CYCLIC COMPLEXITY OF SOME INFINITE WORDS AND GENERALIZATIONS

Colin Krawchuk[1]
Department of Mathematics and Statistics, University of Winnipeg
Winnipeg, Canada
n.rampersad@uwinnipeg.ca

Narad Rampersad[2]
Department of Mathematics and Statistics, University of Winnipeg
Winnipeg, Canada

Received: 6/26/17, Accepted: 1/21/18, Published: 3/16/18

Abstract

Cassaigne et al. introduced the cyclic complexity function $c_x(n)$, which gives the number of cyclic conjugacy classes of length-n factors of a word x. We study the behavior of this function for the Fibonacci word f and the Thue–Morse word t. If $\varphi = (1+\sqrt{5})/2$, we show that $\limsup_{n\to\infty} c_f(n)/n \geq 2/\varphi^2$ and conjecture that equality holds. Similarly, we show that $\limsup_{n\to\infty} c_t(n)/n \geq 2$ and conjecture that equality holds. We also propose a generalization of the cyclic complexity function and suggest some directions for further investigation. Most results are obtained by computer proofs using Mousavi's Walnut software.

–In honour of Jeffrey Shallit's 60th birthday.

1. Introduction

One classical measure of the complexity of an infinite word x is given by the *factor complexity function*: that is, the function $p_x(n)$ which gives the number of distinct factors of x of length n. Furthermore, there is a well-known connection between the factor complexity and the periodicity/aperiodicity of x, namely, the Morse–Hedlund Theorem, which asserts that x is ultimately periodic if and only if $p_x(n)$ is bounded.

Other complexity functions have been introduced for infinite words: for instance, the *abelian complexity function* $a_x(n)$ counts the number of equivalence classes of length-n factors of x with respect to the *abelian equivalence relation*. This is the equivalence relation under which words u and v are equivalent if the letters of u can be rearranged to obtain v.

[1]Supported by an NSERC USRA.
[2]Supported by an NSERC Discovery Grant.

Cassaigne, Fici, Sciortino, and Zamboni [2] introduced the *cyclic complexity function* $c_x(n)$, which counts the number of equivalence classes of length-n factors of x with respect to the *conjugation relation*. Two words u and v are equivalent under this relation if it is possible to write $u = rs$ and $v = sr$ for some words r and s. We denote this by $u \sim v$. Cassaigne et al. proved the analogue of the Morse–Hedlund Theorem: i.e., that x is ultimately periodic if and only if $c_x(n)$ is bounded. Furthermore, they examined the limit inferior of the cyclic complexity function for several classes of aperiodic words, notably:

- $\liminf_{n\to\infty} c_x(n) = 2$ if x is Sturmian (but this does not characterize Sturmian words);
- $\liminf_{n\to\infty} c_t(n) = \infty$, where t is the Thue–Morse word.

In the first part of this paper, we try to provide some additional information on the behavior of the function $c_x(n)$ for the Fibonacci word f and the Thue–Morse word t. This behavior seems rather hard to analyze in general, and we have only some partial results. For instance, if $\varphi = (1 + \sqrt{5})/2$, we show that $\limsup_{n\to\infty} c_f(n)/n \geq 2/\varphi^2$ and conjecture that equality holds. Similarly, we show that $\limsup_{n\to\infty} c_t(n)/n \geq 2$ and conjecture that equality holds.

In the second part of the paper, we propose a generalization of the cyclic complexity function and suggest some directions for further investigation.

Our main tool here is the software tool Walnut, which can be used to obtain automated proofs of certain types of results concerning automatic sequences. To understand most of this paper, the reader should have a basic understanding of what Walnut does. For background on this, see [6] or [5, 7]. The software itself can be downloaded at

`https://cs.uwaterloo.ca/~shallit/Papers/Walnut.zip`

and the specific Walnut commands used in this paper can be downloaded at

`http://ion.uwinnipeg.ca/~nrampers/cyclic_walnut.zip`

2. Cyclic Complexity of the Thue–Morse and Fibonacci Words

Let $t = 0110100110010110\cdots$ be the *Thue–Morse word*; i.e., the word generated by iterating the morphism $0 \to 01$, $1 \to 10$. Since t is an aperiodic *automatic sequence*, the factor complexity function $p_t(n)$ is bounded above and below by linear functions. In particular, we have

$$\limsup_{n\to\infty} \frac{p_t(n)}{n} = \frac{10}{3} \qquad \text{and} \qquad \liminf_{n\to\infty} \frac{p_t(n)}{n} = 3.$$

Cassaigne et al. [2] proved that $\liminf_{n\to\infty} c_t(n)$ is unbounded. We now give some additional information on the growth of $c_t(n)$.

