*Fifth Colloquium on Mathematics and Computer Science* DMTCS proc. **AI**, 2008, 351–362

# *Hopcroft's automaton minimization algorithm and Sturmian words*

Jean Berstel[1] and Luc Boasson[2] and Olivier Carton[2]

[1] *Institut Gaspard-Monge (IGM), Université Paris-Est and CNRS, F-77454 Marne-la-Vallée Cedex 2*
[2] *Laboratoire d'informatique algorithmique: fondements et applications (LIAFA), Université Denis-Diderot (Paris VII) and CNRS, F-75251 Paris Cedex 05*

This paper is concerned with the analysis of the worst case behavior of Hopcroft's algorithm for minimizing deterministic finite state automata. We extend a result of Castiglione, Restivo and Sciortino. They show that Hopcroft's algorithm has a worst case behavior for the automata recognizing Fibonacci words.

We prove that the same holds for all standard Sturmian words having an ultimately periodic directive sequence (the directive sequence for Fibonacci words is $(1, 1, \ldots)$).

**Keywords:** finite automata, minimization algorithm, complexity, Sturmian words

## 1 Introduction

An algorithm for the minimization of deterministic finite state automata that runs in time $O(n \log n)$ on automata with $n$ states was given by Hopcroft (1971). It is, up to now, the most efficient algorithm known in the general case.

We address here the problem of showing that the running time $O(n \log n)$ for Hopcroft algorithms is tight. This algorithm has a degree of freedom because, in each step of its main loop, it allows a free choice of a set of states to be processed. Berstel and Carton (2004) introduced a family of finite automata based on de Bruijn words, and they showed that there exist some "unlucky" sequence of choices that slows down the computation to achieve lower bound $\Omega(n \log n)$. In the paper Castiglione et al. (2007), Castiglione, Restivo and Sciortino replace the de Bruijn words by Fibonacci words. They show that, for this word, there is no more choice in Hopcroft's algorithm, and that the unique execution of Hopcroft's algorithm runs in time $\Omega(n \log n)$. The computation is carried out explicitly, using connections between Fibonacci numbers and Lucas numbers.

The uniqueness of the execution of Hopcroft's algorithm comes from the fact that automata for Sturmian words are what we have called "slow automata" in another context Berstel et al. (2007). In this paper, we present a generalization of the result of Castiglione et al. (2007) to any sequence of Sturmian words that is constructed with an eventually periodic directive sequence. These words are well-known and have special properties (see Berstel and Séébold (2002); Allouche and Shallit (2003); Pytheas Fogg

(2002)). For instance, the corresponding infinite words are precisely those that are fixed points of non-trivial morphisms. The case of Fibonacci words corresponds the directive sequence $(1, 1, \ldots)$. We show that, for this family of words, the execution time of Hopcroft's algorithm is also $\Omega(n \log n)$.

The proof is along the same lines as Castiglione et al. (2007), but it involves new constructions and some rather delicate analysis of the behavior of a particular system of equations.

The steps that lead to the equations describing the running time of Hopcroft's algorithm are similar to those in Castiglione et al. (2007), but we get a system of equations instead of a single one. The solution of the system requires some additional work. It is interesting to observe that we are led quite naturally to consider a cyclic version of continuants, as they are usually introduced for continued fractions (see Graham et al. (1994)).

**Outline** The paper is organized as follows. After some definitions, the first section sketches Hopcroft's automata minimization algorithm in the case we are interested in, namely for alphabets formed of a single letter. We then derive the equations for the generating function of the the running time of the algorithm. We describe a closed form for these equations, using a cyclic version of the continuant polynomials. The last section is concerned with the solution of these systems of equations.

## 2 Definitions and notation

In this section, we recall some definitions concerning finite Sturmian words, and fix conventions concerning circular occurrences of factors.

**Directive sequence** A *directive sequence* is a sequence $d = (d_1, d_2, d_3, \ldots)$ of positive integers. If $d$ is eventually periodic and has period $k$, that is if $d_{n+k} = d_n$ for $n > \ell$, then one writes also $d = (d_1 \ldots, d_\ell, \overline{d_{\ell+1}, \ldots, d_{\ell+k}})$. In particular, if $d$ is purely periodic and has period $k$, then one writes $d = (\overline{d_1, \ldots, d_k})$.

**Standard words** A directive sequence $d$ generates a sequence $(s_n)_{n\geq 0}$ of words, called the *standard words* generated by $d$, as follows

$$s_0 = 1\,, \quad s_1 = 0\,, \qquad s_{n+1} = s_n^{d_n} s_{n-1} \quad (n \geq 1)\,.$$

Thus in particular $s_2 = 0^{d_1}1$ and $s_3 = (0^{d_1}1)^{d_2}0$. We observe that in the literature on Sturmian words, the first term of a directive sequence is frequently supposed to be only non-negative. We exclude the case $d_1 = 0$ for convenience. It amounts merely to exchange symbols 0 and 1 in the sequence generated by $d$.

