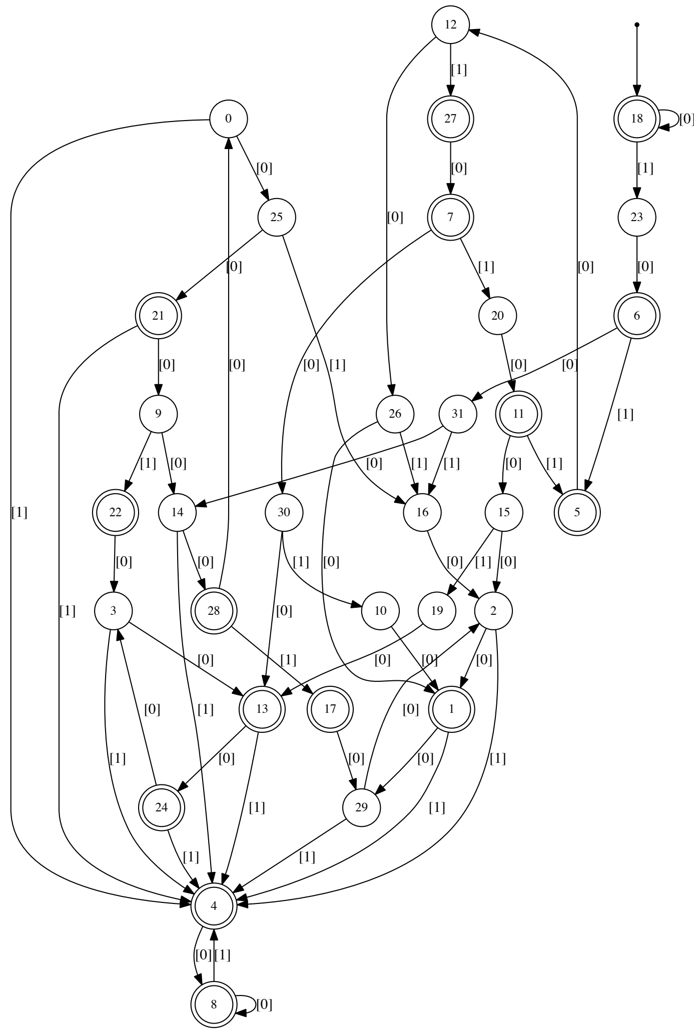

Figure 1: Automaton accepting lengths of prefixes for which no shifts are palindromes

By tracing the path labeled 10^* starting at the initial state labeled 18, we see that the "finality" of the states encountered is ultimately periodic with period 3, proving Theorem 5. $\square$

To finish this section, we reprove a result concerning maximal repetitions in $\mathbf{f}$. Let $p(x)$ denote the length of the least period of x. If $\mathbf{x} = a_0 a_1 \cdots$, by $\mathbf{x}[i..j]$ we mean $a_i a_{i+1} \cdots a_j$. Following Kolpakov and Kucherov [13], we say that $\mathbf{f}[i..i+n-1]$ is a *maximal repetition* if

(a) $p(\mathbf{f}[i..i+n-1]) \leq n/2$;

(b) $p(\mathbf{f}[i..i+n-1]) < p(\mathbf{f}[i..i+n])$;

(c) If $i > 0$ then $p(\mathbf{f}[i..i+n-1]) < p(\mathbf{f}[i-1..i+n-1])$.

They proved the following result on the number $\mathrm{mr}(F_n)$ of occurrences of maximal repetitions in the prefix of $\mathbf{f}$ of length F_n:

