RAIRO-Theor. Inf. Appl. 57 (2023) 1
https://doi.org/10.1051/ita/2022010

RAIRO - Theoretical Informatics and Applications
www.rairo-ita.org

PROPERTIES OF A TERNARY INFINITE WORD

James Currie[1], Pascal Ochem[2], Narad Rampersad[1] and Jeffrey Shallit[3,*]

Abstract. We study the properties of the ternary infinite word

$$\mathbf{p} = 012102101021012101021012\cdots,$$

that is, the fixed point of the map $h : 0 \to 01,\ 1 \to 21,\ 2 \to 0$. We determine its factor complexity, critical exponent, and prove that it is 2-balanced. We compute its abelian complexity and determine the lengths of its bispecial factors. Finally, we give a characterization of $\mathbf{p}$ in terms of avoided factors.

Mathematics Subject Classification. 11B85, 68R15, 03D05, 68Q45.

Received September 19, 2022. Accepted December 5, 2022.

1. Introduction

One of the themes of combinatorics on words is the study of particular infinite words with interesting properties. For example, in one of the very earliest results in this area, Thue proved that the Thue-Morse word

$$\mathbf{t} = 0110100110010110\cdots$$

avoids *overlaps*: factors of the form $axaxa$ with a a single letter and x a possibly empty word [1, 22]. He also proved that the word

$$\mathbf{vtm} = 2102012101202102012021\cdots,$$

avoids *squares*: factors of the form yy with y nonempty.

More generally, one can study other kinds of repetitions. We say that a finite word $w = w[1..n]$ has *period* $p \geq 1$ if $w[i] = w[i+p]$ for $1 \leq i \leq n-p$. The smallest period of a word w is called *the* period, and we write it as $\mathrm{per}(w)$. The *exponent* of a finite word w, written $\exp(w)$ is defined to be $|w|/\mathrm{per}(w)$. We say a word (finite or infinite) is α*-free* if the exponent of all its nonempty factors is $< \alpha$. We say a word is α^+*-free* if the exponent

Keywords and phrases: Factor complexity, critical exponent, Pisot numeration system, automatic sequence, linear representation, finite automaton, synchronized sequence, balanced word, abelian complexity, recurrence, appearance.

[1] Department of Math/Stats, University of Winnipeg, 515 Portage Ave., Winnipeg R3B 2E9, MB, Canada.
[2] LIRMM, CNRS, Université de Montpellier, France.
[3] School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

* Corresponding author: shallit@uwaterloo.ca

of all its nonempty factors is $\geq \alpha$. The *critical exponent* of a finite or infinite word x is the supremum, over all nonempty finite factors w of x, of $\exp(w)$; it is written $\mathrm{ce}(x)$. The critical exponent of a word can be either rational or irrational. If it is rational, then it can either be attained by a particular finite factor, or not attained. For example, the critical exponent of both **t** and **vtm** is 2, but it is attained in the former case and not attained in the latter. If the critical exponent α is attained, we typically write it as α^+. An overlap is a 2^+ power, so the Thue-Morse word is 2^+-free.

The Fibonacci word

$$\mathbf{f} = 01001010010010100 1010\cdots$$

is the fixed point of the morphism $0 \to 01$, $1 \to 0$. Karhumäki proved [11] that **f** has no fourth powers (*i.e.*, blocks of the form $xxxx$, with x nonempty), and Mignosi and Pirillo [12] proved that the critical exponent of **f** is $(5+\sqrt{5})/2$.

Another aspect of infinite words that has been studied is *balance*. We say that a finite or infinite word x is t-balanced if for all equal-length factors y, z of x, and all letters a, the inequality $||y|_a - |z|_a| \leq t$ is satisfied. As is well-known, the Fibonacci word **f** (and more generally, every Sturmian word) is 1-balanced [5, 13].

A third aspect is *factor complexity*, also called *subword complexity*. For an infinite word **x**, the factor complexity function $\rho_{\mathbf{x}}(n)$ counts the number of distinct factors of length n in **x**. Morse and Hedlund [13] proved that $\rho_{\mathbf{f}}(n) = n+1$ for all $n \geq 0$. There is also the abelian analogue of factor complexity, where we count two factors as the same if they are permutations of each other [17].

In this note we study various aspects of the infinite ternary word

$$\mathbf{p} = 0121021010210121010210 12\cdots,$$

fixed point of the map h sending $0 \to 01$, $1 \to 21$, $2 \to 0$. This word is not automatic (because, as we will see, letters occur with irrational densities). It is not Sturmian (because it is over a 3-letter alphabet). Neither is it episturmian [10], because its set of subwords is not closed under reversal: **p** contains 02, but avoids 20.

We determine its factor complexity, its critical exponent, and prove that it is 2-balanced. A novel aspect of our work is that much of it is carried out using the `Walnut` theorem-prover [14, 20]. This software tool can prove or disprove assertions phrased in first-order logic about automatic sequences and their generalizations. These ideas were used previously to study the Tribonacci word [16], but the word **p** provides some new complications. All the files required to carry out the computations are available at the last author's website:

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

The word **p** has been studied previously. For example, it is sequence A287072 in the OEIS. It also appears implicitly in [3], where it is (up to renaming of the letters) the fixed point of the morphism c_2c_1. A very similar word is the word **G** studied in [21, Sect. 3.2]. Indeed, the factors of **p** can be obtained from the factors of **G** by reversal and applying the permutation (012). So **G** and **p** have the same critical exponent.

The results in this paper are applied to an avoidability problem in the companion paper [6].

2. A Pisot numeration system

The properties of **p** are intimately related to a particular numeration system $P4$, which we discuss now.

Consider the following linear recurrence:

$$X_1 = 1, X_2 = 2, X_3 = 4, X_4 = 7, \text{ and } X_n = X_{n-1} + X_{n-2} + X_{n-4} \text{ for } n \geq 0\ .$$

Table 1 gives the first few terms of this recurrence: This is sequence A005251 in the OEIS. It is easy to verify it also satisfies the simpler recurrence $X_n = 2X_{n-1} - X_{n-2} + X_{n-3}$ for $n \geq 4$.

TABLE 1. The recurrence X_n.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
X_n	1	2	4	7	12	21	37	65	114	200	351	616	1081	1897

TABLE 2. Representations in the $P4$ numeration system.

n	$(n)_P$	n	$(n)_P$
1	1	13	10001
2	10	14	10010
3	11	15	10011
4	100	16	10100
5	101	17	10101
6	110	18	10110
7	1000	19	11000
8	1001	20	11001
9	1010	21	100000
10	1011	22	100001
11	1100	23	100010
12	10000	24	100011

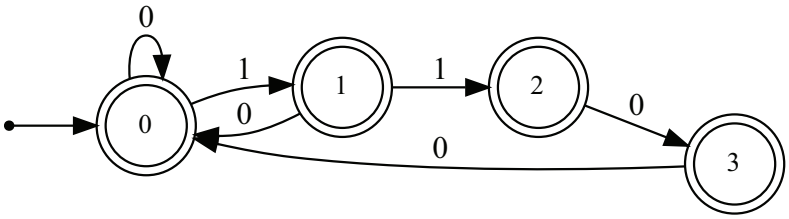

FIGURE 1. Automaton recognizing the valid representations.

