

SOME ALGORITHMS TO COMPUTE THE CONJUGATES OF EPISTURMIAN MORPHISMS

GWENAELE RICHOMME¹

Abstract. Episturmian morphisms generalize Sturmian morphisms. They are defined as compositions of exchange morphisms and two particular morphisms L , and R . Epistandard morphisms are the morphisms obtained without considering R . In [14], a general study of these morphisms and of conjugacy of morphisms is given. Here, given a decomposition of an Episturmian morphism f over exchange morphisms and $\{L, R\}$, we consider two problems: how to compute a decomposition of one conjugate of f ; how to compute a list of decompositions of all the conjugates of f when f is epistandard. For each problem, we give several algorithms. Although the proposed methods are fundamentally different, we show that some of these lead to the same result. We also give other algorithms, using the same input, to compute for instance the length of the morphism, or its number of conjugates.

Mathematics Subject Classification. 68R15.

1. INTRODUCTION

Since the works of Morse and Hedlund [12], Sturmian words have been widely studied (see [2] for a recent survey). These infinite words, that are defined over a two-letter alphabet, have a lot of equivalent definitions. When larger alphabets are considered, these definitions give different generalizations of Sturmian words (see for instance [1, 3–7, 9, 10, 13]). *Episturmian words* is one of these generalizations [5, 8], and it partially coincides with previous generalizations [1, 4, 7].

Sturmian (endo)morphisms are defined on two-letter alphabets. They were initially introduced as the morphisms which preserve Sturmian words. In [15], Séébold proved that the monoid of Sturmian morphisms is generated by the exchange (of the two letters) morphism and two other morphisms (L and R).

Keywords and phrases. Combinatorics on words, Sturmian morphisms, conjugacy, algorithms.

¹ LaRIA, Université de Picardie Jules Verne, 5 rue du Moulin Neuf, 80000 Amiens, France; e-mail: richomme@laria.u-picardie.fr

In [5, 8, 9], working on alphabet of arbitrary size, Justin *et al.* called *Episturmian* the (endo)morphisms generated by the permutations and a family of morphisms (two morphisms for each letter in the alphabet) generalizing L and R . In [14], we show that Episturmian morphisms can be defined by exchange morphisms and two morphisms also called L and R , so directly generalizing the binary case. One can note that all these morphisms already appear (even if not explicitly) in some works around generalization of Sturmian words [1, 4, 13]. In [9], Justin and Pirillo show that Episturmian morphisms are the morphisms that preserve Episturmian words. The reader will find a recent survey on Sturmian morphisms in [2].

In [14], a study of intrinsic properties of Episturmian morphisms (without any reference to Episturmian words) is given. We sum up some of these properties. Relations between palindromes and Episturmian morphisms are studied. On binary alphabets, Sturmian morphisms are exactly the invertible morphisms; but, when considering larger alphabets, the monoid of invertible morphisms is no more finitely generated (this result was also proved in [17]). So Episturmian morphisms are invertible, but the converse does not hold necessarily. Generalizing a result from [15], a presentation of the monoid of Episturmian morphisms is stated. This monoid is cancellative and unitary. Consequently, for an Episturmian morphism f given by the images of the letters, as in [2] for Sturmian morphisms, an algorithm is given to compute a decomposition of f over exchange morphisms and $\{L, R\}$.

Most part of [14] concerns conjugacy of Episturmian morphisms. A general study is given and then conjugacy is used in particular to state a presentation of the monoid of Episturmian morphisms. In the present paper, we come back to the conjugacy of Episturmian morphisms. We show that theoretical results in [14] lead to different algorithms to compute any conjugate of an Episturmian morphism, or, to compute the list of conjugates of an epistandard morphism (particular Episturmian morphism).

In Section 2, we recall notions and useful results on words, Episturmian morphisms and conjugacy of morphisms. In Section 3, using Parikh matrices, we present algorithms to compute, given a decomposition over exchanges and $\{L, R\}$ of an Episturmian morphism f , general informations on f as its length or its number of conjugates. In Section 4, we give two algorithms to compute, from a decomposition over exchanges and $\{L, R\}$ of an Episturmian morphism f , a right conjugate of f given by its number. Although the two methods are fundamentally different, we show that when f is epistandard, the two algorithms produce the same decomposition in output. In Section 5, we give two other algorithms for the same purpose. The outputs of these algorithms are different from those of the two algorithms in Section 4. But whatever is f in input, the outputs with these two new algorithms are identical. In Section 6, still with the same input, we give six different algorithms to compute a complete list of conjugates of f . Four of them are based on the algorithms of the previous sections. Once again, although the methods used to design the algorithms are different, we get only two different outputs from these six algorithms: three algorithms give one output, the three others give the other output.

2. WORDS, EPISTURMIAN MORPHISMS, CONJUGACY

In this section, we essentially recall basic notions and results from [14].

2.1. WORDS

Given a finite set X , we will denote by $\#X$ its cardinal, that is, the number of its elements. An *alphabet* A is a set of symbols called *letters*. Here we consider only finite alphabets. A *word* over A is a finite sequence of letters from A . The *empty word* ε is the empty sequence of letters. Equipped with the concatenation operation, the set A^* of words over A is the free monoid with neutral element ε and set of generators A . Given a non-empty word $u = a_1 \dots a_n$ with $a_i \in A$, the *length* $|u|$ of u is the integer n . One has $|\varepsilon| = 0$. For a word u and a letter a , $|u|_a$ is the number of occurrences of a in u . Two words u and v are said *conjugate* if there exists a word w such that $uw = wv$. Powers of a word are defined inductively by $u^0 = \varepsilon$, and for any integer $n \geq 1$, $u^n = uu^{n-1} = u^{n-1}u$.

2.2. EPISTURMIAN MORPHISMS

A (*endo*)*morphism* f on A is an application from A^* to A^* such that for all words u, v over A , $f(uv) = f(u)f(v)$. A morphism is entirely known by the images of the letters of A . The length of f is the value $\|f\| = \sum_{x \in A} |f(x)|$. Given two morphisms f and g , we will denote fg their composition. A particular morphism is the *empty morphism* ε : $\forall a \in A, \varepsilon(a) = \varepsilon$.

Given two letters x, y , the *exchange morphism* of x and y is the morphism defined on A by

$$E_{xy} : \begin{cases} x \rightarrow y, \\ y \rightarrow x, \\ z \rightarrow z, \quad \forall z \notin \{x, y\}. \end{cases}$$

We observe that $E_{xy} = E_{yx}$. Moreover, for any $x \in A$, E_{xx} is the identity morphism (also denoted Id). We denote by $\text{Exch}(A)$ the set of exchange morphisms defined on A (including the identity).