**Example 1** (Fibonacci) For $d = (1, 1, \ldots) = (\overline{1})$, one gets $s_{n+1} = s_n s_{n-1}$ for $n \geq 1$, and the standard words generated by $d$ are the Fibonacci words $1, 0, 01, 010, 01001, \ldots$

For $d = (\overline{2, 3})$, one gets $s_{n+1} = s_n^2 s_{n-1}$ if $n$ is odd, and $s_{n+1} = s_n^3 s_{n-1}$ if $n$ is even. The standard words generated by $d$ are the words 1, 0, 001, 0010010010, 00100100100010010010001,...

**Shift** The *shift* $\tau(d)$ of a directive sequence $d = (d_1, d_2, d_3, \ldots)$ is defined by

$$\tau(d) = \begin{cases} (d_1 - 1, d_2, d_3, \ldots) & \text{if } d_1 > 1 \\ (d_2, d_3, \ldots) & \text{otherwise}\,. \end{cases}$$

**Circular factors** Two words $x$ and $y$ are *conjugate* if there exist words $u, v$ such that $x = uv$ and $y = vu$. We write $x \sim y$ when $x$ and $y$ are conjugate. It is useful to see a class of the conjugacy relation $\sim$ as a single word as drawn on a circle, with no distinguished origin. A word $u$ is a *circular factor* of a word $w$ if it is a factor of some conjugate of $w$. This is equivalent for $u$ to be a prefix of some conjugate of $w$. It is also equivalent to the conditions that $|u| \leq |w|$ and $u$ is a factor of $ww$. The set of circular factors of a word is denoted by $CF(w)$. The *number of circular occurrences* of $u$ in $w$ is denoted by $|w|_u$ and is the number of factorizations $ww = pus$ with $|p| < |w|$ and $|u| \leq |w|$. This is precisely the number of occurrences of $u$ in $w$ viewed as drawn on a circle. Given a word $w$ over the alphabet $\{0, 1\}$, a word $u$ is a *special (circular) factor* of $w$ if $u0$ and $u1$ are both (circular) factors of $w$. We define the *weight* of $w$ by

$$\|w\| = \sum_{u \in CF(w)} \min(|w|_{u0}, |w|_{u1}) \,.$$

The only circular factors which contribute to $\|w\|$ are circular special factors. Two conjugate words have the same weight.

# 3 Hopcroft's algorithm

Hopcroft (1971) has given an algorithm that computes the minimal automaton of a given deterministic automaton. The running time of the algorithm is $O(|A| \times n \log n)$ where $|A|$ is the cardinality of the alphabet and $n$ is the number of states of the given automaton. The algorithm has been described and re-described several times (Gries (1973), Aho et al. (1974), Beauquier et al. (1992), Blum (1996), Knuutila (2001)).

## 3.1 Outline

The algorithm is outlined in the function HOPCROFTMINIMIZATION given below, and is explained then in some more detail.

It is convenient to use the shorthand $T^c = Q \setminus T$ when $T$ is a subset of the set $Q$ of states. We denote by $\min(B, C)$ the set of smaller size of the two sets $B$ and $C$, and any one of them if they have the same size.

Given a deterministic automaton $\mathcal{A}$, Hopcroft's algorithm computes the coarsest congruence which saturates the set $F$ of final states. It starts from the partition $\{F, F^c\}$ which obviously saturates $F$ and refines it until it gets a congruence. These refinements of the partition are always obtained by splitting some class into two classes. Let $B$ and $C$ be two sets of states and let $a$ be a letter. We say that the pair $(C, a)$ *splits* the set $B$ if both sets $(B \cdot a) \cap C$ and $(B \cdot a) \cap C^c$ are nonempty.

The main ingredient in the analysis of the running time of the algorithm is that the splitting of all classes of the current partition according to a pair $(C, a)$ takes a time proportional to the size of $C$. Therefore, the global running time of the algorithm is proportional to the sum of the sizes of the classes processed in the main loop. Note that a pair which is added to the waiting set $\mathcal{W}$ is not necessarily processed later because it can be split by the processing of another pair before it is considered.

It should be noted that the algorithm is not really deterministic because it has not been specified which pair $(C, a)$ is taken from $\mathcal{W}$ to be processed at each iteration of the main loop. This means that for a given automaton, there are many executions of the algorithm. It turns out that all of them produce the right partition of the states. However, different executions may give rise to different sequences of splitting and

```
P ← {F, F^c}
for all a ∈ A do
    ADD((min(F, F^c), a), W)
while W ≠ ∅ do
    (C, a) ← SOME(W)                    ▷ takes some element in W
    for each B ∈ P split by (C, a) do
        B′, B″ ← SPLIT(B, C, a)
        REPLACE B by B′ and B″ in P
        for all b ∈ A do
            if (B, b) ∈ W then
                REPLACE (B, b) by (B′, b) and (B″, b) in W
            else
                ADD((min(B′, B″), b), W)
```

Algorithm 1: HOPCROFTMINIMIZATION

also to different running time. Hopcroft has proved that the running time of any execution is bounded by $O(|A| \times n \log n)$.