We can consider representing natural numbers as a sum of the X_i, as follows: $N = \sum_{1 \le i \le t} e_i X_i$, where $e_i \in \{0, 1\}$. If we impose the following two rules on such a representation, namely

(a) $(e_i, e_{i+1}, e_{i+2}, e_{i+3}) \neq (1, 0, 1, 1)$; and
(b) $(e_i, e_{i+1}, e_{i+2}) \neq (1, 1, 1)$.

then this representation is unique, and can be written as a binary string $(N)_P := e_t e_{t-1} \cdots e_2 e_1$. For example, Table 2 gives the first few representations of numbers in this numeration system:

These representations are, in fact, the ones resulting by applying the greedy algorithm, and the conditions (a) and (b) follow from a theorem of Fraenkel [8].

A representation in this system $P4$ is valid if and only if it contains no occurrence of 111 or 1101. It follows that the language of all valid representations is recognized by the automaton in Figure 1.

The characteristic polynomial of the recurrence is $X^4 - X^3 - X^2 - 1 = (X + 1)(X^3 - 2X^2 + X - 1)$. The second term has one real zero, namely

$$\beta_1 = \frac{(100 + 12\sqrt{69})^{1/3}}{6} + \frac{2}{3(100 + 12\sqrt{69})^{1/3}} + 2/3 \doteq 1.754877666246692760049 5,$$

and two imaginary zeros that lie inside the unit circle. Therefore β_1 is a Pisot number, and so, by the results in [2, 9] we know that there is a finite automaton A recognizing the addition relation $x + y = z$, where x, y, z

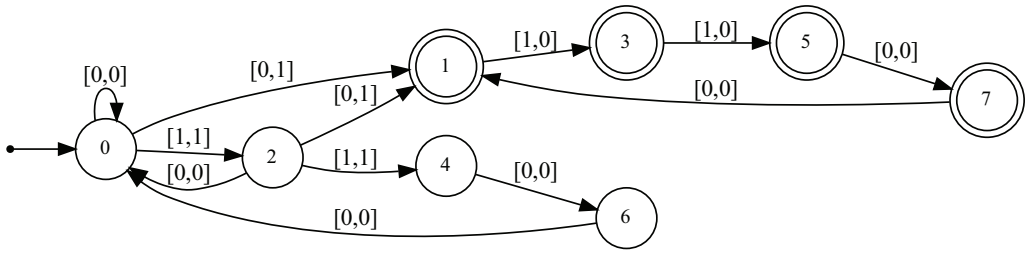

FIGURE 2. Incrementer in the numeration system.

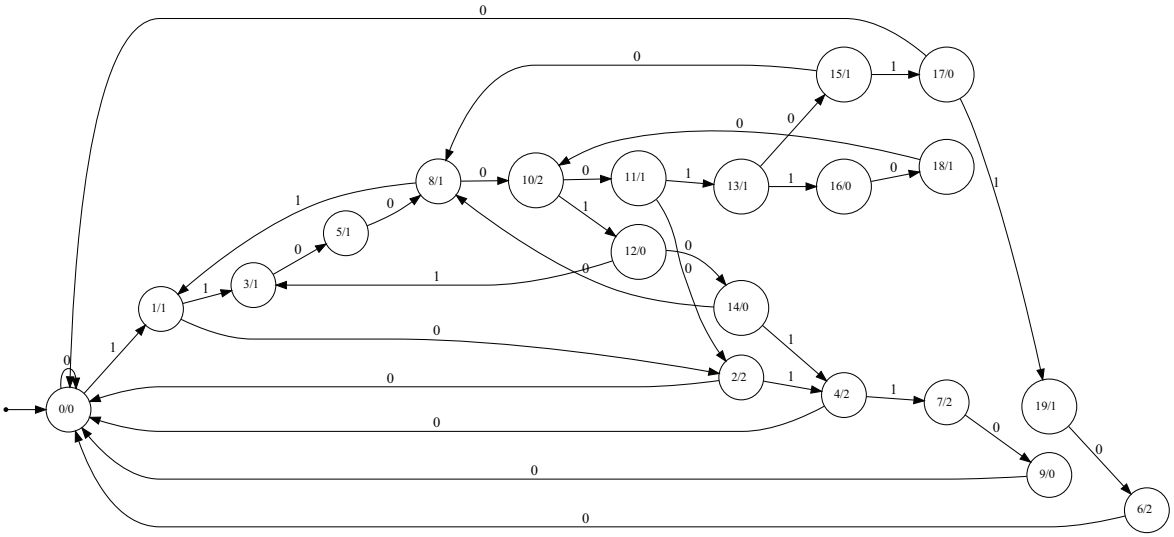

FIGURE 3. DFAO computing **p** in the numeration system.

are represented in the numeration system described above, with inputs in parallel and the shorter padded with leading zeros. Furthermore, there is an algorithm to compute A for any Pisot number.

Instead of applying this algorithm to compute A, we took a different approach. Namely, we "guessed" the adder using the Myhill-Nerode theorem, and then verified it using an incrementer constructed and verified by hand. This incrementer computes the relation $y = x + 1$, and is illustrated in Figure 2. Here the inputs are x and y in parallel, both represented in the numeration system $P4$.

Once we have the incrementer, the correctness of the adder can be verified as done in [15]. The adder has 64 states.

It turns out that the word **p** is automatic in this numeration system; this means it is computed by a deterministic finite automaton with output (DFAO) taking the $P4$ representation of n as input and outputting (in the last state reached) the value of $\mathbf{p}[n]$. The DFAO computing it is depicted in Figure 3.