Proposition 1. *Let $k \geq 3$. For $n = 2^k$ we have $c_t(n) = 2n - 4$.*

Proof. The proof makes use of Walnut. We use a number of logical predicates. The first,

$$\begin{aligned}\text{ConjTM} :=& \exists m\ (m \leq n) \wedge \\ & (\forall k\ k < m \Rightarrow t[i+k] = t[j+n-m+k]) \wedge \\ & (\forall l\ l < n-m \Rightarrow t[j+l] = t[i+m+l]),\end{aligned}$$

says that the factors $t[i..i+n-1]$ and $t[j..j+n-1]$ are conjugates of each other. The Walnut notation for this is

```
def Conj_tm "Em (m<=n) & (Ak k<m => T[i+k]=T[j+n-m+k]) &
(Al l<n-m => T[j+l]=T[i+m+l])";
```

The output that Walnut generates when this command is entered is an automaton with 83 states (which we will not reproduce here).

Next, we define

$$\text{NewConjClassTM} := \forall j\ j < i \Rightarrow \sim\text{ConjTM}(i, j, n),$$

which says that no word in the conjugacy class of $t[i..i+n-1]$ occurs prior to position i. The Walnut notation for this is

```
def NewConjClass_tm "Aj j<i => ~$Conj_tm(i,j,n)";
```

The Walnut output corresponding to this command is a 50 state automaton; however, if we restrict n in this predicate to powers of two, Walnut produces the automaton in Figure 1.

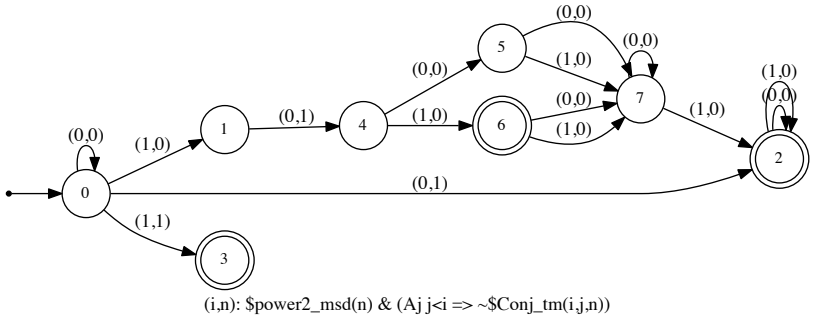


Figure 1: Automaton for new length-2^k conjugacy classes in t

The binary representation of $n = 2^k$ is the string 10^k, so to obtain the desired answer, we must compute the number of paths in this automaton whose second

component spells out the string 10^k. That is, we must compute the number of paths of length k starting from states 2, 3, or 4 and ending in states 2, 3, or 6. The adjacency matrix of the graph of this automaton is

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

which has minimal polynomial $x^3(x-2)(x-1)^2$. For $k \geq 3$ the number of such paths is therefore of the form $A \cdot 2^k + Bk + C$, and it is a routine calculation to verify that this number is in fact $2 \cdot 2^k - 4 = 2n - 4$, as required. □

Proposition 2. *Let $k \geq 2$. For $n = 4^k + 2$ we have $c_t(n) = (4/3)n - 4$.*

Proof. We again use the NewConjClassTM predicate defined in the proof of Proposition 1. Now we restrict n to have the form $n = 4^k + 2$. The resulting automaton computed by Walnut is given in Figure 2.