Theorem 6. *For $n \geq 5$ we have $\mathrm{mr}(F_n) = 2F_{n-2} - 3$.*

Proof. We create an automaton for the language

$$\{(n, i, j)_F \ : \ 0 \leq i \leq j < n \text{ and } \mathbf{f}[i..j] \text{ is a maximal repetition of } \mathbf{f}[0..n-1]\},$$

using the predicate

$$\begin{aligned}
&(i \leq j) \ \wedge \ (j < n) \ \wedge \ \exists p \text{ with } 1 \leq p \leq (j+1-i)/2 \text{ such that} \\
&\qquad\qquad ((\forall k \leq j-i-p \ \mathbf{f}[i+k] = \mathbf{f}[i+k+p]) \ \wedge \\
&(i \geq 1) \implies (\forall q \text{ with } 1 \leq q \leq p \ \exists \ell \leq j-i-q+1 \ \mathbf{f}[i-i+\ell] \neq \mathbf{f}[i-1+\ell+q]) \ \wedge \\
&(j+1 \leq n-1) \implies (\forall r \text{ with } 1 \leq r \leq p \ \exists m \leq j+1-r-i \ \mathbf{f}[i+m] \neq \mathbf{f}[i+m+r])).
\end{aligned}$$

Here the second line of the predicate specifies that there is a period p of $\mathbf{f}[i..j]$ corresponding to a repetition of exponent at least 2. The third line specifies that no period q of $\mathbf{f}[i-1..j]$ (when this makes sense) can be $\leq p$, and the fourth line specifies that no period r of $\mathbf{f}[i..j+1]$ (when $j+1 \leq n-1$) can be $\leq p$.

From the automaton we deduce a linear representation (u, μ, v) of rank 59. Since $(F_n)_F = 10^{n-2}$, it suffices to compute the minimal polynomial of $M_0 = \mu(0)$. When we do this, we discover it is $X^4(X^2-X-1)(X-1)^2(X+1)^2$. It follows from the theory of linear recurrences that

$$\mathrm{mr}(F_n) = e_1\alpha^n + e_2\beta^n + e_3 n + e_4 + (e_5 n + e_6)(-1)^n$$

for constants $e_1, e_2, e_3, e_4, e_5, e_6$ and $n \geq 6$. When we solve for $e_1, \ldots, e_6$ by using the first few values of $\mathrm{mr}(F_n)$ (computed from the linear representation or directly) we discover that $e_1 = (3\sqrt{5}-5)/5$, $e_2 = (-3\sqrt{5}-5)/5$, $e_3 = e_5 = e_6 = 0$, and $e_4 = -3$. From this the result immediately follows. □

In fact, we can prove even more.

Theorem 7. *For $n \geq 0$ the difference $\mathrm{mr}(n+1) - \mathrm{mr}(n)$ is either 0 or 1. Furthermore there is a finite automaton with 10 states that accepts $(n)_F$ precisely when $\mathrm{mr}(n+1) - \mathrm{mr}(n) = 1$.*

Proof. Every maximal repetition $\mathbf{f}[i..j]$ of $\mathbf{f}[0..n-1]$ is either a maximal repetition of $\mathbf{f}[0..n]$ with $j \leq n-1$, or is a maximal repetition with $j = n-1$ that, when considered in $\mathbf{f}[0..n]$, can be extended one character to the right to become one with $j = n$. So the only maximal repetitions of $\mathbf{f}[0..n]$ not (essentially) counted by $\mathrm{mr}(n)$ are those such that

$$\begin{aligned}
&\mathbf{f}[i..n] \text{ is a maximal repetition of } \mathbf{f}[0..n] \text{ and} \\
&\qquad\qquad \mathbf{f}[i..n-1] \text{ is } \textit{not} \text{ a maximal repetition of } \mathbf{f}[0..n-1]. \quad (3)
\end{aligned}$$

We can easily create a predicate asserting this latter condition, and from this obtain the linear representation of $\mathrm{mr}(n+1) - \mathrm{mr}(n)$:

$$u = [0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,]$$

$$\mu(0) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&1&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0 \end{bmatrix}$$

$$\mu(1) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&1&1&0&0&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0 \end{bmatrix}$$

$$v = [0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1]$$

We now use the trick we previously used for the proof of Theorem 5; the monoid generated by $\mu(0)$ and $\mu(1)$ has size 61 and for each matrix M in this monoid we have $uMv \in \{0, 1\}$. It follows that $\mathrm{mr}(n+1) - \mathrm{mr}(n) \in \{0, 1\}$ for all $n \geq 0$.

Knowing this, we can now build an automaton accepting those n for which there exists an i for which (3) holds. When we do so we get the automaton depicted below in Figure 2.