Let A be an alphabet. In [5, 8, 9], Droubay, Justin and Pirillo have introduced for each letter α , the morphisms Ψ_α and $\overline{\Psi}_\alpha$

$$\Psi_\alpha : \begin{cases} \alpha \rightarrow \alpha \\ x \rightarrow \alpha x, \quad \forall x \in A \setminus \{\alpha\} \end{cases} \quad \overline{\Psi}_\alpha : \begin{cases} \alpha \rightarrow \alpha \\ x \rightarrow x\alpha, \quad \forall x \in A \setminus \{\alpha\}. \end{cases}$$

Any morphism obtained by composition of exchange morphisms and morphisms Ψ_α with $\alpha \in A$ will be called (as in [14]) an *epistandard morphism* (standard Episturmian in [8, 9]). In other words, an epistandard morphism is a morphism in

$$\text{Epistand}(A) = (\text{Exch}(A) \cup \{\Psi_\alpha \mid \alpha \in A\})^*.$$

Similarly [8, 9], an *Episturmian morphism* is an element of

$$\text{Episturm}(A) = (\text{Exch}(A) \cup \{\Psi_\alpha, \overline{\Psi}_\alpha \mid \alpha \in A\})^*.$$

When $\#A = 2$, epistandard (resp. Episturmian) morphisms are exactly the *standard* (resp. *Sturmian*) morphisms (see [2]).

Following [14, 15], *in the rest of the paper, we will always consider a finite alphabet A containing at least two letters. We will also distinguish a letter a in A .* Following the original notation of Séébold [15], we denote $L = \Psi_a$ (that is $L(a) = a$, $L(x) = ax$ for $x \neq a$) and $R = \overline{\Psi}_a$ (that is $R(a) = a$, $R(x) = xa$ for $x \neq a$). For any letter α , we have

$$\Psi_\alpha = E_{a\alpha} L E_{a\alpha}, \quad \text{and} \quad \overline{\Psi}_\alpha = E_{a\alpha} R E_{a\alpha}.$$

Thus $\text{Epistand}(A) = (\text{Exch}(A) \cup \{L\})^*$ and $\text{Episturm}(A) = (\text{Exch}(A) \cup \{L, R\})^*$.

Note that, in the particular case where $\#A = 2$, L and R are the morphisms G and \tilde{G} in [2]. So all results here and in [14] can be directly considered for Sturmian morphisms with a usual basis.

In [14], an algorithm is designed to compute a decomposition over $\text{Exch}(A) \cup \{L, R\}$ of a given Episturmian morphism. Such a decomposition is not unique. The following theorem shows what are the basic equalities between decompositions:

Theorem 2.1 ([14], Th. 7.1) (see also [15] for the binary case). *The monoid $\text{Episturm}(A)$ with set of generators $\text{Exch}(A) \cup \{L, R\}$ has the following presentation (x, y, z, t are pairwise different letters):*

$$\begin{aligned} E_{xy} E_{xy} &= Id, \\ E_{xy} E_{yz} &= E_{yz} E_{zx}, \\ E_{xy} E_{zt} &= E_{zt} E_{xy}, \\ E_{xy} L &= L E_{xy} && \text{when } a \notin \{x, y\}, \\ E_{xy} R &= R E_{xy} && \text{when } a \notin \{x, y\}, \\ LE_1 L E_2 \dots L E_k R &= R E_1 R E_2 \dots R E_k L \end{aligned}$$

where $k \geq 1$ is an integer and E_1, \dots, E_k are exchange morphisms such that $E_1 \dots E_k(a) = a$, and for each integer i , $2 \leq i \leq k$, $E_i \dots E_k(a) \neq a$.

2.3. CONJUGACY

The notion of conjugation of Sturmian morphisms was introduced by Séébold [16]. On two-letter alphabets, the definition of conjugation is a bit different from the notion of conjugacy given in [2] but the ideas are the same, and similar results are obtained. Conjugacy can be easierly generalized to arbitrary alphabets than conjugation. Thus, we follow [2, 14].

A morphism g is a *right conjugate* of a morphism f defined on A , in symbols $f \triangleleft g$, if there exists a word w such that $f(x)w = wg(x)$ for all words x in A^* .

Here, we will also say that f is a *left conjugate* of g , and we will sometimes write $f \triangleleft_w g$. For instance, $L \triangleleft_a R$.

Basic properties of conjugacy are given by the following lemma:

Lemma 2.2 ([14], Lem. 3.1) (see also [2]). *Let f, f', g, g', h be some morphisms and let w_1, w_2 be some words.*

- (1) *If $f \triangleleft_{w_1} g$ and $g \triangleleft_{w_2} h$ then $f \triangleleft_{w_1 w_2} h$.*
- (2) *If $g \neq \epsilon$, $f \triangleleft_{w_1} g$, $f' \triangleleft_{w_2} g$ and $|w_1| \leq |w_2|$, then there exists a word w_3 such that $w_2 = w_3 w_1$ and $f' \triangleleft_{w_3} f$.*
- (3) *If $f \neq \epsilon$, $f \triangleleft_{w_1} g$, $f \triangleleft_{w_2} g'$ and $|w_1| \leq |w_2|$, then there exists a word w_3 such that $w_2 = w_1 w_3$ and $g \triangleleft_{w_3} g'$.*
- (4) *If $f \triangleleft_{w_1} g$ and $f' \triangleleft_{w_2} g'$ then $f f' \triangleleft_{f(w_2)w_1} g g'$.*

The family of Episturmian morphisms is self conjugated:

Proposition 2.3 ([14], Cor. 5.5, Cor. 5.6). *Any (right or left) conjugate of an Episturmian morphism is Episturmian.*

Moreover, we have:

Theorem 2.4 ([14], Th. 5.1). *A morphism f is Episturmian if and only if it is a right conjugate of a unique epistandard morphism. This epistandard morphism is obtained from any decomposition of f in elements of $\text{Exch}(A) \cup \{L, R\}$ by replacing all the occurrences of R by L .*

Following this theorem, given an Episturmian morphism f , we denote by $\text{Stand}(f)$ the epistandard morphism which is a left conjugate of f . Note that $\text{Stand}(L) = \text{Stand}(R) = L$, and, for an exchange morphism E , $\text{Stand}(E) = E$. Moreover, if $f = f_1 \dots f_n$ with for all i , $1 \leq i \leq n$, $f_i \in \text{Exch}(A) \cup \{L, R\}$, we have $\text{Stand}(f) = \text{Stand}(f_1) \dots \text{Stand}(f_n)$.

Given an epistandard morphism $f_1 \dots f_n$ with for all i , $1 \leq i \leq n$, $f_i \in \text{Exch}(A) \cup \{R\}$, and given an Episturmian morphism $g_1 \dots g_n$ with for all i , $1 \leq i \leq n$, $g_i \in \text{Exch}(A) \cup \{L, R\}$, we will say that $g_1 \dots g_n$ has property $P(f_1 \dots f_n)$ if for all i , $1 \leq i \leq n$, $\text{Stand}(g_i) = f_i$.