This can be verified as follows. Let $h : 0 \to 01,\ 1 \to 21,\ 2 \to 0$, and let the automaton be represented by a morphism $\varphi : \Sigma \to \Sigma^*$ encoding the transitions of the automaton, and a coding $\tau : \Sigma \to \Delta$ giving the outputs for each state. Here the elements of Σ correspond to the state numbers of the automaton in Figure 3, except that we code the states $10, 11, \ldots, 19$ by the capital letters $A, B, \ldots, J$. Then we need to verify the following identities:

$$\tau(\varphi^n(0)) = h^n(0) \qquad \tau(\varphi^n(1)) = h^n(1)$$
$$\tau(\varphi^n(23)) = h^n(21) \qquad \tau(\varphi^n(45)) = h^n(21)$$

$$\begin{aligned}
\tau(\varphi^n(78)) &= h^n(21) & \tau(\varphi^n(9A)) &= h^n(02)\\
\tau(\varphi^n(BC)) &= h^n(10) & \tau(\varphi^n(2DE3)) &= h^n(2101)\\
\tau(\varphi^n(4FG8)) &= h^n(2101) & \tau(\varphi^n(HIA)) &= h^n(012)\\
\tau(\varphi^n(JA)) &= h^n(12) & \tau(\varphi^n(6)) &= h^n(2),
\end{aligned}$$

where $A = 10$, $B = 11$, etc. This can be done by a tedious induction on n, which we omit. Just to demonstrate one needed identity:

$$\begin{aligned}
\tau(\varphi^n(9A)) &= \tau(\varphi^{n-1}(0BC))\\
&= \tau(\varphi^{n-1}(0))\tau(\varphi^{n-1}(BC))\\
&= h^{n-1}(0)h^{n-1}(10)\\
&= h^{n-1}(010)\\
&= h^n(02),
\end{aligned}$$

as desired.

3. Factor complexity of $\mathbf{p}$

In this section we prove that the factor complexity of $\mathbf{p}$ is $2n+1$. This is also a consequence of more general results of [3]. Also see [4].

There is a well-established computational method for determining factor complexity, as discussed in [18]. The first step is to create an automaton that, given integers i, j, n as input, decides if $\mathbf{p}[i..i+n-1] = \mathbf{p}[j..j+n-1]$. Normally we would do this with the following `Walnut` command, where `PI` is a file containing the automaton in Figure 3:

```
def pisotfaceq "?msd_pisot4 At (t<n) => PI[i+t]=PI[j+t]":
```

However, in this case, the attempt fails. `Walnut` tries to determinize an automaton with 37351 states, and fails even with 5 Terabytes of storage and many hours of computation, so we need a different approach.

Instead, we use the approach discussed in [20, Sect. 6.3]. We "guess" an automaton for "pisotfaceq" using the Myhill-Nerode theorem. States are labeled with prefixes of Pisot representations. Two states are guessed to be the same if all suffixes of length ≤ 5 (representing $1 + 8^2 + \cdots + 8^5 = 37449$ words) give the same result. This gives us an automaton with 1080 states that we can represent as a file called `pisi.txt`.

Next, we can use induction and `Walnut` together to prove that our guess is correct. We do this as follows:

```
eval zeros "?msd_pisot4 Ai,j $pisi(i,j,0)":
eval induc "?msd_pisot4 Ai,j,n ($pisi(i,j,n) & PI[i+n]=PI[j+n])
   => $pisi(i,j,n+1)":
```

Both of these return `TRUE`, so our guess is correct.

Next, we create a linear representation (v, ζ, w) for the subword complexity function, using the following `Walnut` command:

```
eval pisotsc n "?msd_pisot4 Aj j<i => ~$pisi(i,j,n)":
```

This gives us a linear representation of rank 131. When we minimize it using a Maple program, we get this linear representation of rank 16:

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

$$\zeta(0) = \begin{bmatrix}
1 & 0\,0\,0 & 0 & 0\,0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,1\,0 & 0 & 0\,0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,0\,0 & 1 & 0\,0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,0\,0 & 0 & 0\,1 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,0\,0 & 0 & 0\,0 & 1 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
-1 & 0\,1\,0 & 0 & 0\,0 & 1 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
-1 & 0\,0\,0 & 1 & 0\,0 & 1 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
-1 & 0\,1\,0 & -1 & 0\,0 & 2 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
-2 & 0\,2\,0 & -1 & 0\,0 & 2 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,0\,0 & 0 & 0\,0 & 0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 \\
-4 & 0\,3\,0 & -1 & 0\,0 & 3 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,0\,0 & 0 & 0\,0 & 0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 \\
-5 & 0\,2\,0 & 0 & 0\,0 & 4 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0 & 0\,0\,0 & 0 & 0\,0 & 0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 \\
-8 & 0\,3\,0 & 0 & 0\,0 & 6 & 0\,0\,0\,0\,0\,0\,0\,0\,0 \\
-13 & 0\,5\,0 & -1 & 0\,0 & 10 & 0\,0\,0\,0\,0\,0\,0\,0\,0
\end{bmatrix}$$

$$\zeta(1) = \begin{bmatrix}
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0
\end{bmatrix}$$

$$w = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 9 \\ 11 \\ 13 \\ 15 \\ 17 \\ 21 \\ 27 \\ 31 \\ 37 \\ 49 \\ 55 \\ 87 \end{bmatrix}$$

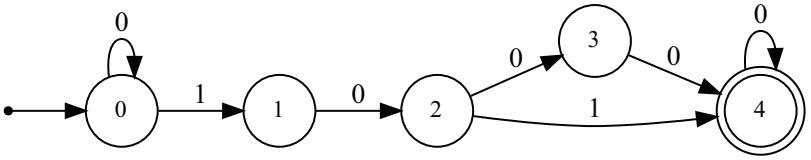

FIGURE 4. DFAO computing orders of overlaps in **p**.

With all these tools at our disposal, we can now prove the desired result.

Theorem 3.1. *The factor complexity of* **p** *is* $2n + 1$.

Proof. It remains to show that the linear representation (v, ζ, w) computes the function $2n + 1$. To do this, we compute a linear representation for $2n + 1$ using the following `Walnut` command:

```
eval c2n1 n "?msd_pisot4 i<2*n+1":
```

This gives us a linear representation (v', ζ', w') of rank 108. When we minimize it, we get *exactly* the same linear representation (v, ζ, w). This shows that the subword complexity is indeed $2n + 1$. □

4. Critical exponent of **p**

In this section we determine the critical exponent of **p**.

Theorem 4.1. *The critical exponent of* **p** *is*

$$\gamma + 1 = (8 - \beta_1 + 2\beta_1^2)/5 \doteq 2.480862716147236962394265321.$$

Here $\gamma + 1$ *is the real zero of* $5X^3 - 26X^2 + 43X - 23$.

Proof. We use the strategy previously employed for the Tribonacci word [16].

Now that we have a DFAO for **p**, we can find all the periods corresponding to overlaps. We do this with the following Walnut command.

```
def pisotlargepow "?msd_pisot4 Ei (n>=1) & At (t>=i & t<=i+n) => PI[t]=PI[t+n]":
```

This gives the automaton depicted in Figure 4. Constructing this automaton involved determinizing an NFA with 10,859 states and minimizing a DFA with 13,114,119 states; the calculation needed 90 Gigs of storage and 1100 seconds of CPU time.