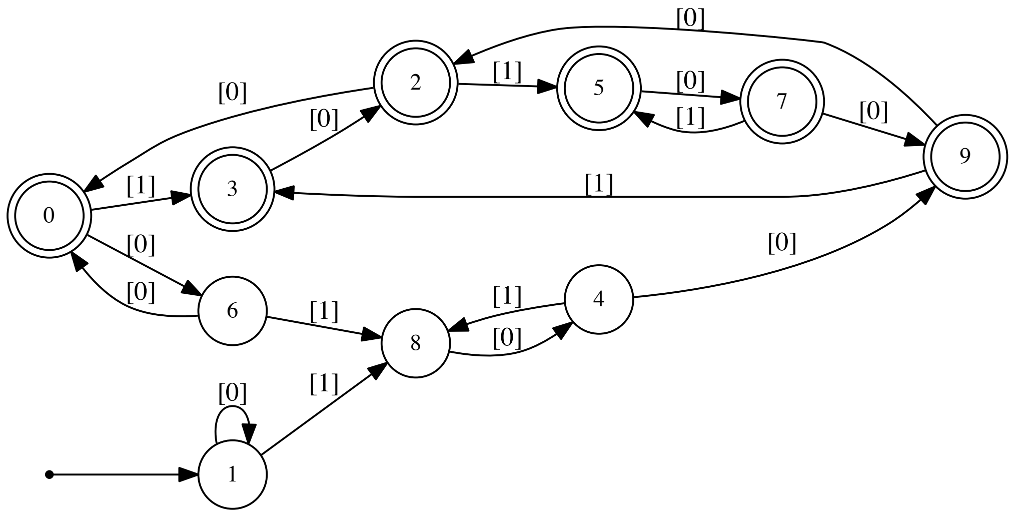

Figure 2: Automaton accepting $(n)_F$ such that $\mathrm{mr}(n+1) - \mathrm{mr}(n) = 1$

$\square$

3 Abelian properties

Our decision procedure does not apply, in complete generality, to abelian properties of infinite words. This is because there is no obvious way to express assertions like $\psi(x) = \psi(x')$ for two factors x, x' of an infinite word. (Here $\psi : \Sigma^* \rightarrow \mathbb{N}^{|\Sigma|}$ is the Parikh map that sends a word to the number of occurrences of each letter.) Indeed, in the 2-automatic case it is provable that there is at least one abelian property that is inexpressible [16, §5.2].

However, the special nature of the Fibonacci word **f** allows us to mechanically prove some assertions involving abelian properties. In this section we describe how we did this.

By an *abelian square of order* n we mean a factor of the form xx' where $\psi(x) = \psi(x')$, where $n = |x|$. In a similar way we can define abelian cubes and higher powers.

We start with the elementary observation that **f** is defined over the alphabet $\{0, 1\}$. Hence, to understand the abelian properties of a factor x it suffices to know $|x|$ and $|x|_0$. Next, we observe that the map that sends n to $a_n := |\mathbf{f}[0..n-1]|_0$ (that is, the number of 0's in the length-n prefix of **f**), is actually *synchronized* (see [5, 3, 4, 12]). That is, there is a DFA accepting the Fibonacci representation of the pairs (n, a_n). In fact we have the following

Theorem 8. *Suppose the Fibonacci representation of n is $e_1 e_2 \cdots e_i$. Then $a_n = [e_1 e_2 \cdots e_{i-1}]_F + e_i$.*

Proof. First, we observe that an easy induction on m proves that $|X_m|_0 = F_{m-1}$ for $m \geq 2$. We will use this in a moment.

The theorem's claim is easily checked for $n = 0, 1$. We prove it for $F_{m+1} \leq n < F_{m+2}$ by induction on m. The base case is $m = 1$, which corresponds to $n = 1$.