If $n = 4^k + 2$ then the binary representation of n is the string $1(00)^{k-1}10$. Thus we need to compute the number of paths of length $2k$ in this automaton that start in states 2, 3, 5, 6, or 13 and end in state 18. We work with the square of the adjacency matrix, whose minimal polynomial is $x^2(x-4)(x-1)^2$. It follows that for $k \geq 2$ the desired number of paths is of the form $A \cdot 4^k + Bk + C$. Plugging in initial values and solving the resulting linear system gives $A = 4/3$, $B = 0$, and $C = -4/3$, which gives the desired result: i.e., if $n = 4^k + 2$ then $c_t(n) = (4/3)n - 4$. □

Based on Propositions 1 and 2 we make the following conjecture:

Conjecture 3.

$$\limsup_{n\to\infty} \frac{c_t(n)}{n} = 2 \qquad \text{and} \qquad \liminf_{n\to\infty} \frac{c_t(n)}{n} = \frac{4}{3}.$$

Empirical results (plotted in Figure 3) provide further evidence for the conjecture. The values for $c_t(n)$ plotted in the figure were obtained by enumerating all factors of t of length n and then counting only the cyclically distinct equivalence classes among these words.

Now we examine the cyclic complexity of the *Fibonacci word*

$$f = 0100101001001010010100 10 \cdots;$$

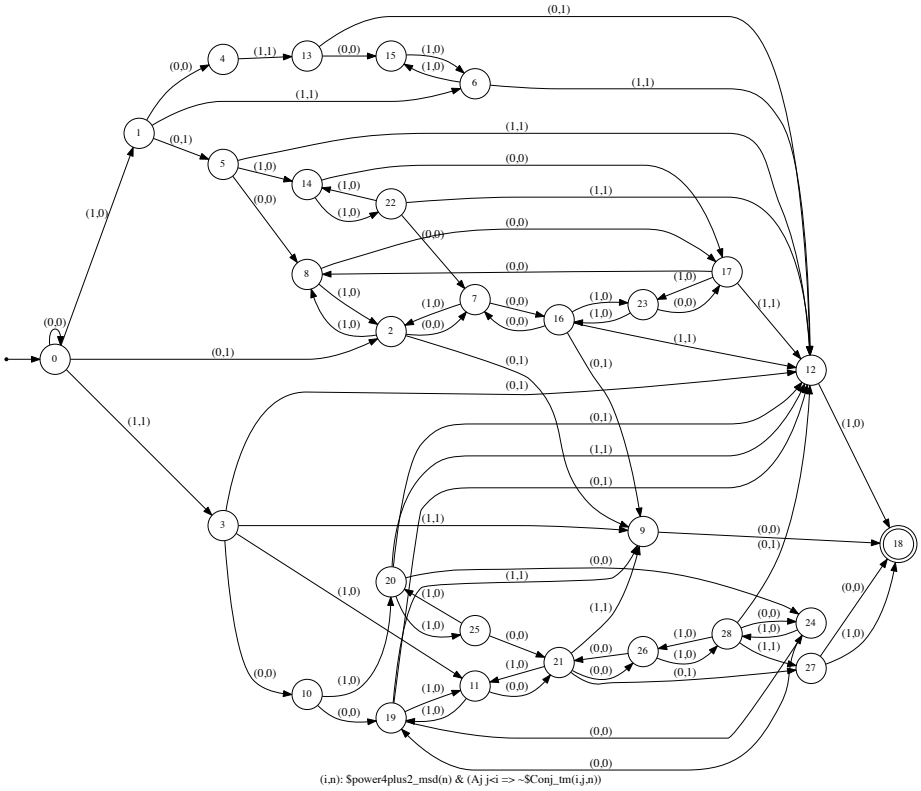


Figure 2: Automaton for new length-$(4^k + 2)$ conjugacy classes in t

i.e., the word generated by iterating the morphism $0 \to 01$, $1 \to 0$. Let F_k denote the k-th Fibonacci number. Recall that since f is a *Sturmian word*, the factor complexity of f is $p_f(n) = n + 1$ for all n. Cassaigne et al. determined that for $n = F_k$, we have $c_f(n) = 2$. We would like to obtain some information that would suggest a possible value for $\limsup_{n\to\infty} c_f(n)/n$.