We denote by $\text{NbR}(f)$ the number of right conjugates of a morphism f . For instance, since the right conjugates of L are L and R , and since the unique right conjugate of R is R itself, $\text{NbR}(L) = 2$ and $\text{NbR}(R) = 1$. For any morphism f , we have $\text{NbR}(f) \geq 1$ since f is always its own right conjugate ($f \triangleleft_\epsilon f$). Similarly we can define the number $\text{NbL}(f)$ of left conjugates of f .

In [14], it is stated that (left and right) conjugates of a morphism can be ordered. In the particular case of Episturmian morphism, we have (see [14], Lem. 3.3, Lem. 3.4, Lem. 3.6):

Lemma 2.5. *Let f, g be Episturmian morphisms.*

- (1) *The morphism g is a right conjugate of f if and only if there exists a unique word w such that $f \triangleleft_w g$. Moreover $0 \leq |w| \leq \text{NbR}(f) - 1$.*
- (2) *The morphism g is a left conjugate of f if and only if there exists a unique word w such that $g \triangleleft_w f$. Moreover $0 \leq |w| \leq \text{NbL}(f) - 1$.*

Note that this lemma is in fact true for all non-periodic morphisms (see [14]).

Using Lemma 2.2, the previous lemma shows that there exists a one-to-one correspondance between conjugates of an Episturmian morphism and integers in $[0..NbR(f) - 1]$.

According to the previous lemma and following [16], for f, g Episturmian morphisms, we say that g is the $|w|^{\text{th}}$ right conjugate of f if w is the word such that $f \triangleleft_w g$. Of course, f is the 0^{th} conjugate of f . If $|w| = 1$, g will be called the *first* (right) conjugate of f , and f will be called the *previous* (left) conjugate of g . If $|w| = NbR(f) - 1$, g will be called the *last* (right) conjugate of f .

Once again using Lemma 2.2, we can see:

Property 2.6. *Given an integer $p \geq 0$, the $(p + 1)^{\text{th}}$ right conjugate (if it exists) of an Episturmian morphism f is the first right conjugate of the p^{th} right conjugate of f .*

Now, let $NbC(f)$ be the total number of left or right conjugates of a morphism f , that is, $NbC(f) = \#\{g \mid g \triangleleft f \text{ or } f \triangleleft g\}$. For instance $NbC(\epsilon) = 1$, $NbC(L) = NbC(R) = 2$. We have:

Lemma 2.7 ([14], Lem. 3.7). *Let f be an Episturmian morphism.*

- (1) $NbC(f) = NbR(f) + NbL(f) - 1$.
- (2) *For any right conjugate g of f , $NbC(f) = NbC(g)$.*

3. COMPUTATION OF THE NUMBERS OF CONJUGATES

In [14] (Prop. 3.8), it was proved that, given an Episturmian morphism f , the values $NbC(f)$, $NbR(f)$ and $NbL(f)$ can be computed in time $O(\|f\|)$. The underlying algorithms assumed that the morphism f was given by the images of the letters. In this section, we prove a similar result when the morphism is given by a decomposition f_1, \dots, f_n of f over $\text{Exch}(A) \cup \{L, R\}$. These computations are related to the computations of $\|f_1 \dots f_n\|$ and of the values of $|f_1 \dots f_{i-1}(a)|$ for $1 \leq i \leq n$ (we take the convention $f_1 \dots f_0 = Id$). Then we have:

Proposition 3.1 ([14], Prop. 6.1). *For any Episturmian morphism f ,*

$$NbC(f) = \frac{\|f\| - 1}{\#A - 1}.$$

Proposition 3.2 ([14], Prop. 6.2). *If $f = f_1 \dots f_n$ is an Episturmian morphism with $f_i \in \text{Exch}(A) \cup \{L, R\}$,*

- a) $NbL(f) = 1 + \sum_{1 \leq i \leq n | f_i = R} |f_1 \dots f_{i-1}(a)|;$
- b) $NbR(f) = 1 + \sum_{1 \leq i \leq n | f_i = L} |f_1 \dots f_{i-1}(a)|;$
- c) $NbC(f) = 1 + \sum_{1 \leq i \leq n | f_i \in \{L, R\}} |f_1 \dots f_{i-1}(a)|.$

Here, we state:

Proposition 3.3. *Let f_1, \dots, f_n be $n \geq 1$ morphisms in $\text{Exch}(A) \cup \{L, R\}$. Considering $\#A$ as a constant, the values $\|f_1 \dots f_n\|$, $\text{NbC}(f_1 \dots f_n)$, $\text{NbR}(f_1 \dots f_n)$, $\text{NbL}(f_1 \dots f_n)$, $|f_1 \dots f_i(b)|$ (for $0 \leq i \leq n$ and $b \in A$) can all be (simultaneously) computed in $O(n)$ arithmetic operations.*

For the proof, we assume that A is totally ordered, and a is the least letter: $A = \{a_1, \dots, a_n\}$, $a_1 = a$. We use the Parikh matrix of a morphism f , that is, the $\#A \times \#A$ matrix P_f such that $P_f[i, j] = |f(a_j)|_{a_i}$.

For instance if $A = \{a, b, c\}$ (with $a < b < c$) then

$$P_L = P_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_{E_{ab}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

One can verify that if f, g are two morphisms, we have $P_{fg} = P_f P_g$. Using this property, we can compute the matrix of an Episturmian morphism from one of its decomposition over $\text{Exch}(A) \cup \{L, R\}$.

Example. When $A = \{a, b, c\}$, the Parikh matrix of $RE_{ac}E_{ab}RRE_{ac}RE_{bc}L$ is

$$P_{E_{ab}} = \begin{bmatrix} 3 & 7 & 9 \\ 2 & 5 & 6 \\ 0 & 0 & 1 \end{bmatrix}.$$

Proof of Proposition 3.3. Let us consider the following sequence of instructions:

1. let M be a $\#A \times \#A$ matrix of integers
 $M \leftarrow$ identity matrix;
2. let NbL , NbR , NbC , k, x, Lgth be integers
 $\text{NbL} \leftarrow 1$
 $\text{NbR} \leftarrow 1$;
3. for k from 1 to n do
 - 3.1. $x \leftarrow \sum_{i=1}^{\#A} M[i, 1]$;
 - 3.2. if $f_k = R$ then $\text{NbL} \leftarrow \text{NbL} + x$;
 - 3.3. if $f_k = L$ then $\text{NbR} \leftarrow \text{NbR} + x$;
 - 3.4. $M \leftarrow MP_{f_k}$
 end for
4. $\text{NbC} \leftarrow \text{NbR} + \text{NbL} - 1$;
5. $\text{Lgth} \leftarrow (\#A - 1) \text{NbC} + 1$.