Now assume the theorem's claim is true for $m - 1$; we prove it for m. Write $(n)_F = e_1 e_2 \cdots e_m$. Then, using the fact that $\mathbf{f}[0..F_{m+2} - 1] = X_{m+2} = X_{m+1} X_m$, we get

$$\begin{aligned} |\mathbf{f}[0..n-1]|_0 &= |\mathbf{f}[0..F_{m+1} - 1]|_0 + |\mathbf{f}[F_{m+1}..n-1]|_0 \\ &= |X_{m+1}|_0 + |\mathbf{f}[0..n-1-F_{m+1}]|_0 \\ &= F_m + |\mathbf{f}[0..n-1-F_{m+1}|_0 \\ &= F_m + [e_2 \cdots e_{m-1}]_F + e_m \\ &= [e_1 \cdots e_{m-1}]_F + e_m, \end{aligned}$$

as desired. $\square$

In fact, the synchronized automaton for $(n, a_n)_F$ is given in the following diagram:

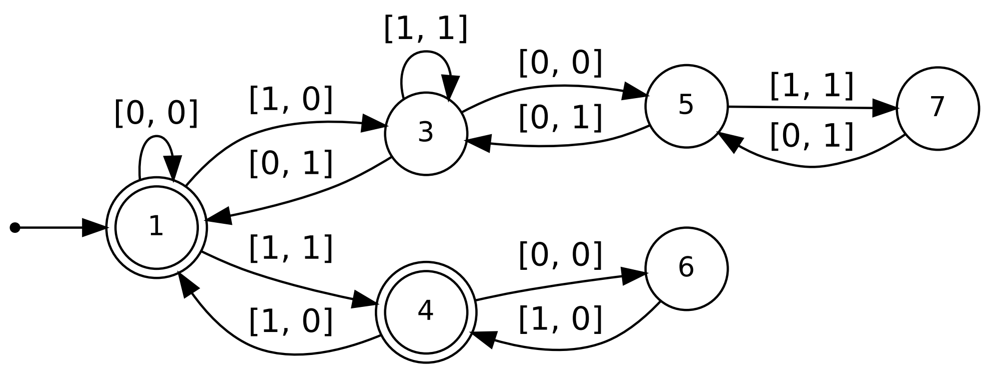

Figure 3: Automaton accepting $(n, a_n)_F$

Here the missing state numbered 2 is a "dead" state that is the target of all undrawn transitions.

The correctness of this automaton can be checked using our prover. Letting $\mathrm{ZC}(x, y)$ denote 1 if $(x, y)_F$ is accepted, it suffices to check that

1. $\forall x \ \exists y \ \mathrm{ZC}(x, y) = 1$ (that is, for each x there is at least one corresponding y accepted);

2. $\forall x \ \forall y \ \forall z \ (\mathrm{ZC}(x, y) = \mathrm{ZC}(x, z)) \implies y = z$ (that is, for each x at most one corresponding y is accepted);

3. $\forall x\ \forall y\ ((\mathrm{ZC}(x,y)=1)\ \wedge\ (\mathbf{f}[x]=1)) \implies (\mathrm{ZC}(x+1,y+1)=1)$;

4. $\forall x\ \forall y\ ((\mathrm{ZC}(x,y)=1)\ \wedge\ (\mathbf{f}[x]=0)) \implies (\mathrm{ZC}(x+1,y)=1)$;

Another useful automaton computes, on input n, i, j the function

$$\mathrm{FAB}(n,i,j) := |\mathbf{f}[i..i+n-1]|_0 - |\mathbf{f}[j..j+n-1]|_0 = a_{i+n} - a_i - a_{j+n} + a_j.$$

From the known fact that the factors of **f** are "balanced" we know that FAB takes only the values $-1, 0, 1$. This automaton can be deduced from the one above. However, we calculated it by "guessing" the right automaton and then verifying the correctness with our prover.