## *3.2 Cyclic automata*

The behavior of Hopcroft's algorithm is better understood on a family of automata that we introduce now, called *cyclic automata*.

Let $w = b_1 \cdots b_n$ be a word of length $n$ over the binary alphabet $\{0, 1\}$. We define an automaton $\mathcal{A}_w$ over the unary alphabet $\{a\}$ as follows. The state set of $\mathcal{A}_w$ is $\{1, \ldots, n\}$ and the next state function is defined by $i \cdot a = i+1$ for $i < n$ and $n \cdot a = 1$. Note that the underlying labeled graph of $\mathcal{A}_w$ is just a cycle of length $n$. The final states really depend on $w$. The set of final states of $\mathcal{A}_w$ is $F = \{1 \leq i \leq n \mid b_i = 1\}$.

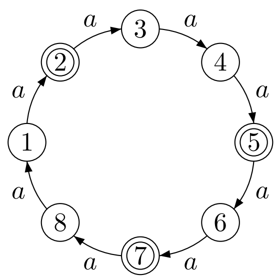


**Fig. 1:** Cyclic automaton $\mathcal{A}_w$ for $w = 01001010$.

For a binary word $u$, we define $Q_u$ to be the set of states of $\mathcal{A}_w$ which are the starting positions of circular occurrences of $u$ in $w$. If $u$ is the empty word, then $Q_u$ is by convention the set $Q$ of all states of $\mathcal{A}_w$. By definition, the set $F$ of final states of $\mathcal{A}_w$ is $Q_1$ while its complement $F^c$ is $Q_0$.

Consider the automaton $\mathcal{A}_w$ for $w = 01001010$ given in is Fig. 1. The sets $Q_1$, $Q_{01}$ and $Q_{11}$ of states are respectively $\{2, 5, 7\}$, $\{1, 4, 6\}$ and $\emptyset$.

The following statements appears in Castiglione et al. (2007):

**Proposition 2** *Let $w$ be a standard word. Hopcroft's algorithm on the cyclic automaton $\mathcal{A}_w$ is uniquely determined. In particular, at each step, the waiting set is a singleton.*

In fact, the execution is described in the next statement. The first part of the statement is from Borel and Reutenauer (2006). They prove that the number of circular (special) factors in fact characterizes standard words.

**Proposition 3** *Let $w$ be a standard word and set $m = |w|$. For $0 \leq i \leq m-2$, the word $w$ has exactly $i+1$ circular factors of length $i$ and exactly one circular special factor of length $i$.*

*At each step $i$ of the execution of Hopcroft's algorithm, the current partition is composed of the $i+1$ classes $Q_v$ indexed by the cyclic factors of length $i$, and the waiting set is a singleton. This singleton is the smaller of the sets $Q_{u0}$, $Q_{u1}$, where $u$ is the unique cyclic special factor of length $i-1$.*

As already mentioned, the complexity of Hopcroft's algorithm is proportional to the sum of the sizes of the sets that are processed in the waiting set. As a consequence of the previous description, one gets the following corollary.

**Corollary 4** *Let $d = (d_1, d_2, \ldots)$ be a directive sequence. Let $(s_n)_{n\geq 0}$ be the standard sequence defined by $d$. Then the complexity of Hopcroft's algorithm on the automaton $\mathcal{A}_{s_n}$ is proportional to $\|s_n\|$.*

Recall that $\|w\| = \sum_{u\in CF(w)} \min(|w|_{u0}, |w|_{u1})$. The main result of this paper is the following theorem.

**Theorem 5** *Let $d = (d_1, d_2, \ldots)$ be an ultimately periodic directive sequence. Let $(s_n)_{n\geq 0}$ be the standard sequence defined by $d$. Then $n = \Theta(\log|s_n|)$, $\|s_n\| = \Theta(n|s_n|)$, and the complexity of Hopcroft's algorithm on the automata $\mathcal{A}_{s_n}$ is in $\Theta(N \log N)$ with $N = |s_n|$.*

This theorem is proved in Castiglione et al. (2007) in the case of the Fibonacci words, with directive sequence $d = (\overline{1})$. It remains open whether it holds for arbitrary directive sequences.

# 4 The generating series of the complexity

Let $d = (d_1, d_2, \ldots)$ be a directive sequence. Let $(s_n)_{n\geq 0}$ be the standard sequence defined by $d$, and set

$$a_n = |s_n|_1\,, \quad c_n = \|s_n\|\,.$$

These quantities depend of course on $d$, although this is not indicated in the notation. Observe that $a_0 = 1, a_1 = 0$, $a_2 = 1$ and $a_3 = d_2$ because $s_2 = 0^{d_1}1$ and $s_3 = (0^{d_1}1)^{d_2}0$. In particular, the value $d_1$ plays no role in the sequence $(a_n)_{n\geq 0}$. This will be used later.