We verify that this algorithm allows to get all the expected values. At Step 1, we initialize M in such a way that for all i, j , $1 \leq i, j \leq n$, $M[i, j] = |f_1 \dots f_0(a_j)|_{a_i}$ (recall $f_1 \dots f_0$ denotes the identity morphism). By induction, at the end of the k^{th} loop, thanks to Instruction 3.4, we have for all i, j , $1 \leq i, j \leq n$, $M[i, j]$

$= |f_1 \dots f_k(a_j)|_{a_i}$. From this value of M , we can get in a bounded number of arithmetic operations (since $\#A$ is a constant) for a_j in A , the value $|f_1 \dots f_k(a_j)| = \sum_{i=1}^{\#A} |f_1 \dots f_k(a_j)|_{a_i} = \sum_{i=1}^{\#A} M[i, j]$. In particular, after Instruction 3.1, we have $x = |f_1 \dots f_{k-1}(a)|$. Thus by induction, we can see that after Instruction 3.2 (resp. Instruct. 3.3), $\text{NbL} = 1 + \sum_{1 \leq i \leq k | f_i = R} |f_1 \dots f_{i-1}(a)|$ (resp. $\text{NbR} = 1 + \sum_{1 \leq i \leq k | f_i = L} |f_1 \dots f_{i-1}(a)|$). So at the end of Instruction 3, with an additional $O(1)$ number of arithmetic operations, we have been able to compute the values $|f_1 \dots f_i(b)|$ for i , $0 \leq i \leq n$ and for $b \in A$. We also have $\text{NbL} = 1 + \sum_{1 \leq i \leq n | f_i = R} |f_1 \dots f_{i-1}(a)|$, and, $\text{NbR} = 1 + \sum_{1 \leq i \leq n | f_i = L} |f_1 \dots f_{i-1}(a)|$. From Proposition 3.2, this means $\text{NbL} = \text{NbL}(f_1 \dots f_n)$ and $\text{NbR} = \text{NbR}(f_1 \dots f_n)$. It follows from Lemma 2.7 that Instruction 4 computes $\text{NbC}(f_1 \dots f_n)$. Finally from Proposition 3.1, Instruction 5 computes the length of $f_1 \dots f_n$, that is, $\|f_1 \dots f_n\|$.

To end, we let to the reader to verify that the given sequence of instructions acts in $O(n)$ arithmetic operations (recall that $\#A$ is a constant – if it is not the case, the sequence acts in time $O(n(\#A)^3)$). \square

Note that the values in Proposition 3.3 (and in the sequence of instructions in the proof) can grow exponentially with n so that arithmetic operations can not be considered to be made in bounded time (for instance one can see when $A = \{a, b\}$, for $n \geq 0$, $\text{NbC}((LE)^n) = f_{n+2} - 1 = O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$ where $(f_n)_{n \geq 0}$ is the Fibonacci sequence defined by $f_0 = 1$, $f_1 = 1$ and $\forall n \geq 0$ $f_{n+2} = f_{n+1} + f_n$). Using Lemma 2.7 and Proposition 3.1, we can also see that all these values are in $O(\|f_1 \dots f_n\|)$.

4. COMPUTATION OF A RIGHT CONJUGATE

Let f be an Episturmian morphism, and let $p \geq 0$ be an integer. We want to compute the p^{th} right conjugate (if it exists) of f . In case f is known by the images of the letters, the computation of the images by g of the letters can be easily made. Indeed, first we have to compute the word w of length p such that $f \triangleleft_w g$: for instance, this word is the prefix of length p of $f(a^p)$. Then for each x in A , we can compute $g(x)$ since $f(x)w = wg(x)$ (note that if w is not a prefix of $f(x)w$, then the p^{th} right conjugate of f does not exist).

From now on, we consider that the input morphism is given by one of its decomposition over $\text{Exch}(A) \cup \{L, R\}$. We study the following:

Problem 1. Let $p \geq 0$ and let f be an Episturmian morphism given by a decomposition f_1, \dots, f_n over $\text{Exch}(A) \cup \{L, R\}$ ($n \geq 1$). How to compute the empty sequence if f has not a p^{th} right conjugate, and to compute otherwise a decomposition g_1, \dots, g_n of the p^{th} right conjugate of f such that $g_1 \dots g_n$ has Property $P(\text{Stand}(f_1) \dots \text{Stand}(f_n))$.

Let us recall that the 0^{th} right conjugate of a morphism f is f itself.

When $p = 1$, Problem 1 has already been solved in [14] by the following proposition:

Proposition 4.1 ([14], Prop 5.2). *Let f be an Episturmian morphism on A . Let f_1, \dots, f_n be some elements of $\text{Exch}(A) \cup \{L, R\}$ such that $f = f_1 \dots f_n$.*

The morphism f has a right conjugate different from f if and only if there exists an integer k between 1 and n such that $f_k = L$.

When it is the case, let k be the least integer between 1 and n such that $f_k = L$. For each i between 1 and $k - 1$, let g_i be the morphism defined by:

- $g_i = L$ if $f_i = R$ and the first letter of $f_{i+1} \dots f_k(a)$ is different from a ;
- $g_i = f_i$ otherwise.

Then the first right conjugate of f is the morphism $g_1 g_2 \dots g_{k-1} R f_{k+1} \dots f_n$ ($R f_2 \dots f_n$ when $k = 1$).

This proposition can be (quite verbosely) rewritten into Algorithm 1 that will be used for comparison. We use a function named `first` that, when applied on a non-empty word w , gives the first letter of w . Note that, since morphisms in $\text{Exch}(A) \cup \{L, R\}$ are not erasing, for $i \leq k - 2$, $\text{first}(f_{i+1} \dots f_k(a)) = \text{first}(f_{i+1}(\text{first}(f_{i+2} \dots f_k(a))))$.

Algorithm 1 solves Problem 1 when $p = 1$.

local: i, k integer

x letter

Step 1. Compute the least integer k such that $1 \leq k \leq n$ and $f_k = L$.

If k does not exist, then quit with an empty sequence as output.

Step 2. $x \leftarrow a$

for i from $k - 1$ downto 1 do

$x \leftarrow \text{first}(f_{i+1}(x))$

if $f_i = R$ and $x \neq a$ then $g_i \leftarrow L$

else $g_i \leftarrow f_i$

Step 3. $g_k \leftarrow R$

$g_{k+1}, \dots, g_n \leftarrow f_{k+1}, \dots, f_n$.

As said in [14], an implementation can be done in time $O(n)$. Indeed function `first`, comparisons of morphisms with L , affectations of morphisms, and, list constructions can all be implemented in bounded time.

Now, let us take back the example of the previous section to illustrate Algorithm 1.