The automaton for $\mathrm{FAB}(n,i,j)$ has 30 states, numbered from 1 to 30. Inputs are in Σ_2^3. The transitions, as well as the outputs, are given in the table below.

q	$[0,0,0]$	$[0,0,1]$	$[0,1,0]$	$[0,1,1]$	$[1,0,0]$	$[1,0,1]$	$[1,1,0]$	$[1,1,1]$	$\tau(q)$
1	1	2	3	4	4	5	6	7	0
2	8	1	9	3	3	4	10	6	0
3	11	12	1	2	2	13	4	5	0
4	14	11	8	1	1	2	3	4	0
5	15	11	16	1	1	2	3	4	1
6	17	18	8	1	1	2	3	4	-1
7	19	18	16	1	1	2	3	4	0
8	1	2	3	4	4	20	6	21	0
9	11	12	1	2	2	22	4	20	0
10	18	23	1	2	2	13	4	5	-1
11	1	2	3	4	4	5	24	25	0
12	8	1	9	3	3	4	26	24	0
13	16	1	27	3	3	4	10	6	1
14	1	2	3	4	4	20	24	28	0
15	2	13	4	5	5	20	25	28	-1
16	2	13	4	5	5	20	7	21	-1
17	3	4	10	6	6	21	24	28	1
18	3	4	10	6	6	7	24	25	1
19	4	5	6	7	7	21	25	28	0
20	15	14	16	8	8	1	9	3	1
21	19	17	16	8	8	1	9	3	0
22	16	8	27	9	9	3	29	10	1
23	9	3	29	10	10	6	26	24	1
24	17	18	14	11	11	12	1	2	-1
25	19	18	15	11	11	12	1	2	0
26	18	23	11	12	12	30	2	13	-1
27	12	30	2	13	13	22	5	20	-1
28	19	17	15	14	14	11	8	1	0
29	18	23	1	2	2	22	4	20	-1
30	16	1	27	3	3	4	26	24	1

Table 1: Automaton to compute FAB

Once we have guessed the automaton, we can verify it as follows:

1. $\forall i\ \forall j\ \text{FAB}[0][i][j] = 0$. This is the basis for an induction.

2. Induction steps:

    - $\forall i\ \forall j\ \forall n\ (\mathbf{f}[i+n] = \mathbf{f}[j+n]) \implies (\text{FAB}[n][i][j] = \text{FAB}[n+1][i][j])$.
    - $\forall i\ \forall j\ \forall n\ ((\mathbf{f}[i+n] = 0) \wedge (\mathbf{f}[j+n] = 1)) \implies (((\text{FAB}[n][i][j] = -1) \wedge (\text{FAB}[n+1][i][j] = 0)) \vee ((\text{FAB}[n][i][j] = 0) \wedge (\text{FAB}[n+1][i][j] = 1)))$

- $\forall i\ \forall j\ \forall n\ ((\mathbf{f}[i+n]=0) \wedge (\mathbf{f}[j+n]=1)) \implies (((\mathrm{FAB}[n][i][j]=1) \wedge (\mathrm{FAB}[n+1][i][j]=0)) \vee ((\mathrm{FAB}[n][i][j]=0) \wedge (\mathrm{FAB}[n+1][i][j]=-1)))$.

As the first application, we prove

Theorem 9. *The Fibonacci word* $\mathbf{f}$ *has abelian squares of all orders.*

Proof. We use the predicate

$$\exists i\ (\mathrm{FAB}[n][i][i+n]=0).$$

The resulting automaton accepts all $n \geq 0$. The total computing time was 141 ms. □

Cummings and Smyth [9] counted the total number of all occurrences of (nonempty) abelian squares in the Fibonacci words X_i. We can do this by using the predicate

$$(k>0) \wedge (i+2k \leq n) \wedge (\mathrm{FAB}[k][i][i+k]=0),$$

using the techniques in Section 2 and considering the case where $n = F_i$.

When we do, we get a linear representation of rank 127 that counts the total number $w(n)$ of occurrences of abelian squares in the prefix of length n of the Fibonacci word.