Observe also that $c_0 = 0$, $c_1 = 0$ because $s_0 = 1$ and $s_1 = 0$, and that $c_2 = d_1$. Indeed, the circular special factors of $s_2$ are the words $0^\ell$ for $0 \leq \ell < d_1$, and each factor $0^\ell 1$ for $0 \leq \ell < d_1$ has exactly one circular occurrence in $s_2$.

### 4.1 A first equation

The *generating series* $A_d(x)$ and $C_d(x)$ are defined by

$$A_d(x) = \sum_{n\geq 1} a_n x^n\,, \qquad C_d(x) = \sum_{n\geq 0} c_n x^n\,.$$

Observe that we do not include the constant $a_0$ in the series $A_d$. Observe also that the series $A_d$ does not depend on the value of the first term $d_1$ of $d$.

Given a directive sequence $d = (d_1, d_2, \ldots)$, we define a shorthand notation $T(d)$ by $T(d) = \tau^{d_1}(d)$. Thus $T(d_1, d_2, d_3, \ldots) = (d_2, d_3, \ldots)$. Observe that $T^i(d) = (d_{i+1}, d_{i+2}, \ldots)$. We also define a *Kronecker symbol* $\delta$ by

$$\delta(d) = \begin{cases} 0 & \text{if } d_1 > 1\,, \\ 1 & \text{otherwise.} \end{cases}$$

The following equation results from the combinatorial lemmas given below.

**Proposition 6** *For any directive sequence $d = (d_1, d_2, \ldots)$, one has*

$$C_d(x) = A_d(x) + x^{\delta(d)} C_{\tau(d)}(x) + x^{1+\delta(T(d))} C_{\tau(T(d))}(x)\,. \tag{1}$$

**Example 7** Consider the directive sequence $d = (\overline{1})$ of the Fibonacci words. Since $\tau(d) = T(d) = d$, and $\delta(d) = 1$, Equation (1) becomes

$$C_d(x) = A_d(x) + (x + x^2) C_d(x)\,.$$

### 4.2 Acceleration

Iterated application of Equation (1) gives a system of equations which is finite when the directive sequence $d$ is periodic. Indeed, call *suffix* of order $m$ of $d$ any of the directive sequence $\tau^m(d)$. If $d$ is purely periodic with (minimal) period $k$, then $d = \tau^{d_1+\cdots+d_k}(d)$, so $d$ has exactly $N = d_1 + \cdots + d_k$ distinct suffixes. Let $U$ be this set of suffixes. Each of the suffixes $u$ in $U$ satisfies Equation (1) with $d$ replaced by $u$, so we get a system of $N$ equations in the variables $C_u(x)$, for $u \in U$.

$$C_u(x) = A_u(x) + x^{\delta(u)} C_{\tau(u)}(x) + x^{1+\delta(T(u))} C_{\tau(T(u))}(x)\,.$$

Each of the $C_u$ depends only linearly on $C_{\tau(u)}$ and $C_{\tau(T(u))}$, with coefficients which are monomials among $1, x, x^2$. We will show how to replace this system of $N$ equations by a system of only $k$ equations. This is done by collapsing appropriate equations of the previous system. We start with an example.

**Example 8** Consider the directive sequence $d = (\overline{2,3})$. Equation (1) applied iteratively gives the following five equations:

$$\begin{aligned}
C_{(\overline{2,3})} &= A_{(\overline{2,3})} + C_{(1,\overline{3,2})} + xC_{(2,\overline{2,3})} \\
C_{(1,\overline{3,2})} &= A_{(1,\overline{3,2})} + xC_{(\overline{3,2})} + xC_{(2,\overline{2,3})} \\
C_{(2,\overline{2,3})} &= A_{(2,\overline{2,3})} + C_{(1,\overline{2,3})} + xC_{(1,\overline{3,2})} \\
C_{(\overline{3,2})} &= A_{(\overline{3,2})} + C_{(2,\overline{2,3})} + xC_{(1,\overline{3,2})} \\
C_{(1,\overline{2,3})} &= A_{(1,\overline{2,3})} + xC_{(\overline{2,3})} + xC_{(1,\overline{3,2})}
\end{aligned}$$

Observe that $A_{(\overline{2,3})} = A_{(1,\overline{3,2})}$ and $A_{(\overline{3,2})} = A_{(2,\overline{2,3})} = A_{(1,\overline{2,3})}$ because these series do not depend on the first term of the directive sequence. Setting $D_1 = C_{(1,\overline{3,2})}$ and $D_2 = C_{(2,\overline{2,3})}$, we get

$$C_{(\overline{2,3})} = A_{(\overline{2,3})} + D_1 + xD_2\,,$$

where $D_1$ and $D_2$ satisfy the equations

$$\begin{aligned} D_1 &= A_{(\overline{2,3})} + xA_{(\overline{3,2})} + 2xD_2 + x^2 D_1 \\ D_2 &= 2A_{(\overline{3,2})} + xA_{(\overline{2,3})} + 3xD_1 + x^2 D_2\,. \end{aligned}$$

Thus the original system of 5 equations in the $C_u$ is replaced by a system of 2 equations in $D_1$ and $D_2$.