Proposition 4. *Let $k \geq 7$. For $n = F_k + 1$ we have*

$$c_f(n) = \begin{cases} 2F_{k-2} - 1 & \text{if } k \text{ is odd,} \\ 2F_{k-2} & \text{if } k \text{ is even.} \end{cases}$$

Proof. We would like to define a predicate ConjFib in exactly the same way as the

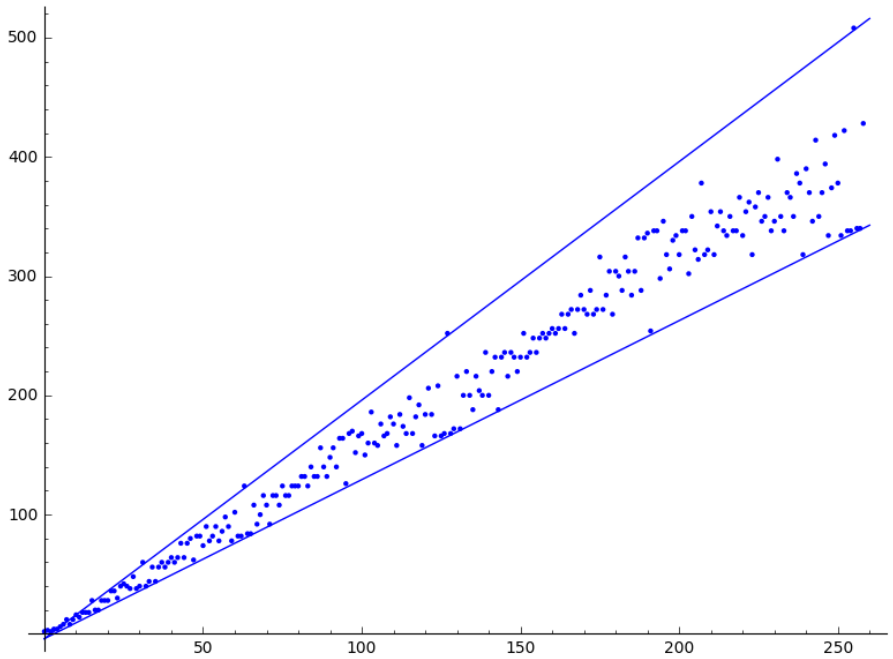


Figure 3: Plot of $c_t(n)$ along with lines of slope 2 and 4/3

predicate ConjTM was defined in the proof of Proposition 1, i.e.,

$$\begin{aligned}\text{ConjFib} :=& \exists m\ (m \le n) \wedge \\ & (\forall k\ k < m \Rightarrow f[i+k] = f[j+n-m+k]) \wedge \\ & (\forall l\ l < n-m \Rightarrow f[j+l] = f[i+m+l]).\end{aligned}$$

However, when given the predicate defined in this way as input, Walnut cannot compute the desired automaton in the available computer memory. The same issue was encountered by Du et al. [5, bottom of p. 157]; they were able to resolve the problem by rewriting the predicate. We therefore rewrite the predicate using the

same trick:

$$\begin{aligned}\text{ConjFib} :=& i<j \wedge \exists m\ (m \le n) \wedge \\ & (\exists d\ d+i+m=j+n \wedge (\forall u\ (i \le u \wedge u < i+m) \Rightarrow f[u]=f[u+d])) \wedge \\ & (i+m \ge j \Rightarrow \exists e\ e+j=i+m \wedge \\ & (\forall v\ (j \le v \wedge v+m < j+n) \Rightarrow f[v]=f[v+e])) \wedge \\ & (i+m<j \Rightarrow \exists f\ f+i+m=j \wedge \\ & (\forall w\ (i+m \le w \wedge w < i+n) \Rightarrow f[w]=f[w+f])).\end{aligned}$$

For technical reasons we try to avoid using subtraction whenever possible in the predicate. The Walnut command is:

```
def Conj_fib "?msd_fib i<j & Em (m<=n) &
(Ed d+i+m=j+n & (Au (i<=u & u<i+m) => F[u]=F[u+d])) &
(i+m>=j => Ee e+j=i+m & (Av (j<=v & v+m<j+n) => F[v]=F[v+e])) &
(i+m<j => Ef f+i+m=j & (Aw (i+m<=w & w<i+n) => F[w]=F[w+f]))";
```

Here the `msd_fib` command indicates that Walnut is doing arithmetic using the Zeckendorf expansion of natural numbers. The resulting Walnut output is an automaton with 143 states.