Example (continued). We work on alphabet $\{a, b, c\}$, with $n = 9$ and

$$f_1, \dots, f_n = R, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, L.$$

At Step 1, we get $k = 9$. During Step 2, we get successively $g_8 = E_{bc}$, $g_7 = R$, $g_6 = E_{ac}$, $g_5 = L$, $g_4 = L$, $g_3 = E_{ab}$, $g_2 = E_{ac}$, $g_1 = R$. At Step 3, we state

$g_9 = R$. Thus,

$$g_1, \dots, g_n = R, E_{ac}, E_{ab}, L, L, E_{ac}, R, E_{bc}, R.$$

By computing the images of letters, it can be verified that the morphism $g = g_1 \dots g_n$ is the first conjugate of $f = f_1 \dots f_9$:

$$\begin{aligned} f_1 \dots f_9(a) &= a(ba)^2, & g_1 \dots g_9(a) &= (ba)^2a, \\ f_1 \dots f_9(b) &= a(ba)^3a(ba)^2, & g_1 \dots g_9(b) &= (ba)^3a(ba)^2a, \\ f_1 \dots f_9(c) &= a(ba)^2ca(ba)^2a(ba)^2, & g_1 \dots g_9(c) &= (ba)^2ca(ba)^2a(ba)^2a. \end{aligned}$$

And so $f(a)a = ag(a)$, $f(b)a = ag(b)$, $f(c)a = ag(a)$. It follows $f \triangleleft_a g$.

From Property 2.6, using Algorithm 1, we can design naturally Algorithm 2 which is an answer for Problem 1 (for arbitrary value of p).

Algorithm 2 solves Problem 1.

$$g_1, \dots, g_n \leftarrow f_1, \dots, f_n$$

Apply p times Algorithm 1 with input and output g_1, \dots, g_n :

if g_1, \dots, g_n becomes the empty sequence, quit with the empty sequence as output.

Example (continued). If we apply Algorithm 2 with $p = 3$, $n = 9$ and f_1, \dots, f_n as previously, then g_1, \dots, g_n take successively the values:

$$\begin{aligned} &R, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, L; \\ &R, E_{ac}, E_{ab}, L, L, E_{ac}, R, E_{bc}, R; \\ &L, E_{ac}, E_{ab}, R, L, E_{ac}, R, E_{bc}, R; \\ &R, E_{ac}, E_{ab}, R, L, E_{ac}, R, E_{bc}, R. \end{aligned}$$

Time complexity of Algorithm 2 is in $O(n \times \min(p, \text{NbC}(f)))$ and so in $O(n||f||)$. We now consider Algorithm 3 based on Proposition 3.2 that acts in $O(n)$ arithmetic operations.

Example (continued). Let us illustrate Algorithm 3 with the same input as we take with Algorithm 2: $p = 3$, $n = 9$ and $f_1, \dots, f_n = R, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, L$. The initial values of NL , NC and the $|f_1 \dots f_{i-1}(a)|$ for i from 1 to n can be computed using Section 3. By Proposition 3.2, we know that

$$\begin{aligned} \text{NbL}(f_1 \dots f_9) &= 1 + |Id(a)| + |RE_{ac}E_{ab}| + |RE_{ac}E_{ab}R| + |RE_{ac}E_{ab}RRE_{ac}(a)| \\ &= 1 + 1 + 2 + 2 + 5 = 11. \\ \text{NbC}(f_1 \dots f_9) &= \text{NbL}(f_1 \dots f_9) + |RE_{ac}E_{ab}RRE_{ac}(a)RE_{bc}(a)| \\ &= \text{NbL}(f_1 \dots f_9) + 5 = 16. \end{aligned}$$

Thus t is initialized to the value 2.

When f_i is an exchange, that is when $i \in \{8, 6, 3, 2\}$, we get $g_i = f_i$ and the value of t does not change.

For $i = 9$ and $i = 7$, $|f_1 \dots f_{i-1}(a)| = 5 > t$, thus $g_9 \leftarrow R$ and $g_7 \leftarrow R$. For $i = 5$, $|f_1 \dots f_{i-1}(a)| = 2$, thus $g_5 \leftarrow L$ and $t \leftarrow 0$. For $i = 4$ and $i = 1$, $|f_1 \dots f_{i-1}(a)| > 0$, thus $g_4 \leftarrow R$ and $g_1 \leftarrow R$. Observe that we get the same result as with Algorithm 2. We will see in Theorem 4.2 that it is not by chance.

Algorithm 3 solves Problem 1.

```

local:  $i, t, NL, NC$  integer
 $NL \leftarrow \text{NbL}(f_1 \dots f_n)$ 
 $NC \leftarrow \text{NbC}(f_1 \dots f_n)$ 
 $t \leftarrow NC - p - NL$ 
if ( $t < 0$ ) then exit with the empty sequence as output
for  $i$  from  $n$  downto 1 do
  if  $f_i \in \{L, R\}$  then
    if  $|f_1 \dots f_{i-1}(a)| \leq t$  then
       $g_i \leftarrow L$ 
       $t \leftarrow t - |f_1 \dots f_{i-1}(a)|$ 
    else
       $g_i \leftarrow R$ 
  else
     $g_i \leftarrow f_i$ 

```

Proof of validity of Algorithm 3. Let $f = f_1 \dots f_n$ and let g be its p^{th} right conjugate. Let also g_1, \dots, g_n be the result of Algorithm 3. We have to verify that $g = g_1 \dots g_n$. Note that the morphism g has $\text{NbL}(f) + p$ left conjugates. Indeed, using Lemma 2.2, we can verify that the left conjugates of g are the left conjugates of f , together with the i^{th} right conjugate of f for each i such that $1 \leq i \leq p$.

Let t_0 be the value of t after initialization. From Lemma 2.7, the number of right conjugates of g is $\text{NbC}(f) - (\text{NbL}(f) + p) + 1 = t_0 + 1$. So t_0 is the number of right conjugates of g different from g . To continue, we must have $t_0 \geq 0$. Assume that it is the case: $t_0 = \text{NbR}(g) - 1$.

Following Theorem 2.4, the morphism f is a conjugate of a unique epistandard morphism $s_1 \dots s_n$ with (for $1 \leq i \leq n$) $s_i = \text{Stand}(f_i)$: $f_1 \dots f_n$ has Property $P(s_1 \dots s_n)$.

The morphism g can have several decompositions verifying Property $P(s_1 \dots s_n)$. Let us consider the unique one h_1, \dots, h_n such that, for any other different decomposition (if it exists) h'_1, \dots, h'_n verifying Property $P(s_1 \dots s_n)$, if k is the greatest integer such that $h_k \neq h'_k$, then $h_k = L$ and $h'_k = R$. In the rest of the proof, we show that the output of Algorithm 3 is h_1, \dots, h_n .

Before let us observe (by induction) that for any decomposition h'_1, \dots, h'_n verifying Property $P(s_1 \dots s_n)$, we have for all i , $1 \leq i \leq n$, $|h'_1 \dots h'_{i-1}(a)| = |f_1 \dots f_{i-1}(a)|$.

From what precedes and Proposition 3.2, we have $t_0 = \sum_{1 \leq i \leq n | h_i = L} |h_1 \dots h_{i-1}(a)|$.