To recover the Cummings-Smyth result we compute the minimal polynomial of the matrix M_0 corresponding to the predicate above. It is

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

This means that $w(F_n)$, that is, w evaluated at 10^{n-2} in Fibonacci representation, is a linear combination of the roots of this polynomial to the n'th power (more precisely, the $(n-2)$th, but this detail is unimportant). The roots of the polynomial are

$$-1, 1, (-1 \pm i\sqrt{3})/2, (3 \pm \sqrt{5})/2, (1 \pm i\sqrt{3})/2, (-1 \pm \sqrt{5})/2, (1 \pm \sqrt{5})/2.$$

Solving for the coefficients as we did in Section 2 we get

Theorem 10. *For all* $n \geq 0$ *we have*

$$w(F_n) = c_1\left(\frac{3+\sqrt{5}}{2}\right)^n + c_1\left(\frac{3-\sqrt{5}}{2}\right)^n + c_2\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n +$$
$$c_3\left(\frac{1+i\sqrt{3}}{2}\right)^n + \overline{c_3}\left(\frac{1-i\sqrt{3}}{2}\right)^n + c_4\left(\frac{-1+i\sqrt{3}}{2}\right)^n + \overline{c_4}\left(\frac{-1-i\sqrt{3}}{2}\right)^n + c_5(-1)^n,$$

where

$$\begin{aligned} c_1 &= 1/40 \\ c_2 &= -\sqrt{5}/20 \\ c_3 &= (1-i\sqrt{3})/24 \\ c_4 &= i\sqrt{3}/24 \\ c_5 &= -2/15, \end{aligned}$$

and here $\overline{x}$ denotes complex conjugate. Here the parts corresponding to the constants c_3, c_4, c_5 form a periodic sequence of period 6.

Next, we turn to what is apparently a new result. Let $h(n)$ denote the total number of distinct factors (not occurrences of factors) that are abelian squares in the Fibonacci word X_n.

In this case we need the predicate

$$(k \geq 1) \wedge (i + 2k \leq n) \wedge (\text{FAB}[k][i][i+k] = 0) \wedge (\forall j < i\ (\exists t < 2k\ (\mathbf{f}[j+t] \neq \mathbf{f}[i+t]))).$$

We get the minimal polynomial

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

Using the same technique as above we get

Theorem 11. *For $n \geq 2$ we have $h(n) = a_1 c_1^n + \cdots + a_{10} c_{10}^n$ where*

$$\begin{aligned}
a_1 &= (-2+\sqrt{5})/20 \\
a_2 &= (-2-\sqrt{5})/20 \\
a_3 &= (5-\sqrt{5})/20 \\
a_4 &= (5+\sqrt{5})/20 \\
a_5 &= 1/30 \\
a_6 &= -5/6 \\
a_7 &= (1/12) - i\sqrt{3}/12 \\
a_8 &= (1/12) + i\sqrt{3}/12 \\
a_9 &= (1/6) + i\sqrt{3}/12 \\
a_{10} &= (1/6) - i\sqrt{3}/12
\end{aligned}$$

and

$$\begin{aligned}
c_1 &= (3+\sqrt{5})/2 \\
c_2 &= (3-\sqrt{5})/2 \\
c_3 &= (1+\sqrt{5})/2 \\
c_4 &= (1-\sqrt{5})/2 \\
c_5 &= -1 \\
c_6 &= 1 \\
c_7 &= (1/2) + i\sqrt{3}/2 \\
c_8 &= (1/2) - i\sqrt{3}/2 \\
c_9 &= (-1/2) + i\sqrt{3}/2 \\
c_{10} &= (-1/2) - i\sqrt{3}/2.
\end{aligned}$$

For another new result, consider counting the total number $a(n)$ of distinct factors of length $2n$ of the infinite word $\mathbf{f}$ that are abelian squares.

This function is rather erratic. The following table gives the first few values:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$a(n)$	1	3	5	1	9	5	5	15	3	13	13	5	25	9	15	25	1	27	19	11

We use the predicate

$$(n \geq 1) \wedge (\mathrm{FAB}[n][i][i+n] = 0) \wedge (\forall j < i \; (\exists t < 2n \; (\mathbf{f}[j+t] \neq \mathbf{f}[i+t]))).$$

to create the matrices and vectors.