Next we define the NewConjClassFib predicate in the same way that the NewConjClassTM predicate is defined in the proof of Proposition 1. If we restrict n in this latter predicate to $n = F_k + 1$, Walnut produces the automaton in Figure 4.

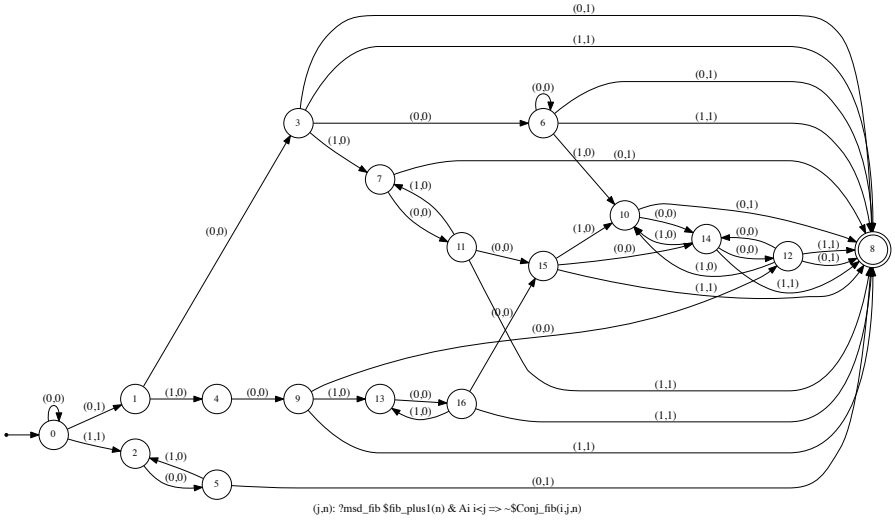


Figure 4: Automaton for new length-$(F_k + 1)$ conjugacy classes in f

The Zeckendorf representation of $n = F_k + 1$ is the string $10^{k-3}1$, so to obtain the desired answer, we must compute the number of paths of length $k-2$ in this automaton starting from states 1 or 2 and ending in state 8. The adjacency matrix of the graph of this automaton is

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0
\end{pmatrix},$$

which has minimal polynomial $x^5(x+1)(x-1)^2(x^2-x-1)$. It follows that for $k \geq 7$ the number of paths of length $k-2$ starting from states 1 or 2 and ending in state 8 is of the form

$$A\left(\frac{1+\sqrt{5}}{2}\right)^{k-2} + B\left(\frac{1-\sqrt{5}}{2}\right)^{k-2} + C(-1)^{k-2} + Dk + E.$$

Plugging in initial values and solving the resulting linear system (using SAGE) gives $A = 2/\sqrt{5}$, $B = -2/\sqrt{5}$, $C = 1/2$, $D = 0$, and $E = -1/2$. From the well-known formula

$$F_{k-2} = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{k-2} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{k-2},$$

we obtain the claimed result. □

Writing $\varphi = (1+\sqrt{5})/2$, we have the asymptotic result $c_f(F_k+1) \sim (2/\sqrt{5})\varphi^{k-2}$. Since $F_k \sim \varphi^k/\sqrt{5}$, we therefore propose the following (see Figure 5):

Conjecture 5.

$$\limsup_{n\to\infty} \frac{c_f(n)}{n} = \frac{2}{\varphi^2} \approx 0.7369\cdots$$