Let m be an integer between 1 and n . Assume that, $\forall j$, $m+1 \leq j \leq n$, $g_j = h_j$, and, assume that, before the execution of the block of the instruction “for” with $i = m$, $t = \sum_{1 \leq i \leq m | h_i = L} |h_1 \dots h_{i-1}(a)|$. We prove $g_m = h_m$, and, the following:

Fact F: after the execution of the block of the instruction “for” with $i = m$,

$$t = \sum_{1 \leq i \leq m-1 | h_i=L} |h_1 \dots h_{i-1}(a)|.$$

The proof of validity follows by induction.

If $f_m \notin \{L, R\}$, by definition, $f_m = \text{Stand}(f_m) = \text{Stand}(h_m)$. It follows $f_m = h_m$. Moreover by algorithm, $g_m = f_m$. Thus $g_m = h_m$. Fact F follows from $h_m \neq L$.

If $f_m \in \{L, R\}$, then by definition, $h_m \in \{L, R\}$.

If $f_m \in \{L, R\}$ and $|f_1 \dots f_{m-1}(a)| > t$, then, from the value of t , and since $|f_1 \dots f_{m-1}(a)| = |h_1 \dots h_{m-1}(a)|$, we deduce that $h_m \neq L$, that is, since $\text{Stand}(h_m) = \text{Stand}(f_m)$, $h_m = R$. By the algorithm, it follows $g_m = h_m$. Once again, Fact F follows from $h_m \neq L$.

If $f_m \in \{L, R\}$ and $|f_1 \dots f_{m-1}(a)| \leq t$, then we get $g_m = L$. Assume $h_m = R$. The number of right conjugates of $h_1 \dots h_{m-1}$ is $1 + \sum_{1 \leq i \leq m-1 | h_i=L} |h_1 \dots h_{i-1}(a)| = 1 + t > |f_1 \dots f_{m-1}(a)|$. Let h' be the $|f_1 \dots f_{m-1}(a)|^{\text{th}}$ right conjugate of $h_1 \dots h_{m-1}$. The number of right conjugates of h' is $1 + t - |f_1 \dots f_{m-1}(a)|$. Let h'_1, \dots, h'_{m-1} be a decomposition of h' verifying Property $P(s_1 \dots s_{m-1})$. The morphism $h'_1 \dots h'_{m-1} g_m h_{m+1} \dots h_n$ verifies Property $P(s_1 \dots s_n)$ and, by Proposition 3.2(b), has exactly the same number of right conjugates as $h_1 \dots h_n$. So $h'_1 \dots h'_{m-1} g_m h_{m+1} \dots h_n$ is a decomposition of the p^{th} conjugate of f . From $g_m = L$ and $h_m = R$, we get a contradiction with the last part of the definition of h_1, \dots, h_m . Thus $h_m = L = g_m$. Fact F follows from the diminution of t that occurs. \square

From Proposition 3.3, considering $\#A$ as a constant, the initial values of NL , NC and the $|f_1 \dots f_{i-1}(a)|$ for i from 1 to n can be computed in $O(n)$ arithmetic operations. It follows that Algorithm 3 can be implemented in $O(n)$ arithmetic operations. By the remark at the end of Section 3, Algorithm 3 has complexity in time in $O(n||f||)$ as Algorithm 2. Now, we compare the decompositions obtains with Algorithms 2 and 3.

Theorem 4.2. *Let f_1, \dots, f_n be morphisms in $\text{Exch}(A) \cup \{L\}$. With input f_1, \dots, f_n , whatever is p in input, Algorithms 2 and 3 give the same output.*

Example (continued). The decomposition $g_1, \dots, g_n = R, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, L$ is the decomposition obtained by Algorithm 2 from input $n = 9$, $p = 11$, and

$$f_1, \dots, f_n = L, E_{ac}, E_{ab}, L, L, E_{ac}, L, E_{bc}, L.$$

Consequently the decomposition of the third conjugate of g_1, \dots, g_n obtained previously with Algorithm 2 is the same as the decomposition of the 14^{th} conjugate of f_1, \dots, f_n obtained with Algorithm 3.

Theorem 4.2 shows that it is not a matter of chance to have obtained the same decomposition of the third conjugate of g_1, \dots, g_n with Algorithm 3. Indeed, we can see that the initializations of Algorithm 3 when input is g_1, \dots, g_n and $p = 3$, or, when input is f_1, \dots, f_n and $p = 14$, lead to the same initial value of t .

Observe that $RE_{bc}L = LE_{bc}R$. Thus $RE_{ac}E_{ab}RRE_{ac}RE_{bc}L = RE_{ac}E_{ab}RRE_{ac}LE_{bc}R$.

Consequently, when the input is $n = 9$, $p = 0$ and $f_1, \dots, f_n = R, E_{ac}, E_{ab}, R, R, E_{ac}, L, E_{bc}, R$, then Algorithms 2 and 3 do not give the same output. Theorem 4.2 cannot be stated with arbitrary input in $\text{Exch}(A) \cup \{L, R\}$.

Proof of Theorem 4.2. We act by induction on p . Let f_1, \dots, f_n be morphisms in $\text{Exch}(A) \cup \{L\}$.

Assume first $p = 0$. The outputs of Algorithms 2 and 3 are both a decomposition g_1, \dots, g_n that verifies Property $P(f_1 \dots f_n)$. By Theorem 2.1, since $g_1 \dots g_n = f_1 \dots f_n$, we have, for all i , $1 \leq i \leq n$, $f_i = L$ if and only if $g_i = L$. Thus the outputs of Algorithms 2 and 3 are the same.

Assume now $1 \leq p \leq \text{NbC}(f_1 \dots f_n) - 1$. Let g_1, \dots, g_n be the morphisms obtained by Algorithm 3 applied with the integer $p - 1$. Let h_1, \dots, h_n be the morphisms obtained by Algorithm 3 applied with the integer p . By inductive hypothesis, $g_1 \dots g_n$ is also the decomposition obtained from $f_1 \dots f_n$ by Algorithm 2. We have to prove that when we apply Algorithm 1 on $g_1 \dots g_n$, we obtain $h_1 \dots h_n$.

We have $h_1 \dots h_n \neq g_1 \dots g_n$ by Lemma 2.5. Let k be the greatest integer such that $h_k \neq g_k$.

Let t_i (resp. t'_i) be the value of t just before the test " $f_i \in \{L, R\}$ " in Algorithm 3 applied with the integer $(p - 1)$ (resp. p). We have $t_n = t'_n + 1$. Let also t_0 (resp. t'_0) be the value of t at the end of Algorithm 3 applied with the integer $(p - 1)$ (resp. p). By definition of k , we get for each integer i , $k \leq i \leq n$, $t_i = t'_i + 1$. In particular $t_k = t'_k + 1$. Since $g_k \neq h_k$, this implies $|f_1 \dots f_{k-1}(a)| = t_k$, $g_k = L$, $h_k = R$. Moreover for each i , $0 \leq i \leq k - 1$, $t_i = 0$, and thus $g_i \in \text{Exch}(A) \cup \{R\}$. It follows that k is the same as the one which is computed in Algorithm 1.