Theorem 12. *$a(n) = 1$ infinitely often and $a(n) = 2n-1$ infinitely often. More precisely $a(n) = 1$ iff $(n)_F = 1$ or $(n)_F = (100)^i 101$ for $i \geq 0$, and $a(n) = 2n-1$ iff $(n)_F = 10^i$ for $i \geq 0$.*

Proof. For the first statement, we create a DFA accepting those $(n)_F$ for which $a(n) = 1$, via the predicate

$$\forall i \; \forall j \; ((\mathrm{FAB}[n][i][i+n] = 0) \wedge (\mathrm{FAB}[n][j][j+n] = 0)) \implies (\forall t < 2n \; (\mathbf{f}[j+t] = \mathbf{f}[i+t])).$$

The resulting 6-state automaton accepts the set specified.

For the second result, we first compute the minimal polynomial of the matrix M_0 of the linear representation. It is $x^5(x-1)(x+1)(x^2-x-1)$. This means that, for $n \geq 5$, we have $a(F_n) = c_1 + c_2(-1)^n + c_3\alpha^n + c_4\beta^n$ where, as usual, $\alpha = (1+\sqrt{5})/2$ and $\beta = (1-\sqrt{5})/2$. Solving for the constants, we determine that $a(F_n) = 2F_n - 1$ for $n \geq 2$, as desired.

To show that these are the only cases for which $a(n) = 2n-1$, we use a predicate that says that there are not at least three different factors of length $2n$ that are not abelian squares. Running this through our program results in only the cases previously discussed. $\square$

Finally, we turn to abelian cubes. Unlike the case of squares, some orders do not appear in $\mathbf{f}$.

Theorem 13. *The Fibonacci word $\mathbf{f}$ contains, as a factor, an abelian cube of order n iff $(n)_F$ is accepted by the automaton below.*

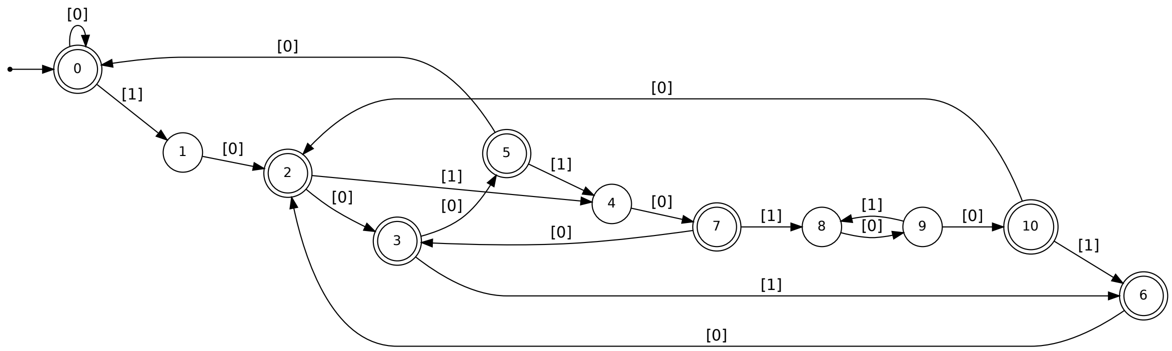

Figure 4: Automaton accepting orders of abelian cubes in $\mathbf{f}$

Theorem 8 has the following interesting corollary.

Corollary 14. *Let* $h : \{0,1\}^* \to \Delta^*$ *be an arbitrary morphism such that* $h(01) \neq \epsilon$*. Then* $h(\mathbf{f})$ *is an infinite Fibonacci-automatic word.*

Proof. From Theorem 8 we see that there is a predicate $\mathrm{ZC}(n, n')$ which is true if $n' = |\mathbf{f}[0..n-1]|_0$ and false otherwise, and this predicate can be implemented as a finite automaton taking the inputs n and n' in Fibonacci representation.

Suppose $h(0) = w$ and $h(1) = x$. Now, to show that h($\mathbf{f}$) is Fibonacci-automatic, it suffices to show that, for each letter $a \in \Delta$, the language of "fibers"

$$L_a = \{(n)_F : (h(\mathbf{f}))[n] = a\}$$

is regular.