As you can see from inspecting it, the periods are given by the representations 1010^* and 10000^*. We now compute the maximal repetitions in **p** by looking only at the periods given above:

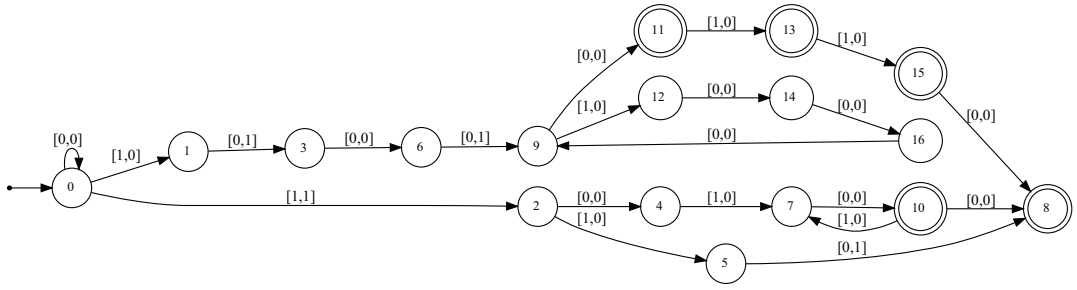

FIGURE 5. DFAO computing maximal powers.

```
def maximalreps "?msd_pisot4 Ei
(PI[i+n] != PI[i+n+p]) & $pisotlargepow(p) &
    (Aj (j<n & $pisotlargepow(p)) => PI[i+j] = PI[i+j+p])":
def highestpow "?msd_pisot4 (p>=1) & $pisotlargepow(p) &
    $maximalreps(n,p) & (Am $maximalreps(m,p) => m <= n)":
```

This gives the automaton in Figure 5. It recognizes all pairs (n, p) such that $n/p + 1$ is a maximal power in $\mathbf{p}$ with period p, for the p given above. By inspection we see these are as follows:

$$
\begin{aligned}
&[1,1][1,0][0,1]\\
&[1,1][0,0][1,0]([0,0][1,0])^*[0,0]\\
&[1,1][0,0][1,0]([0,0][1,0])^*[0,0][0,0]\\
&[1,0][0,1][0,0][0,1]([1,0][0,0][0,0][0,0])^*[0,0]\\
&[1,0][0,1][0,0][0,1]([1,0][0,0][0,0][0,0])^*[0,0][1,0]\\
&[1,0][0,1][0,0][0,1]([1,0][0,0][0,0][0,0])^*[0,0][1,0][1,0]\\
&[1,0][0,1][0,0][0,1]([1,0][0,0][0,0][0,0])^*[0,0][1,0][1,0][0,0]
\end{aligned}
$$

These correspond to, respectively, exponents of

$$
\begin{aligned}
&6/5\\
&(\sum_{1\le i\le n+2} X_{2i})/X_{2n+4}, \quad n\ge 0\\
&(\sum_{1\le i\le n+2} X_{2i+1})/X_{2n+5}, \quad n\ge 0\\
&(\sum_{1\le i\le n+1} X_{4i+1})/(X_{4n+4}+X_{4n+2}), \quad n\ge 0\\
&(1+\sum_{1\le i\le n+1} X_{4i+2})/(X_{4n+5}+X_{4n+3}), \quad n\ge 0\\
&(3+\sum_{1\le i\le n+1} X^{4i+3})/(X_{4n+6}+X_{4n+4}), \quad n\ge 0\\
&(6+\sum_{1\le i\le n+1} X^{4i+4})/(X_{4n+7}+X_{4n+5}), \quad n\ge 0.
\end{aligned}
$$

It now remains to check that these expressions are all less than γ and the last six tend to γ from below. We explain how to do this for the second expression; the others are similar.

First, by the standard theory of linear recurrences (see, *e.g.*, [7]), we know that

$$X_n = \alpha_1\beta_1^n + \alpha_2\beta_2^n + \alpha_3\beta_3^n,$$

where $\beta_1, \beta_2, \beta_3$ are the zeros of $X^3 - 2X^2 + X - 1$, and $\beta_1 \doteq 1.754877666246692760049 5$ is the unique real zero. By solving the appropriate linear system, we find that $\alpha_1 = (\beta_1^2 + 6\beta_1 + 3)/23$ and $\alpha_2 + \alpha_3 = (20 - \beta_1^2 - 6\beta_1)/23 = 0.277875581696887158856\cdots < 1$. Furthermore, $|\alpha_2| + |\alpha_3| = ((8\beta_1^2 - 20\beta_1 + 16)/23)^{1/2} < 1/2$. Since $|\beta_2| = |\beta_3|$ and the product of the zeros is equal to the constant term of the defining polynomial, which is 1, we get $|\beta_2|^2\beta_1 = 1$, which gives $|\beta_2| = \sqrt{1/\beta_1} = \beta_1 - 1$ by the defining equation. It follows that

$$|X_n - \alpha_1\beta_1^n| = |\alpha_2\beta_2^n + \alpha_3\beta_3^n| \leq (|\alpha_2| + |\alpha_3|)(\beta_1 - 1)^n < (\beta_1 - 1)^n.$$

Next, one can prove by induction (or using Walnut!) that

$$\sum_{1\leq i\leq n} X_{2i} = \frac{3X_{2n} - X_{2n+1} + 2X_{2n+2} - 6}{5}.$$

It follows that

$$\begin{aligned}\sum_{1\leq i\leq n} X_{2i} &\leq \frac{3X_{2n} - X_{2n+1} + 2X_{2n+2} - 6}{5} \\ &\leq \frac{3 - \beta_1 + 2\beta_1^2}{5}(\alpha_1\beta_1^{2n}) + 6(\beta_1 - 1)^{2n} - 6/5.\end{aligned}$$

Dividing by X_{2n}, we see that the quotient tends to γ from below, where $\gamma = \frac{3-\beta_1+2\beta_1^2}{5}$. Then $\gamma + 1$ is the desired critical exponent. □

5. Synchronization and balance

We now show that the functions $c_i : n \to |\mathbf{p}[0..n-1]|_i$ are synchronized in this numeration system, for $i \in \{0, 1, 2\}$. This means that there exists an automaton A_i taking n and x as inputs (in $P4$ representation) and accepting if $x = c_i(n)$, for $i = 0, 1, 2$. For more information about synchronization, see [19].