Let $d = (d_1, d_2, \ldots)$ be a directive sequence, and for $i \geq 1$, set

$$e_i = T^{i-1}(d) = (d_i, d_{i+1}, \ldots)\,.$$

Set also

$$D_i = x^{\delta(e_i)} C_{\tau(e_i)}\,, \qquad B_i = (d_i - 1)A_{e_i} + xA_{e_{i+1}}\,.$$

With these notations, the following system of equation holds.

**Proposition 9** *The following equations hold*

$$C_d = A_d + D_1 + xD_2 \tag{2}$$

$$D_i = B_i + d_i x D_{i+1} + x^2 D_{i+2} \qquad (i \geq 1) \tag{3}$$

Equations (2) and (3) hold for arbitrary, even non-periodic directive sequences. If a directive sequence $d$ is periodic with period $k$, then $D_{k+1} = D_1$ and $D_{k+2} = D_2$. Observe that the equations of the previous example have precisely this form.

We now turn to the derivation of expressions for the number of occurrences of circular special factors in the words generated by some directive sequence $d$ in terms of the corresponding numbers associated to the directive sequence $\tau(d)$ and $T(d)$. For this, we will observe how circular special factors propagate through some particular morphisms.

Let $d = (d_1, d_2, \ldots)$ be a fixed directive sequence, and let $(s_n)_{n\geq 0}$ be the sequence of standard words generated by $d$. The following four lemmas hold. Lemmas 10 and 12 are similar to lemmas proved in Castiglione et al. (2007), the proofs of Lemmas 11 and 13 are similar.

**Lemma 10** *Assume $d_1 = 1$, and let $t_n$ be the sequence of standard words generated by $\tau(d) = (d_2, d_3, \ldots)$. Let $\varphi$ be the morphism defined by $\varphi(0) = 01$ and $\varphi(1) = 0$. Then $s_{n+1} = \varphi(t_n)$ for $n \geq 1$. If $v$ is a circular special factor of $t_n$, then $\varphi(v)0$ is a circular special factor of $s_{n+1}$. Conversely, if $w$ is a circular special factor of $s_{n+1}$ starting with 0, then $w$ has the form $w = \varphi(v)0$ for some circular special factor $v$ of $t_n$. Moreover, $|s_{n+1}|_{w0} = |t_n|_{v1}$ and $|s_{n+1}|_{w1} = |t_n|_{v0}$.*

**Lemma 11** *Assume $d_1 > 1$, and let $t_n$ be the sequence of standard words generated by $\tau(d) = (d_1 - 1, d_2, d_3, \ldots)$. Let $\psi$ be the morphism defined by $\psi(0) = 0$ and $\psi(1) = 01$. Then $s_n = \psi(t_n)$ for $n \geq 1$. If $v$ is a circular special factor of $t_n$, then $\psi(v)0$ is a circular special factor of $s_n$. Conversely, if $w$ is a circular special factor of $s_n$ starting with 0, then $w$ has the form $w = \psi(v)0$ for some circular special factor $v$ of $t_n$. Moreover, $|s_n|_{w0} = |t_n|_{v0}$ and $|s_n|_{w1} = |t_n|_{v1}$.*

**Lemma 12** *Assume $d_2 = 1$, and let $t_n$ be the sequence of standard words generated by $\tau T(d) = (d_3, d_4, \ldots)$. Let $\alpha$ be the morphism defined by $\alpha(0) = 10^{d_1+1}$ and $\alpha(1) = 10^{d_1}$. Then $s_{n+2}0^{d_1} = 0^{d_1}\alpha(t_n)$ for $n \geq 0$. If $v$ is a circular special factor of $t_n$, then $\alpha(v)10^{d_1}$ is a circular special factor of $s_{n+2}$. Conversely, if $w$ is a circular special factor of $s_{n+2}$ starting with 1, then $w$ has the form $w = \alpha(v)10^{d_1}$ for some circular special factor $v$ of $t_n$. Moreover, $|s_{n+2}|_{w0} = |t_n|_{v0}$ and $|s_{n+2}|_{w1} = |t_n|_{v1}$.*

**Lemma 13** *Assume $d_2 > 1$, and let $t_n$ be the sequence of standard words generated by $\tau T(d) = (d_2 - 1, d_3, d_4, \ldots)$. Let $\beta$ be the morphism defined by $\beta(0) = 10^{d_1}$ and $\beta(1) = 10^{d_1+1}$. Then $s_{n+1}0^{d_1} = 0^{d_1}\beta(t_n)$ for $n \geq 1$. If $v$ is a circular special factor of $t_n$, then $\beta(v)10^{d_1}$ is a circular special factor of $s_{n+1}$. Conversely, if $w$ is a circular special factor of $s_{n+1}$ starting with 1, then $w$ has the form $w = \beta(v)10^{d_1}$ for some circular special factor $v$ of $t_n$. Moreover, $|s_{n+1}|_{w0} = |t_n|_{v1}$ and $|s_{n+1}|_{w1} = |t_n|_{v0}$.*

# 5 Closed forms for the generating series

The series $A_d(x) = \sum a_n x^n$ and $C_d = \sum c_n x^n$ have an expression in closed form in the particular case where the directive sequence is periodic.