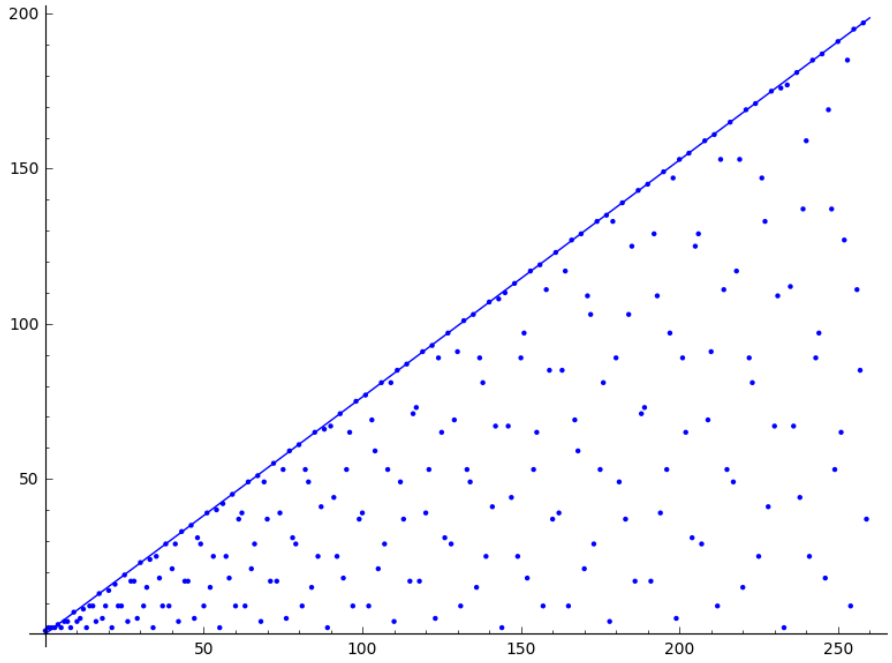


Figure 5: Plot of $c_f(n)$ along with line of slope 0.7369

In addition to the conjectures given above, one can pose the following questions/problems:

> Explore further the cyclic complexity of Sturmian words. In particular, is the quantity $\limsup_{n\to\infty} c_x(n)/n$ maximal for the Fibonacci word among all Sturmian words x? Also, is it possible to have $c_x(n) = o(n)$ for an aperiodic word x? More generally, is it possible to have $c_x(n) = o(p_x(n))$?

3. Generalized Cyclic Complexity

Charlier, Puzynina, and Zamboni [3] introduced a very general framework for studying certain types of complexity functions. Let S_n denote the symmetric group on n symbols (here we will use $\{0, 1, \cdots, n-1\}$). For any subgroup $G \subseteq S_n$, we can define an equivalence relation on words of length n as follows: If $u = u_0u_1 \cdots u_{n-1}$ and $v = v_0v_1 \cdots v_{n-1}$, where the u_i and v_i are single letters, then $u \sim v$ if there

exists $\sigma \in G$ such that

$$u_0u_1 \cdots u_{n-1} = v_{\sigma(0)}v_{\sigma(1)} \cdots v_{\sigma(n-1)}.$$

Now let $(G_n)_{n\geq 1}$ be an infinite sequence of subgroups $G_n \subseteq S_n$. For any infinite word x we can define a complexity function f_x, where $f_x(n)$ is equal to the number of distinct equivalence classes of factors of x of length n under the action of G_n. For instance, if $G_n = \{\mathrm{id}_n\}$ for all n, then f_x is the usual factor complexity function p_x. If $G_n = S_n$ for all n, then f_x is the abelian complexity function a_x.

Recall that by the Morse–Hedlund Theorem, a word x is aperiodic if and only if p_x is unbounded. On the other hand this result does not hold with a_x in place of p_x, since, for example, any Sturmian word x is aperiodic but has $a_x(n) = 2$ for all n. The natural question, then, is for which complexity functions f_x as defined above does an analogue of the Morse–Hedlund Theorem hold? Charlier et al. partially answer this question by proving that

$$f_x(n) \geq \varepsilon(G_n) + 1,$$

where $\varepsilon(G_n)$ is the number of distinct G_n-orbits of $\{0, 1, \cdots, n-1\}$.

The cyclic complexity function also fits into this framework in a way that suggests the following generalization of cyclic complexity. For $a, d \in \mathbb{Z}_n$, let $\sigma_{a,d} \in S_n$ be defined by

$$\sigma_{a,d}(j) = a + dj \pmod{n} \text{ for } j \in \{0, 1, \ldots, n-1\}.$$

If

$$G_n = \{\sigma_{a,1} : a \in \mathbb{Z}_n\}$$

for all n, then f_x is the cyclic complexity function c_x. If

$$G_n = \{\sigma_{a,d} : a, d \in \mathbb{Z}_n, (d, n) = 1\}$$

for all n, then we call f_x the *generalized cyclic complexity function* and denote it by gc_x. From now on we use the symbol $\sim$ to denote cyclic equivalence of words and the the symbol $\sim_{gc}$ to denote the generalized cyclic equivalence of words. This latter equivalence was introduced by Djokovic et al. [4] (and they called the equivalence classes under this relation "charm bracelets").