To do so, we "guess" the automata for the c_i using the Myhill-Nerode theorem and then verify our guesses using `Walnut`. Here is the code for verification, where `psynch0, psynch1, psynch2` are the guessed automata:

```
eval tmp0 "?msd_pisot4 An,x (($psynch0(n,x) & PI[n]=@0) => $psynch0(n+1,x+1))
   & (($psynch0(n,x) & PI[n]!=@0) => $psynch0(n+1,x))":
eval tmp1 "?msd_pisot4 An,x (($psynch1(n,x) & PI[n]=@1) => $psynch1(n+1,x+1))
   & (($psynch1(n,x) & PI[n]!=@1) => $psynch1(n+1,x))":
eval tmp2 "?msd_pisot4 An,x (($psynch2(n,x) & PI[n]=@2) => $psynch2(n+1,x+1))
   & (($psynch2(n,x) & PI[n]!=@2) => $psynch2(n+1,x))":
```

and all of these return `TRUE`.

Next, we create synchronized automata for $|\mathbf{p}[i..i+n-1]|_i$:

```
def pcount0 "?msd_pisot4 Ex,y $psynch0(i,x) & $psynch0(i+n,y) & y=x+z":
# 1687 states
def pcount1 "?msd_pisot4 Ex,y $psynch1(i,x) & $psynch1(i+n,y) & y=x+z":
# 2626 states
```

```
def pcount2 "?msd_pisot4 Ex,y $psynch2(i,x) & $psynch2(i+n,y) & y=x+z":
# 2773 states
```

We can now prove the following result.

Theorem 5.1. *The word* **p** *is* 2*-balanced.*

Proof. We use the following `Walnut` commands.

```
def twobalanced0 "?msd_pisot4 Ai,j,n,x,y ($pcount0(i,n,x) &
   $pcount0(j,n,y)) => (y<=x+2 & x<=y+2)":
def twobalanced1 "?msd_pisot4 Ai,j,n,x,y ($pcount1(i,n,x) &
   $pcount1(j,n,y)) => (y<=x+2 & x<=y+2)":
def twobalanced2 "?msd_pisot4 Ai,j,n,x,y ($pcount2(i,n,x) &
   $pcount2(j,n,y)) => (y<=x+2 & x<=y+2)":
```

`Walnut` returns `TRUE` for all of these. □

6. Abelian complexity

We can compute the abelian complexity of **p** with `Walnut` in much the same way that it was done for the Tribonacci word in [16], with some minor modifications. For each n and $i \in \{0,1,2\}$, we compute the vector $u_i(n) = \min_x |x|_i$, where the minimum is over all the length-n factors of **p**.

```
def min0 "?msd_pisot4 Ei $pcount0(i,n,x) & Aj,y $pcount0(j,n,y) => y>=x":
# 169 states

def min1 "?msd_pisot4 Ei $pcount1(i,n,x) & Aj,y $pcount1(j,n,y) => y>=x":
# 169 states

def min2 "?msd_pisot4 Ei $pcount2(i,n,x) & Aj,y $pcount2(j,n,y) => y>=x":
# 223 states
```

Once we have this, we can show that

$$\psi(\mathbf{p}[i..i+n-1]) - (u_0(n), u_1(n), u_2(n)) \in \{(0,0,1),(0,0,2),(0,1,0),(0,1,1),(0,1,2),\\ (0,2,0),(0,2,1),(1,0,0),(1,0,1),(1,0,2),(1,1,0),(1,1,1),(1,2,0),(2,0,0),(2,0,1),(2,1,0)\}$$

for $i \geq 0$ and $n \geq 1$, as follows:

```
def validtriples "?msd_pisot4 Ei,n,a,b,c $pcount0(i,n,a+x) & $min0(n,a) &
   $pcount1(i,n,b+y) & $min1(n,b) & $pcount2(i,n,c+z) & $min2(n,c)":
```

Next, we can show that

$$\{\psi(\mathbf{p}[i..i+n-1]) - (u_0(n), u_1(n), u_2(n)) \,:\, i \geq 0\}$$

is one of the following 18 possible sets:

$$S_1 = \{(0,0,1),(0,1,0),(1,0,0)\}$$
$$S_2 = \{(0,1,1),(1,0,1),(1,1,0)\}$$
$$S_3 = \{(0,1,1),(1,0,1),(1,1,0),(2,0,0)\}$$
$$S_4 = \{(0,0,2),(0,1,1),(1,0,1),(1,1,0)\}$$

$$
\begin{aligned}
S_5 &= \{(0,1,1),(0,2,0),(1,0,1),(1,1,0)\}\\
S_6 &= \{(0,0,2),(0,1,1),(1,0,1),(1,1,0),(2,0,0)\}\\
S_7 &= \{(0,1,2),(1,0,2),(1,1,1),(2,0,1),(2,1,0)\}\\
S_8 &= \{(0,2,1),(1,1,1),(1,2,0),(2,0,1),(2,1,0)\}\\
S_9 &= \{(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)\}\\
S_{10} &= \{(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0)\}\\
S_{11} &= \{(0,1,2),(0,2,1),(1,0,2),(1,1,1),(1,2,0)\}\\
S_{12} &= \{(0,1,2),(0,2,1),(1,1,1),(1,2,0),(2,0,1),(2,1,0)\}\\
S_{13} &= \{(0,1,2),(0,2,1),(1,0,2),(1,1,1),(2,0,1),(2,1,0)\}\\
S_{14} &= \{(0,1,2),(0,2,1),(1,0,2),(1,1,1),(1,2,0),(2,1,0)\}\\
S_{15} &= \{(0,1,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0)\}\\
S_{16} &= \{(0,2,1),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0)\}\\
S_{17} &= \{(0,1,2),(0,2,1),(1,0,2),(1,1,1),(1,2,0),(2,0,1)\}\\
S_{18} &= \{(0,1,2),(0,2,1),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0)\}.
\end{aligned}
$$

Theorem 6.1. *There is a P4-automaton of 144 states that, on input a P4 representation of n, computes the number of distinct length-n factors of $\mathbf{p}$, up to abelian equivalence.*