**Theorem 14** *If $d$ is a periodic directive sequence with period $k$, then*

$$A_d(x) = \sum a_n x^n = x\frac{R(x)}{Q(x)} \quad \textit{and} \quad C_d(x) = \sum c_n x^n = \frac{S(x)}{Q(x)^2},$$

*where $R(x)$ and $S(x)$ are polynomials and*

$$Q(x) = 1 - Z(d_1, \ldots, d_k)x^k + (-1)^k x^{2k}$$

*where $Z(x_1, \ldots, x_k)$ is a polynomial in the variables $x_1, \ldots, x_k$. Moreover,*

$$a_n = \Theta(\rho^n) \quad \textit{and} \quad c_n = \Theta(n\rho^n)$$

*where $\rho$ is the unique real root greater than 1 of the reciprocal polynomial of $Q(x)$.*

The polynomials $Z$ have a special form that will be given below, and they have an interesting combinatorial interpretation. They will be called *circular continuant polynomials*. We will show later that the reciprocal polynomial of $Q(x)$ have two real roots which are irrational numbers, and that the greater of them is strictly greater than 1.

Let $d$ be a directive sequence. By definition, the sequence $a_n = |s_n|_1$ counting the number of 1's in the standard words $s_n$ defined by $d$ verify $a_0 = 1$, $a_1 = 0$, and satisfy the recurrence relations

$$a_{n+1} = d_n a_n + a_{n-1} \qquad (n \geq 1).$$

Assume the directive sequence $d$ has period $k$. Then the coefficients $d_n$ in the recurrence relation are repeating with period $k$. Our aim is to prove that the sequence $a_n$ is a solution of a unique linear recurrence relation, with constant coefficients, of order $2k$.

## 5.1 Circular continuant

We present here a generalization of continuant polynomials that arise naturally when considering circular words. We use these polynomials for solving the recurrence relation counting occurrences of the letter 1 in standard words.

**Condensation** Let $x_1, \ldots, x_n$ be variables. We consider circular words over these variables viewed as letters, and a replacement operation that we call *condensation* which replaces a factor $x_i x_{i+1}$ of variables with consecutive indexes by 1. The replacement of $x_n x_1$ by 1 is allowed in this operation. Thus for $x_1x_2x_3x_4$, there are four condensations, to $x_3x_4$ (by removing $x_1x_2$), to $x_1x_4$ (by removing $x_2x_3$), to $x_1x_2$ (by removing $x_3x_4$), to $x_2x_3$ (by removing $x_4x_1$). Two terms can be condensed further, namely $x_3x_4$ and $x_1x_2$. Both are condensed to 1. Observe that $x_1x_4$ cannot be condensed since the indexes are not consecutive.

We call *circular continuant polynomial* of order $n$ the polynomial $Z(x_1, \ldots, x_n)$ in commutative variables which is the sum of all monomials obtained from $x_1x_2 \cdots x_n$ by repeated condensation. The following are the first circular continuant polynomials.

$$\begin{aligned}
Z(x_1) &= x_1 \\
Z(x_1, x_2) &= x_1x_2 + 2 \\
Z(x_1, x_2, x_3) &= x_1x_2x_3 + x_1 + x_2 + x_3 \\
Z(x_1, x_2, x_3, x_4) &= x_1x_2x_3x_4 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 + 2\,.
\end{aligned}$$

Observe the constant term 2 in $Z(x_1, x_2)$ which comes from the condensations of both $x_2x_1$ and of $x_1x_2$. The constant term 2 always appears when the number of variables is even. The terminology has been chosen because of the resemblance between the polynomials and the (non-circular) continuant polynomials $K(x_1, \ldots, x_k)$ as they arise in the study of continued fractions (see e.g. Graham et al. (1994)). These polynomials are defined by the condensation rule applied to the non-circular word $x_1x_2 \cdots x_n$. In particular, the block $x_nx_1$ is *not* condensed. The first continuant polynomials are

$$\begin{aligned}
K(x_1) &= x_1 \\
K(x_1, x_2) &= x_1x_2 + 1 \\
K(x_1, x_2, x_3) &= x_1x_2x_3 + x_1 + x_3 \\
K(x_1, x_2, x_3, x_4) &= x_1x_2x_3x_4 + x_1x_2 + x_3x_4 + x_1x_4 + 1\,.
\end{aligned}$$