To see this, we write a predicate for the n in the definition of L_a, namely

$$\begin{aligned} &\exists q\ \exists r_0\ \exists r_1\ \exists m\ (q \leq n < q + |h(\mathbf{f}[m])|)\ \wedge\ \mathrm{ZC}(m, r_0)\ \wedge\ (r_0 + r_1 = m) \wedge \\ &\qquad (r_0|w| + r_1|x| = q)\ \wedge\ ((\mathbf{f}[m] = 0\ \wedge\ w[n-q] = a)\ \vee\ (\mathbf{f}[m] = 1\ \wedge\ x[n-q] = a)). \end{aligned}$$

Notice that the predicate looks like it uses multiplication, but this multiplication can be replaced by repeated addition since $|w|$ and $|x|$ are constants here.

Unpacking this predicate we see that it asserts the existence of m, q, r_0, and r_1 having the meaning that

- the n'th symbol of h($\mathbf{f}$) lies inside the block $h(\mathbf{f}[m])$ and is in fact the $(n-q)$'th symbol in the block (with the first symbol being symbol 0)
- $\mathbf{f}[0..m-1]$ has r_0 0's in it
- $\mathbf{f}[0..m-1]$ has r_1 1's in it

- the length of $h(\mathbf{f}[0..m-1])$ is q

Since everything in this predicate is in the logical theory $(\mathbb{N}, +, <, F)$ where F is the predicate for the Fibonacci word, the language L_a is regular. □

Remark 15. Notice that everything in this proof goes through for other numeration systems, provided the original word has the property that the Parikh vector of the prefix of length n is synchronized.

4 Acknowledgments

We thank Kalle Saari for bringing our attention to the small error in [11]. We thank Narad Rampersad and Michel Rigo for useful suggestions.

References

[1] J.-P. Allouche, K. Scheicher, and R. F. Tichy. Regular maps in generalized number systems. *Math. Slovaca* **50** (2000), 41–58.

[2] J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*, Vol. 137 of *Encylopedia of Mathematics and Its Applications*. Cambridge University Press, 2011.

[3] A. Carpi and V. D'Alonzo. On the repetitivity index of infinite words. *Internat. J. Algebra Comput.* **19** (2009), 145–158.

[4] A. Carpi and V. D'Alonzo. On factors of synchronized sequences. *Theoret. Comput. Sci.* **411** (2010), 3932–3937.

[5] A. Carpi and C. Maggi. On synchronized sequences and their separators. *RAIRO Inform. Théor. App.* **35** (2001), 513–524.

[6] E. Charlier, N. Rampersad, and J. Shallit. Enumeration and decidable properties of automatic sequences. *Internat. J. Found. Comp. Sci.* **23** (2012), 1035–1066.

[7] W.-F. Chuan. Symmetric Fibonacci words. *Fibonacci Quart.* **31** (1993), 251–255.

[8] M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inform. Process. Lett.* **12** (1981), 244–250.

[9] L. J. Cummings and W. F. Smyth. Weak repetitions in strings. *J. Combin. Math. Combin. Comput.* **24** (1997), 33–48.

[10] C. F. Du, H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, II: Related sequences and avoidability. Submitted, 2015.

[11] A. S. Fraenkel and J. Simpson. The exact number of squares in Fibonacci words. *Theoret. Comput. Sci.* **218** (1999), 95–106.

[12] D. Goc, L. Schaeffer, and J. Shallit. The subword complexity of k-automatic sequences is k-synchronized. In M.-P. Béal and O. Carton, editors, *DLT 2013*, Vol. 7907 of *Lecture Notes in Computer Science*, pp. 252–263. Springer-Verlag, 2013.

[13] R. Kolpakov and G. Kucherov. On maximal repetitions in words. In G. Ciobanu and G. Păun, editors, *Fundamentals of Computation Theory: FCT '99*, Vol. 1684 of *Lecture Notes in Computer Science*, pp. 374–385. Springer-Verlag, 1999.

[14] M. Morse and G. A. Hedlund. Symbolic dynamics II. Sturmian trajectories. *Amer. J. Math.* **62** (1940), 1–42.

[15] H. Mousavi, L. Schaeffer, and J. Shallit. Decision algorithms for Fibonacci-automatic words, I: Basic results. Submitted, 2015.

[16] L. Schaeffer. Deciding properties of automatic sequences. Master's thesis, University of Waterloo, 2013.