Proof. To create the automaton, use the following `Walnut` code:

```
def a001 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y)& $pcount2(i,n,z+1) & $min2(n,z)":
#6 states

def a002 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y) & $pcount2(i,n,z+2) & $min2(n,z)":
#125 states

def a010 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y+1) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
#6 states

def a011 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y+1) & $min1(n,y) & $pcount2(i,n,z+1) & $min2(n,z)":
# 132 states

def a012 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y+1) & $min1(n,y) & $pcount2(i,n,z+2) & $min2(n,z)":
# 129 states

def a020 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y+2) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
#126 states

def a021 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x) & $min0(n,x) &
   $pcount1(i,n,y+2) & $min1(n,y) & $pcount2(i,n,z+1) & $min2(n,z)":
#131 states
```

```
def a100 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+1) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
# 6 states

def a101 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+1) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y) & $pcount2(i,n,z+1) & $min2(n,z)":
# 132 states

def a102 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+1) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y) & $pcount2(i,n,z+2) & $min2(n,z)":
# 127 states

def a110 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+1) & $min0(n,x) &
   $pcount1(i,n,y+1) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
# 132 states

def a111 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+1) & $min0(n,x) &
   $pcount1(i,n,y+1) & $min1(n,y) & $pcount2(i,n,z+1) & $min2(n,z)":
# 131 states

def a120 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+1) & $min0(n,x) &
   $pcount1(i,n,y+2) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
# 134 states

def a200 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+2) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
# 110 states

def a201 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+2) & $min0(n,x) &
   $pcount1(i,n,y) & $min1(n,y) & $pcount2(i,n,z+1) & $min2(n,z)":
# 131 states

def a210 "?msd_pisot4 Ei,x,y,z $pcount0(i,n,x+2) & $min0(n,x) &
   $pcount1(i,n,y+1) & $min1(n,y) & $pcount2(i,n,z) & $min2(n,z)":
# 127 states

def num1 "?msd_pisot4 $a001(n) & ~$a002(n) & $a010(n) & ~$a011(n) & ~$a012(n) &
   ~$a020(n) & ~$a021(n) & $a100(n) & ~$a101(n) & ~$a102(n) & ~$a110(n) &
   ~$a111(n) & ~$a120(n) & ~$a200(n) & ~$a201(n) & ~$a210(n)":

def num2 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
   ~$a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & ~$a200(n) & ~$a201(n) & ~$a210(n)":

def num3 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
   ~$a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & $a200(n) & ~$a201(n) & ~$a210(n)":

def num4 "?msd_pisot4 ~$a001(n) & $a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
```

```
   ~$a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & ~$a200(n) & ~$a201(n) & ~$a210(n)":

def num5 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
   $a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & ~$a200(n) & ~$a201(n) & ~$a210(n)":

def num6 "?msd_pisot4 ~$a001(n) & $a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
   ~$a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & $a200(n) & ~$a201(n) & ~$a210(n)":

def num7 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & ~$a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & ~$a120(n) & ~$a200(n) & $a201(n) & $a210(n)":

def num8 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & ~$a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & ~$a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & $a201(n) & $a210(n)":

def num9 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
   $a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & $a200(n) & ~$a201(n) & ~$a210(n)":

def num10 "?msd_pisot4 ~$a001(n) & $a002(n) & ~$a010(n) & $a011(n) & ~$a012(n) &
   $a020(n) & ~$a021(n) & ~$a100(n) & $a101(n) & ~$a102(n) & $a110(n) &
   ~$a111(n) & ~$a120(n) & ~$a200(n) & ~$a201(n) & ~$a210(n)":

def num11 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & ~$a201(n) & ~$a210(n)":

def num12 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & ~$a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & $a201(n) & $a210(n)":

def num13 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & ~$a120(n) & ~$a200(n) & $a201(n) & $a210(n)":

def num14 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & ~$a201(n) & $a210(n)":

def num15 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & ~$a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & $a201(n) & $a210(n)":

def num16 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & ~$a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & $a201(n) & $a210(n)":
```

```
def num17 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & $a201(n) & ~$a210(n)":

def num18 "?msd_pisot4 ~$a001(n) & ~$a002(n) & ~$a010(n) & ~$a011(n) & $a012(n) &
   ~$a020(n) & $a021(n) & ~$a100(n) & ~$a101(n) & $a102(n) & ~$a110(n) &
   $a111(n) & $a120(n) & ~$a200(n) & $a201(n) & $a210(n)":

eval coverall "?msd_pisot4 An (n>=1) => ($num1(n)|$num2(n)|$num3(n)|$num4(n)|$num5(n)|
   $num6(n)|$num7(n)|$num8(n)|$num9(n)|$num10(n)|$num11(n)|$num12(n)|
   $num13(n)|$num14(n)|$num15(n)|$num16(n)|$num17(n)|$num18(n))":

combine pab num1 num2 num3 num4 num5 num6 num7 num8 num9 num10 num11 num12
   num13 num14 num15 num16 num17 num18:

morphism abc "0->1 1->3 2->3 3->4 4->4 5->4 6->5 7->5 8->5 9->5 10->5 11->5
   12->6 13->6 14->6 15->6 16->6 17->6 18->7":

image BC3 abc pab:
```

□

Corollary 6.2. *The abelian complexity of* $\mathbf{p}$*, for* $n \geq 1$*, lies in* $\{3, 4, 5, 6, 7\}$*. Each possibility occurs infinitely often.*

7. Palindromes

Theorem 7.1. *The only palindromes occurring in* $\mathbf{p}$ *are*

$$\{0, 1, 2, 121, 101, 010, 01210, 21012, 1012101\}.$$

Proof. It suffices to list all the factors of length ≤ 9, since any longer palindrome would have these as factors. We can be sure we have examined all of them, by Theorem 3.1. □

8. Bispecial factors

We say a factor w of an infinite word $\mathbf{x}$ is *right-special* (resp., *left-special*) if there exist two distinct letters a, b such that wa and wb (resp., aw and bw) are both factors of $\mathbf{x}$. A word w is *bispecial* if it is both right- and left-special.

We can determine the lengths of bispecial factors occurring in $\mathbf{p}$.

```
def pisotrightspec "?msd_pisot4 Ej $pisi(i,j,n) & PI[i+n]!=PI[j+n]":
def pisotleftspec "?msd_pisot4 Ej $pisi(i,j,n) & PI[i-1]!=PI[j-1]":
def pisotbispec "?msd_pisot4 $pisotrightspec(i,n) & $pisotleftspec(i,n)":
def pisotbispeclen "?msd_pisot4 Ei $pisotbispec(i,n)":
```

The result is displayed in Figure 6.

Theorem 8.1. *The word* $\mathbf{p}$ *has a bispecial factor of length* n *iff*

$$(n)_P \in \{1, 11, 110\} \cup (10)^+\{\epsilon, 0\} \cup (1000)^+\{0, 01, 011, 0110\}.$$

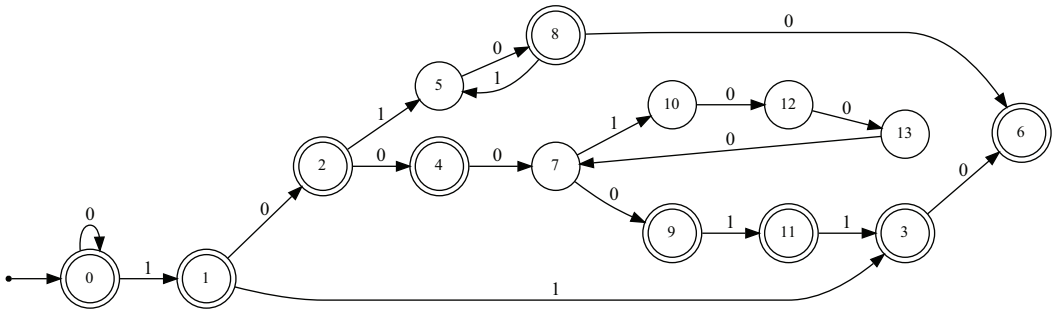

FIGURE 6. Automaton recognizing lengths of bispecial factors in **p**.