For $1 \leq i \leq k$, let x_i be the first letter of $g_i \dots g_k(a)$. Note that $x_k = g_k(a) = L(a) = a$, and for $1 \leq i < k$, x_i is the first letter of $g_i(x_{i+1})$. To end the proof, we show by induction on i from $k - 1$ to 1, that $t'_i = |g_1 \dots g_i(x_{i+1})| - 1$ and, if $g_i = R$ then $h_i = L$ if and only if $x_i \neq a$. We already know that $t'_k = t_k - 1 = |f_1 \dots f_{k-1}(a)| - 1$. Note that since $g_1 \dots g_i$ is a right conjugate of $f_1 \dots f_i$ for all i ($0 \leq i \leq n$), $|g_1 \dots g_i(x)| = |f_1 \dots f_i(x)|$. For instance, $t'_k = |g_1 \dots g_{k-1}(a)| - 1$. Since $h_k = R$, we also have $t'_{k-1} = t'_k$. From $x_k = a$, we get $t'_{k-1} = |g_1 \dots g_{k-1}(x_k)| - 1$.

Let i be an integer, $1 \leq i \leq k - 1$, such that $t'_i = |g_1 \dots g_i(x_{i+1})| - 1$.

If g_i is an exchange, we have $x_i = g_i(x_{i+1})$. It follows by Algorithm 3 that $t'_{i-1} = t'_i = |g_1 \dots g_i(x_{i+1})| - 1 = |g_1 \dots g_{i-1}(x_i)| - 1$.

Assume $g_i = R$ (thus $f_i = L$). We have $x_i = x_{i+1}$.

If $x_i = a$, then $x_i = g_i(x_{i+1})$ and so $|f_1 \dots f_{i-1}(a)| = |f_1 \dots f_{i-1}(x_i)| = |g_1 \dots g_{i-1}(x_i)| = |g_1 \dots g_i(x_{i+1})| > t'_i$. It follows $h_i = R$, and $t'_{i-1} = t'_i = |g_1 \dots g_{i-1}(x_i)| - 1$.

If $x_i \neq a$, $t'_i = |g_1 \dots g_{i-1}R(x_{i+1})| - 1 = |g_1 \dots g_{i-1}(x_i a)| - 1 = |g_1 \dots g_{i-1}(x_i)| + |g_1 \dots g_{i-1}(a)| - 1 = |g_1 \dots g_{i-1}(x_i)| + |f_1 \dots f_{i-1}(a)| - 1$. It follows $h_i = L$, and $t'_{i-1} = t'_i - |f_1 \dots f_{i-1}(a)| = |g_1 \dots g_{i-1}(x_i)| - 1$. \square

5. COMPUTATION OF A RIGHT CONJUGATE USING LEFT CONJUGACY

In the previous section, we have recalled a proposition (Prop. 4.1) that allows to compute a decomposition of the first conjugate (when exists) of an Episturmian morphism. The following proposition allows to compute the previous conjugate of an Episturmian morphism.

Proposition 5.1 ([14], Prop. 5.4). *Let f be an Episturmian morphism on A . Let f_1, \dots, f_n be some elements of $\text{Exch}(A) \cup \{L, R\}$ such that $f = f_1 \dots f_n$.*

The morphism f is a right conjugate of another morphism if and only if there exists an integer k between 1 and n such that $f_k = R$.

When it is the case, let k be the least integer between 1 and n such that $f_k = R$. For each i between 1 and $k - 1$, let g_i be the morphism defined by:

- $g_i = R$ if $f_i = L$ and the last letter of $f_{i+1} \dots f_k(a)$ is different from a ;
- $g_i = f_i$ otherwise.

Then the previous right conjugate of f is the morphism $g_1 g_2 \dots g_{k-1} L f_{k+1} \dots f_n$ ($L f_2 \dots f_n$ when $k = 1$).

We let to the reader to design a corresponding algorithm that we will call **Algorithm 4**.

Now let us come back to Problem 1. Let $f = f_1 \dots f_n$ be an Episturmian morphism, let g be its p^{th} right conjugate (if it exists), and let h be its last right conjugate. Let NR be the number of right conjugates of f . The number of left conjugates of g is $NR - p$ (we must have $NR - p \geq 1$). In other words, g is the $(NR - p - 1)^{\text{th}}$ left conjugate of its last conjugate. By Proposition 4.1 and Lemma 2.2, we can see that the last conjugate of g is also the last conjugate of f , and one of its decomposition verifying Property $P(\text{Stand}(f_1) \dots \text{Stand}(f_n))$ is obtained by replacing each $f_i \in \{L, R\}$ by R . Thus we obtain Algorithm 5.

Algorithm 5 solves Problem 1.

local: NR integer

$NR \leftarrow \text{NbR}(f_1 \dots f_n)$

if $NR - p < 1$ then quit with the empty sequence as output.

$g_1 \dots g_n \leftarrow$ last conjugate of $f_1 \dots f_n$

Apply $NR - p - 1$ times Algorithm 4 with input and output g_1, \dots, g_n .

Example (continued). Once again, we take as input $n = 9$, $f_1, \dots, f_n = R, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, L$ and $p = 3$.

We have $NR = 6$. The decomposition of the last conjugate of f_1, \dots, f_n which has to be computed is

$$R, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, R.$$

We iterate twice Algorithm 4. We obtain successively:

$$\begin{aligned} &L, E_{ac}, E_{ab}, R, R, E_{ac}, R, E_{bc}, R. \\ &R, E_{ac}, E_{ab}, L, R, E_{ac}, R, E_{bc}, R. \end{aligned}$$

Observe that we do not get the same output as with Algorithm 2.

As for Algorithm 2, the time complexity of Algorithm 5 is in $O(n||f||)$. As in the previous section, we consider Algorithm 6 which is a greedy algorithm to compute in $O(n)$ arithmetic operations the p^{th} conjugate of an Episturmian morphism.

Algorithm 6 solves Problem 1.

```

local:  $i, t, NL, NC$  integer
 $NL \leftarrow \text{NbL}(f_1 \dots f_n)$ 
 $NC \leftarrow \text{NbC}(f_1 \dots f_n)$ 
 $t \leftarrow p + NL - 1$ 
if ( $t > NC$ ) then exit with the empty sequence as output
for  $i$  from  $n$  downto 1 do
  if  $f_i \in \{L, R\}$  then
    if  $|f_1 \dots f_{i-1}(a)| \leq t$  then
       $g_i \leftarrow R$ 
       $t \leftarrow t - |f_1 \dots f_{i-1}(a)|$ 
    else
       $g_i \leftarrow L$ 
  else
     $g_i \leftarrow f_i$ 

```

We let to the reader to verify as in the previous section the validity of Algorithm 6.