For example, the cyclic equivalence class of 00201 is the set of words

$$00201 \quad 02010 \quad 20100 \quad 01002 \quad 10020$$

and the generalized cyclic equivalence class of 00201 is the set of words

$$\begin{array}{lllll} 00201 & 02010 & 20100 & 01002 & 10020 \\ 02100 & 21000 & 10002 & 00021 & 00210 \\ 00012 & 00120 & 01200 & 12000 & 20001 \\ 01020 & 10200 & 02001 & 20010 & 00102. \end{array}$$

Recall that Cassaigne et al. proved that x is aperiodic if and only if $c_x(n)$ is unbounded. One can then ask whether this holds with gc_x in place of c_x. We believe that this is true, but have not been able to prove this.

Before proceeding, let us introduce one additional complexity function: the function $cr_x(n)$ (for *cyclic/reversal complexity*) defined as above by taking

$$G_n = \{\sigma_{a,1} : a \in \mathbb{Z}_n\} \cup \{\sigma_{a,n-1} : a \in \mathbb{Z}_n\}.$$

Note that under this relation, two words u and v are equivalent if either $u \sim v$ or $u \sim \tilde{v}$, where $\tilde{v}$ denotes the reversal of the word v. Observe that

$$a_x(n) \leq gc_x(n) \leq cr_x(n) \leq c_x(n) \leq p_x(n).$$

Let f be the Fibonacci word. Some initial values for $c_f(n)$, $cr_f(n)$, and $gc_f(n)$ are given below:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$c_f(n)$	2	2	2	3	2	4	4	2	7	4	5	8	2	9	9	4	13
$cr_f(n)$	2	2	2	3	2	3	4	2	5	3	4	6	2	6	6	3	8
$gc_f(n)$	2	2	2	3	2	3	2	2	5	3	4	6	2	6	6	3	8

Note that cr_f and gc_f are equal for these initial values of n, except $n = 7$. Indeed this appears to be the case in general; computer calculations suggest the following.

Conjecture 6. Let f be the Fibonacci word. Then $cr_f(n) = gc_f(n)$ for all $n \geq 1$, except $n = 7$.

Let t be the Thue–Morse word. Some initial values for $c_t(n)$, $cr_t(n)$, and $gc_t(n)$ are given below:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$c_t(n)$	2	3	2	4	4	6	8	12	8	12	16	14	18	18	18	28	20
$cr_t(n)$	2	3	2	4	4	5	6	7	6	8	10	8	12	10	10	16	12
$gc_t(n)$	2	3	2	4	2	5	4	7	4	8	10	8	12	10	6	16	6

Computer calculations suggest the following.

Conjecture 7. Let t be the Thue–Morse word. Then $cr_t(n) = gc_t(n)$ for all $n \geq 1$, except when n equals 5 or is of the form $n = 2^k \pm 1$ for $k \geq 3$.

If the set of factors of x is closed under reversal, we can obtain a relationship between the cyclic complexity and the cyclic/reversal complexity in a word x by calculating the number of cyclic equivalence classes of palindrome pairs of length n that appear in x. A word w is a *palindrome pair* if $w = uv$, where each of u and v is a palindrome (possibly empty). Note that every cyclic shift of a palindrome pair is again a palindrome pair. For more on palindrome pairs, see [1].

Proposition 8. *A word w is a palindrome pair if and only if $w \sim \tilde{w}$.*

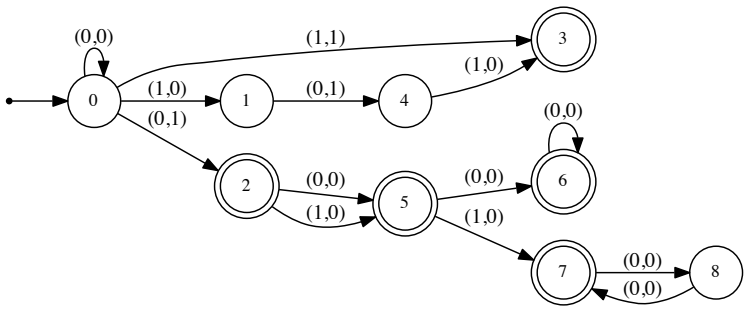
Proof. ($\Rightarrow$) Write $w = uv$ where $u = \tilde{u}$ and $v = \tilde{v}$. Then $\tilde{w} = \tilde{v}\tilde{u} = vu$, so $w \sim \tilde{w}$.