## 5.2 Continued fraction

Let $d = (d_1, d_2, d_3, \ldots)$ be a sequence of non-zero numbers. The *continued fraction* defined by $d$ is denoted $\alpha = [d_1, d_2, d_3, \ldots]$ and is defined by

$$\alpha = d_1 + \cfrac{1}{d_2 + \cfrac{1}{d_3 + \cdots}}\,.$$

The finite initial parts $[d_1, d_2 \ldots, d_n]$ of $d$ define rational numbers called partial quotients. It is well-known that these partial quotients are the numbers

$$\frac{K(d_1, d_2, \ldots, d_n)}{K(d_2, d_3, \ldots, d_n)}\,.$$

It follows easily that for the standard words $s_n$ defined by $d$ that for $n \geq 3$

$$|s_n|_0 = K(d_1, \ldots, d_{n-1})\,, \quad a_n = |s_n|_1 = K(d_2, \ldots, d_{n-1})\,, \text{ and } \alpha = \lim \frac{|s_n|_0}{|s_n|_1}\,.$$

# 6 Proof of Theorem 14

Let $d_1, \ldots, d_k$ be a sequence of non-zero numbers. Let $(a_n)_{n\geq 0}$ be a sequence of numbers satisfying, for $i = 1, \ldots, k$, the recurrence relations

$$a_{n+1} = d_i a_n + a_{n-1}\,, \quad n \equiv i \bmod k\,. \tag{4}$$

**Example 15** For $k = 2$, $d_1 = 2, d_2 = 3$, one gets the relations

$$a_{n+1} = \begin{cases} 2a_n + a_{n-1} & \text{if } n \text{ is odd,} \\ 3a_n + a_{n-1} & \text{if } n \text{ is even.} \end{cases}$$

If the initial conditions are for instance $a_0 = 1, a_1 = 0$, then the sequence $(a_n)_{n\geq 0}$ starts with $1, 0, 1, 3, 7, 24, 55, 189, \ldots$. Observe on the first values that $a_n = 8a_{n-2} - a_{n-4}$ for $n \geq 4$. This is a consequence of the result to be proved.

The theorem is a consequence of the following proposition, which gives the precise form of the polynomial $Q(x)$.

**Proposition 16** *If $a_n$ is a sequence satisfying the recurrence relations* (4)*, then for $n \geq 2k$*

$$a_n = Z(d_1, \ldots, d_k)a_{n-k} + (-1)^{k+1} a_{n-2k}\,.$$

The following lemma describes the roots of the polynomial $Q(x)$.

**Lemma 17** *The polynomial $H(x) = x^2 - Z(d_1, \ldots, d_k)x + (-1)^k$ is irreducible over $\mathbb{Q}$ and has one real root strictly greater than 1. The roots of the reciprocal polynomial of $Q(x)$ are the numbers $\rho_1\lambda$, $\rho_2\lambda$, where $\lambda$ ranges over the $k$-th roots of 1 and where $\rho_1^k, \rho_2^k$ are the roots of $H(x)$.*

## 6.1 Computation of the characteristic polynomial

Given a periodic sequence $d = (\overline{d_1, \ldots, d_k})$, the system of equations given in Proposition 9 is finite. In matrix form the $k$ equations (3) take the form

$$M(x)\begin{pmatrix} D_1 \\ \vdots \\ D_k \end{pmatrix} = \begin{pmatrix} B_1 \\ \vdots \\ B_k \end{pmatrix}$$

where

$$M(x) = \begin{pmatrix} 1 & -d_1x & -x^2 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -d_2x & -x^2 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -d_3x & -x^2 & \cdots & 0 \\ & \vdots & & & & & \vdots \\ -x^2 & 0 & 0 & & & 1 & -d_{k-1}x \\ -d_kx & -x^2 & 0 & & & 0 & 1 \end{pmatrix}$$

In order to solve this system, we compute $\det M(x)$, which is of course a polynomial in $x$. Recall that $Q(x) = 1 - Z(d_1, \ldots, d_k)x^k + (-1)^k x^{2k}$.

**Lemma 18** *One has* $\det M(x) = Q(x)$.

### 6.2 Computation of the adjugate matrix

**Lemma 19** *Let*

$$M^{-1}(x) = \frac{1}{Q(x)} \big(P_{ij}(x)\big) \,,$$

*where* $\big(P_{i,j}(x)\big)$ *is the adjugate matrix of* $M(x)$*. For each* $i, j$*, there are integers* $m_{i,j}, \alpha_{i,j}, \beta_{i,j}$ *with* $m_{i,j} < k$ *such that*

$$P_{i,j}(x) = x^{m_{i,j}}(\alpha_{i,j} + x^k \beta_{i,j}) \,.$$

*Moreover, the polynomials* $P_{i,j}(x)$ *and* $Q(x)$ *are relatively prime.*

**Lemma 20** *Let* $\big(P_{i,j}(x)\big)$ *be the adjugate matrix of* $M(x)$*, and let* $\rho$ *be the greater of the real numbers* $\rho_1, \rho_2$*. Each of the series*

$$\frac{P_{i,j}(x)}{Q(x)} = \sum_{n \geq 0} u_n^{(i,j)} x^n \,,$$

*has integral non-negative coefficients. Moreover, if* $P_{i,j}(x)$ *is not null, then there is an integer* $\ell$ *with* $0 \leq \ell < k$ *such that* $u_n^{(i,j)} = \Theta(\rho^n)$ *whenever* $n \equiv \ell$ *mod* $k$*.*

### 6.3 Proof of Theorem 5

We are now ready for the proof of the main result.