9. Letter density

In this section we obtain the letter densities of **p**. We know from [3] that the densities exist, and therefore it suffices to determine them on prefixes of the form $h^n(0)$.

Theorem 9.1.

- *The density of* 0 *is* $1/\beta_1^2 \doteq 0.324717957244746$*;*
- *The density of* 1 *is* $1/\beta_1^2 + 1/\beta_1^4 \doteq 0.430159709001946734$*;*
- *The density of* 2 *is* $1/\beta_1^3 + 1/\beta_1^5 \doteq 0.2451223337533$.

Proof. An easy induction gives $\psi(|h^n(0)|) = (X_{n-1}, X_{n-1} + X_{n-3}, X_{n-2} + X_{n-4})$, from which the desired result follows immediately. □

Remark 9.2. These densities were also given in [4] for a slightly different morphism with the same incidence matrix.

10. Recurrence and appearance

Let **x** be an infinite word. The recurrence function $R(n)$ is defined to be the smallest positive integer m such that every occurrence of a length-n factor x is followed by another occurrence of the same word at distance at most $R(n)$. We now compute it for the word **p**.

Theorem 10.1. *Define the sequences* (B_i) *and* (C_i) *as follows:*

$$\begin{aligned}
B_{2i+3} &= [1(01)^i10]_P \\
B_{2i+4} &= [1(01)^i100]_P \\
C_{4i+2} &= [10(0010)^i]_P \\
C_{4i+3} &= [10(0010)^i0]_P \\
C_{4i+4} &= [10(0010)^i00]_P \\
C_{4i+5} &= [10(0010)^i001]_P
\end{aligned}$$

for $i \geq 0$. Then

$$R(n) = \begin{cases} 5, & \text{if } n = 1; \\ 12, & \text{if } n = 2; \\ 16, & \text{if } n = 3; \\ 21, & \text{if } n = 4; \\ 28, & \text{if } n = 5; \\ X_{i+4}, & \text{if } B_i \leq n \leq C_{i+1} \text{ for } i \geq 3; \\ X_{i+4} + X_{i+2}, & \text{if } C_{i+1} < n < B_{i+1} \text{ for } i \geq 3. \end{cases}$$

Proof. We use the following `Walnut` code to compute R.

```
def poccur "?msd_pisot4 Ai Ej j>i & j<=i+x & $pisi(i,j,n)":
def precur "?msd_pisot4 $poccur(n,x) & ~$poccur(n,x-1)":
reg pix msd_pisot4 "0*10*":
def precB "?msd_pisot4 Ex $pix(x) & $precur(n,x) & ~$precur(n-1,x)":
def precC "?msd_pisot4 Ex $pix(x) & $precur(n,x) & ~$precur(n+1,x)":
```

□

Corollary 10.2. *We have $R(n) \leq \xi n$ for $\xi = (2\beta_1^2 - \beta_1 + 2) \doteq 6.40431358$.*

The appearance function $A(n)$ is defined to be the smallest positive integer m such that every occurrence of a length-n factor x begins at a position $\leq m$. We now compute it for the word $\mathbf{p}$.

Theorem 10.3. *Define the sequences (B_n) and (C_n) as in Theorem 10.1. We have*

$$A(n) = \begin{cases} 2, & \text{if } n = 1; \\ 4, & \text{if } n = 2; \\ 7, & \text{if } n = 3; \\ 11, & \text{if } n = 4; \\ 13, & \text{if } n = 5; \\ X_{2i+4} - 1, & \text{if } B_{2i+1} < n \leq C_{2i+2} \text{ for } i \geq 1; \\ C_{2i+4}, & \text{if } C_{2i+2} < n < B_{2i+2} \text{ for } i \geq 1; \\ X_{2i+5} - 1, & \text{if } B_{2i+2} \leq n \leq C_{2i+3} \text{ for } i \geq 1; \\ C_{2i+5}, & \text{if } C_{2i+3} < n \leq B_{2i+3} \text{ for } i \geq 1. \end{cases}$$

Corollary 10.4. *We have $A(n) \leq \zeta n$ for $\zeta = (2\beta_1^2 - 2\beta_1 + 1) \doteq 3.6494359$.*

11. Characterization of factors

In this section, we give a useful definition of $\mathbf{p}$ by forbidden factors. We thank Lucas Mol for his work on this result.

Theorem 11.1. *Every bi-infinite ternary cube-free word avoiding*

$$F = \{00, 11, 22, 20, 212, 0101, 02102, 121012, 01021010, 21021012102\}$$

has the same set of factors as $\mathbf{p}$.

Proof. First, we check that $\mathbf{p}$ is cube-free by Theorem 4.1 and contains no factor in F.

Let $\mathbf{w}$ be a bi-infinite ternary cube-free word avoiding F. Since $\mathbf{w}$ avoids $\{00, 11, 22, 20, 212\}$, the only factors of $\mathbf{w}$ of the form $0z0$ with $z \in \{1,2\}^*$ are 010, 01210, 0210. So $w \in \{01, 0121, 021\}^\omega$ and thus $w \in \{01, 21, 0\}^\omega$. So we write $\mathbf{w} = h(\mathbf{v})$ where h is the morphism $0 \to 01$, $1 \to 21$, $2 \to 0$.

Now it suffices to show that L contains $\mathbf{v}$ too. Since $\mathbf{w}$ is cube-free, its pre-image $\mathbf{v}$ is also cube-free.

To show that $\mathbf{v}$ avoids F, we consider every $f \in F$ and we show by contradiction that f is not a factor of $\mathbf{v}$.