($\Leftarrow$) Write $w = uv$ and $\tilde{w} = vu$. However $\tilde{w} = \tilde{v}\tilde{u}$, so we get $v = \tilde{v}$ and $u = \tilde{u}$. Thus w is a palindrome pair. □

Observe that if $w \not\sim \tilde{w}$, then the cyclic equivalence classes of w and $\tilde{w}$ merge into one cyclic/reversal equivalence class, and if $w \sim \tilde{w}$, then w and $\tilde{w}$ already belong to the same cyclic/reversal equivalence class. For example, for the Thue–Morse word, one can use this observation to obtain certain values of $cr_t(n)$.

Proposition 9. *Let $k \geq 3$. For $n = 2^k$, the number of cyclic equivalence classes of palindrome pairs among all length-n factors of t is either 2 if k is odd or 4 if k is even.*

Proof. The proof is again done with Walnut. We compute the automaton in Figure 6 and note that for $k \geq 3$ the number of paths from state 2 to a final state is either 2 or 4 accordingly as k is odd or even. □



(i,n): $power2_msd(n) & (Ek $palindrome_tm(i,k)& $palindrome_tm(k,i+n)) & (Aj j<i => ~$Conj_tm(i,j,n))

Figure 6: Cyclic equivalence classes of length-2^k palindrome pairs in t

Proposition 10. *Let $k \geq 3$. For $n = 2^k$ we have*

$$cr_t(n) = \begin{cases} n-1 & \text{if } k \text{ is odd;} \\ n & \text{if } k \text{ is even.} \end{cases}$$

Proof. From Proposition 1 we have $c_t(n) = 2n - 4$. By Proposition 9 and the discussion preceeding it, we have

$$cr_t(n) = (c_t(n) - 2)/2 + 2 = (2n - 4 - 2)/2 + 2 = n - 1$$

if k is odd, and

$$cr_t(n) = (c_t(n) - 4)/2 + 4 = (2n - 4 - 4)/2 + 4 = n$$

if k is even. □

For our two examples, the Fibonacci word and the Thue–Morse word, it appears that the generalized cyclic complexity classes are almost always exactly equal to the cyclic/reversal equivalence classes. One wonders to what extent this is true for certain classes of infinite words: i.e., automatic, Sturmian, morphic, etc.

References

[1] A. Borchert, N. Rampersad, Words with many palindrome pair factors, *Electron. J. Combinatorics* **22** (2015), Paper #P4.23.

[2] J. Cassaigne, G. Fici, M. Sciortino, L. Q. Zamboni, Cyclic complexity of words, *J. Combin. Theory Ser. A* **145** (2017), 36–56.

[3] E. Charlier, S. Puzynina, L. Q. Zamboni, On a group theoretic generalization of the Morse–Hedlund theorem, *Proc. Amer. Math. Soc.* **145** (2017), 3381–3394.

[4] D. Dokovic, I. Kotsireas, D. Recoskie, J. Sawada, Charm bracelets and their application fo the construction of periodic Golay pairs, *Discrete Appl. Math.* **188** (2015), 32–40.

[5] C. F. Du, H. Mousavi, E. Rowland, L. Schaeffer, J. Shallit, Decision algorithms for Fibonacci-automatic words, II: Related sequences and avoidability, *Theoret. Comput. Sci.* **657** (2017) 146–162.

[6] H. Mousavi, Automatic theorem proving in Walnut. `https://arxiv.org/abs/1603.06017`

[7] H. Mousavi, L. Schaeffer, J. Shallit, Decision Algorithms for Fibonacci-Automatic Words, I: Basic Results, *RAIRO Theor. Inform. Appl.* **50** (2016), 39–66.