**Proof:** First, the coefficients of the series $A_d(x) = \sum_{n\geq 0} a_n x^n$ verify $a_n = \Theta(\rho^n)$ for all $n$. Next, the series $B_i$ are defined as a combination with positive polynomial coefficients of series $A_d$, so their coefficients have the same asymptotic behavior, and each $B_i(x)$ of the form $R_i(x)/Q(x)$ for some polynomial $R_i(x)$.

Each series $D_i$, viewed as a component of the solution of the system of equations, has the form

$$D_i = \frac{1}{Q(x)} \sum_{j=1}^{k} P_{i,j} B_j = \sum_{j=1}^{k} \frac{P_{i,j}(x)}{Q(x)} \frac{R_j(x)}{Q(x)}$$

The polynomials $P_{i,j}$, for $j = 1, \ldots, k$ are not all null. Each of the series $B_i(x)$ has coefficients which are $\Theta(\rho^n)$ for all $n$, and at least one of the series $P_{i,j}(x)/Q(x)$ has coefficients that are $\Theta(\rho^n)$ for all $n$ in some arithmetic progression modulo $k$. This suffices to guarantee that the coefficients of the series $D_i$ are $\Theta(n\rho^n)$ for all $n$. Since $C_d = A_d + D_1 + xD_2$, it follows that $c_n = \Theta(n\rho^n)$. Next $|s_n| = \Theta(\rho^n)$. Indeed $|s_n| = |s_n|_1 + |s_n|_0 \sim |s_n|_1(1+\alpha)$, where $\alpha = [d_1, d_2, \ldots]$, so $|s_n| \sim a_n(1+\alpha)$ and consequently $|s_n| = \Theta(\rho^n)$. In order to complete the proof, it remains to observe that the ultimately periodic can be easily handled by a similar argument. □

## Acknowledgements

We thank Isabelle Fagnot, Christophe Reutenauer and Georges Skandalis for fruitful discussions on various stages of the preparation of the manuscript.


Since acceptation of this communication, the authors have obtained a partial answer to the question whether Theorem 5 holds in general : the conclusion holds for bounded directive sequences but is false in general. A full characterization is still missing.

## References


A. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

J.-P. Allouche and J. Shallit. *Automatic sequences*. Cambridge University Press, Cambridge, 2003.

D. Beauquier, J. Berstel, and P. Chrétienne. *Éléments d'algorithmique*. Masson, 1992.

J. Berstel and O. Carton. On the complexity of Hopcroft's state minimization algorithm. In *Implementation and application of automata*, volume 3317 of *Lect. Notes in Comput. Sci.*, pages 35–44. Springer-Verlag, 2004.

J. Berstel and P. Séébold. Sturmian words. In M. Lothaire, editor, *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and its Applications*, chapter 2, pages 45–110. Cambridge University Press, 2002.

J. Berstel, L. Boasson, O. Carton, and I. Fagnot. A first investigation of Sturmian trees. In W. Thomas and P. Weil, editors, *STACS'2007*, volume 4393 of *Lect. Notes in Comput. Sci.*, pages 73–84. Springer Verlag, 2007.

N. Blum. A $O(n \log n)$ implementation of the standard method for minimizing $n$-state finite automata. *Inform. Proc. Letters*, 57:65–69, 1996.

J.-P. Borel and C. Reutenauer. On Christoffel classes. *Theor. Inform. Appl.*, 40(1):15–27, 2006.

G. Castiglione, A. Restivo, and M. Sciortino. Circular words and automata minimization. In P. Arnoux, N. Bédaride, and J. Cassaigne, editors, *Words 2007*, pages 79–89. Institut de Mathématiques de Luminy, 17–21 september 2007.

R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics*. Addison-Wesley, second edition, 1994.

D. Gries. Describing an algorithm by Hopcroft. *Acta Inform.*, 2:97–109, 1973.

J. E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi and A. Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

T. Knuutila. Re-describing an algorithm by Hopcroft. *Theoret. Comput. Sci.*, 250:333–363, 2001.

N. Pytheas Fogg. *Substitutions in dynamics, arithmetics and combinatorics*, volume 1794 of *Lecture Notes in Mathematics*. Springer Verlag, 2002. Edited by V. Berthé, S. Ferenczi, C. Mauduit and A. Siegel.