(a) if $\mathbf{v}$ contains 00, then $h(00) = 0101 \in F$.
(b) if $\mathbf{v}$ contains 11, then $h(11) = 2121$ contains $212 \in F$.
(c) if $\mathbf{v}$ contains 22, then $h(22) = 00 \in F$.
(d) if $\mathbf{v}$ contains 20, then $h(20) = 001$ contains $00 \in F$.
(e) if $\mathbf{v}$ contains 212, then $\mathbf{v}$ contains 2121 by (c) and (d).
$h(2121) = 021021$ contains $02102 \in F$.
(f) if $\mathbf{v}$ contains 0101, then $h(0101) = 01210121$ contains $121012 \in F$.
(g) if $\mathbf{v}$ contains 02102, then $h(02102) = 01021010 \in F$.
(h) if $\mathbf{v}$ contains 121012, then $\mathbf{v}$ contains 1210121 by (c) and (d).
$h(1210121) = 210210121021$ contains $21021012102 \in F$.
(i) if $\mathbf{v}$ contains 01021010, then $\mathbf{v}$ contains 010210102 by (a) and (f).
$\mathbf{v}$ contains 0102101021 by (c) and (d).
$\mathbf{v}$ contains 01021010210 by (b) and (e).
$\mathbf{v}$ contains 010210102101 by (a) and (g).
$\mathbf{v}$ contains 1010210102101 by (a) and (d).
$\mathbf{v}$ contains 21010210102101 by (f) and (b).
$\mathbf{v}$ contains 210102101021012 by (b) and to avoid $(21010)^3$.
$\mathbf{v}$ contains 1210102101021012 by (c) and to avoid $(02101)^3$.
$h(1210102101021012) = 2102101210102102101021012102 = 2(102101210)^3$.
(j) if $\mathbf{v}$ contains 21021012102, then $\mathbf{v}$ contains 121021012102 by (g) and (c).
$\mathbf{v}$ contains 0121021012102 by (a) and (e).
$\mathbf{v}$ contains 10121021012102 by (a) and (d).
$\mathbf{v}$ contains 210121021012102 by (f) and (b).
$\mathbf{v}$ contains 0210121021012102 by (h) and (c).
$\mathbf{v}$ contains 10210121021012102 by (a) and (d).
$\mathbf{v}$ contains 102101210210121021 by (c) and (d).
$\mathbf{v}$ contains 1021012102101210210 by (b) and (e).
$\mathbf{v}$ contains 10210121021012102101 by (a) and (g).
$\mathbf{v}$ contains 102101210210121021010 by (b) and to avoid $(1021012)^3$.
$h(102101210210121021010) = 21010210121021010210121021010210121012101 = (210102101210)^3 1$.

□

Acknowledgements. We thank Arseny Shur for reminding us about the relevance of the paper [21]. We are very grateful to Lucas Mol for providing the proof of Theorem 11.1. Finally, we thank the referee for several helpful suggestions.

References

[1] J. Berstel, Axel Thue's Papers on Repetitions in Words: a Translation. Number 20 in Publications du Laboratoire de Combinatoire et d'Informatique Mathématique. Université du Québec à Montréal (February 1995).
[2] V. Bruyère and G. Hansel, Bertrand numeration systems and recognizability. *Theoret. Comput. Sci.* **181** (1997) 17–43.
[3] J. Cassaigne, S. Labbé and J. Leroy, A set of sequences of complexity $2n+1$. In S. Brlek *et al.*, editors, WORDS 2017, Vol. 10432 of *Lecture Notes in Computer Science*. Springer-Verlag (2017), pp. 144–156.
[4] J. Cassaigne, S. Labbé and J. Leroy, Almost everywhere balanced sequences of complexity $2n+1$. Preprint https://arxiv.org/abs/2102.10093 (2022).

[5] E.M. Coven and G.A. Hedlund, Sequences with minimal block growth. *Math. Systems Theory* **7** (1973) 138–153.
[6] J.D. Currie, P. Ochem, N. Rampersad and J. Shallit, Complement avoidance in binary words. Preprint https://arxiv.org/abs/2209.09598 (2022).
[7] G. Everest, A. van der Poorten, I. Shparlinski and T. Ward, Recurrence Sequences, Vol. 104 of *Mathematical Surveys and Monographs*. Amer. Math. Soc. (2003).
[8] A.S. Fraenkel, Systems of numeration. *Am. Math. Monthly* **92** (1985) 105–114.
[9] C. Frougny and B. Solomyak, On representation of integers in linear numeration systems. In M. Pollicott and K. Schmidt, editors, Ergodic Theory of $\mathbb{Z}^d$ Actions (Warwick, 1993–1994), Vol. 228 of *London Mathematical Society Lecture Note Series*. Cambridge University Press (1996), pp. 345–368.
[10] A. Glen and J. Justin, Episturmian words: a survey. *RAIRO Inform. Théor. App.* **43** (2009) 403–442.
[11] J. Karhumäki, On cube-free ω-words generated by binary morphisms. *Disc. Appl. Math.* **5** (1983) 279–297.
[12] F. Mignosi and G. Pirillo, Repetitions in the Fibonacci infinite word. *RAIRO Inform. Théor. App.* **26** (1992) 199–204.
[13] M. Morse and G.A. Hedlund, Symbolic dynamics II. Sturmian trajectories. *Am. J. Math.* **62** (1940) 1–42.
[14] H. Mousavi, Automatic theorem proving in `Walnut`. Preprint http://arxiv.org/abs/1603.06017 (2016).
[15] H. Mousavi, L. Schaeffer and J. Shallit, Decision algorithms for Fibonacci-automatic words, I: Basic results. *RAIRO Inform. Théor. App.* **50** (2016) 39–66.
[16] H. Mousavi and J. Shallit, Mechanical proofs of properties of the Tribonacci word. In F. Manea and D. Nowotka, editors, Proc. WORDS 2015, Vol. 9304 of *Lecture Notes in Computer Science*. Springer-Verlag (2015), pp. 1–21.
[17] G. Richomme, K. Saari and L.Q. Zamboni, Abelian complexity in minimal subshifts. *J. London Math. Soc.* **83** (2011) 79–95.
[18] J. Shallit, Abelian complexity and synchronization. *INTEGERS — Elect. J. Comb. Numb. Theory* **21** (2021), #A36 (electronic).
[19] J. Shallit, Synchronized sequences. In T. Lecroq and S. Puzynina, editors, WORDS 2021, Vol. 12847 of *Lecture Notes in Computer Science*. Springer-Verlag (2021), pp. 1–19.
[20] J. Shallit, The Logical Approach To Automatic Sequences: Exploring Combinatorics on Words with `Walnut`, Vol. 482 of *London Math. Soc. Lecture Note Series*. Cambridge University Press (2022).
[21] J. Shallit and A. Shur, Subword complexity and power avoidance. *Theoret. Comput. Sci.* **792** (2019) 96–116.
[22] A. Thue, Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Norske vid. Selsk. Skr. Mat. Nat. Kl.* **1** (1912) 1–67. Reprinted in *Selected Mathematical Papers of Axel Thue*, T. Nagell, editor, Universitetsforlaget, Oslo, 1977, pp. 413–478.