Example (continued). Once again with the same input, we have $NL = 11$, $NC = 16$, and t is initialize to the value $3 + 11 - 1 = 13$. Since $|f_1 \dots f_3(a)| = 2 = |f_1 \dots f_4(a)|$, $|f_1 \dots f_6(a)| = 5$, and $|f_1 \dots f_8(a)| = 5$, we get $g_9 = R$, $g_8 = E_{bc}$, $g_7 = R$, $g_6 = E_{ac}$, $g_5 = R$, $g_4 = L$, $g_3 = E_{ab}$, $g_2 = E_{ac}$ and $g_1 = R$. We obtain the same decomposition as with Algorithm 5.

We also let to the reader to prove:

Theorem 5.2. *When used with the same input, Algorithms 5 and 6 give the same output.*

In Theorem 5.2, there is no restriction on the input (it is the case in Th. 4.2) since the computation in Algorithm 5 is, whatever is the value of p , done from the same decomposition: that of the last conjugate of the input.

To end this section, let us mention that all we have done in this section and in the previous one can be adapted to get algorithms to compute a left conjugate of an Episturmian morphism.

6. COMPUTATION OF ALL THE CONJUGATES

In this section, we want to compute, not only one particular conjugate of an Episturmian morphisms, but all the conjugates.

Any left or right conjugate of f is also a right conjugate of $\text{Stand}(f)$. Thus we treat the

Problem 2. Let f be an epistandard morphism given by a decomposition f_1, \dots, f_n over $\text{Exch}(A) \cup \{L\}$ ($n \geq 1$). How to compute an ordered list $\mathcal{L} = (h_0, \dots, h_{\text{NbC}(f_1 \dots f_n) - 1})$ of the conjugates of f such that for each $1 \leq i \leq \text{NbC}(f_1 \dots f_n) - 1$, h_i is a decomposition of the i^{th} right conjugate of $f_1 \dots f_n$ verifying Property $P(f_1 \dots f_n)$.

Solutions to Problem 1 give naturally solutions to Problem 2. Algorithm 2 (resp. Algorithm 5) can be transformed to give an $O(n\text{NbC}(f))$ (and so $O(n\|f\|)$) time algorithm to solve Problem 2: we call **Algorithm 7** (resp. **Algorithm 8**) these transformations. Moreover, applying Algorithm 3 (resp. Algorithm 6), for each value of p , $0 \leq p \leq \text{NbC}(f) - 1$, we get **Algorithm 9** (resp. **Algorithm 10**) that solves Problem 2 in $O(n\text{NbC}(f))$ arithmetic operations (and so in time $O(n\|f\|^2)$). By Theorem 4.2, Algorithms 7 and 9 (resp. Algorithms 8 and 10) give the same output.

All these algorithms show that, for any conjugate g of an epistandard morphism $f_1 \dots f_n$ (with $f_i \in \text{Exch}(A) \cup \{L\}$), g has at least one decomposition $g_1 \dots g_n$ (with $g_i \in \text{Exch}(A) \cup \{L, R\}$) that verifies Property $P(f_1 \dots f_n)$. Conversely by Theorem 2.4, for any decomposition $g_1 \dots g_n$ with Property $P(f_1 \dots f_n)$, $g_1 \dots g_n$ is a right conjugate of $f_1 \dots f_n$. Thus one idea to solve Problem 2 can be to make out the list of decompositions we can obtain from $f_1 \dots f_n$ replacing some occurrences of L by R . The problem is that we can obtain several decompositions of the same conjugate. For instance, from LL , the two decompositions LR and RL are obtained for the first conjugate. Thus we have to eliminate some decompositions to keep only one for each conjugate (this can be done using Th. 2.1) and we have to order the list. A simpler way to obtain the list is to compute it inductively. Here again, we propose two algorithms for this purpose. The first one is Algorithm 11.

Algorithm 11 solves Problem 2

```

if n = 1
  if  $f_1 \in \text{Exch}(A)$  then  $\mathcal{L} \leftarrow (f_1)$ 
    else  $\mathcal{L} \leftarrow (L, R)$ 
else
  apply recursively Algorithm 11 to compute the list
   $(h_0, \dots, h_{k-1})$  of conjugates of  $f_1 \dots f_{n-1}$ 
  if  $f_n \in \text{Exch}(A)$  then
     $\mathcal{L} \leftarrow (h_0 f_n, \dots, h_{k-1} f_n)$ 
  else
     $j \leftarrow |f_1 \dots f_{n-1}(a)|$ 
  (*)  $\mathcal{L} \leftarrow (h_0 L, \dots, h_{k-1} L, h_{k-j} R, \dots, h_{k-1} R)$ 

```

Let us observe that Algorithm 11 was already presented by Levé and Séébold [11] in case of standard morphisms, that is in the binary case.

Algorithm 12 is a variant of Algorithm 11 obtained (by of course applying recursively Algorithm 12 and) by replacing Instruction (*) by

$$\mathcal{L} \leftarrow (h_0 L, \dots, h_{j-1} L, h_0 R, \dots, h_{k-1} R).$$

Example (continued). The list of conjugates we obtain

if input is	with Algorithm 11	with Algorithm 12
L	(L, R)	as with Algorithm 11
$LE_{ac}E_{ab}$	$(LE_{ac}E_{ab}, RE_{ac}E_{ab})$	as with Algorithm 11
$LE_{ac}E_{ab}L$	$(LE_{ac}E_{ab}L, RE_{ac}E_{ab}L,$	as with Algorithm 11
	$LE_{ac}E_{ab}R, RE_{ac}E_{ab}R)$	
$LE_{ac}E_{ab}LL$	$(LE_{ac}E_{ab}LL, RE_{ac}E_{ab}LL,$	$(LE_{ac}E_{ab}LL, RE_{ac}E_{ab}LL,$
	$LE_{ac}E_{ab}RL, RE_{ac}E_{ab}RL,$	$LE_{ac}E_{ab}LR, RE_{ac}E_{ab}LR,$
	$LE_{ac}E_{ab}RR, RE_{ac}E_{ab}RR)$	$LE_{ac}E_{ab}RR, RE_{ac}E_{ab}RR).$

We can observe that the output with Algorithm 11 (resp Algorithm 12) is the same as with Algorithm 7 (resp. with Algorithm 8). Again, it is not by chance as we will see in Theorem 6.2.

To prove the validity of Algorithms 11 and 12, we need the following lemma:

Lemma 6.1. *For all Episturmian morphisms f ,*

- (1) $\text{NbC}(fE) = \text{NbC}(f)$ for all exchange morphisms E ;
- (2) $\text{NbC}(fL) = \text{NbC}(fR) = \text{NbC}(f) + |f(a)|$;
- (3) $\text{NbC}(f) \geq |f(x)|$, for all letters x .

Proof. The first and the second part are direct consequences of Proposition 3.2(c). They are already mentioned in [14] (Lem. 4.2).

To prove the third part, let f be an Episturmian morphism, and let E be an exchange morphism. We have for all x in A :

- $\text{NbC}(E) = 1 = |E(x)|$;
- $\text{NbC}(L) = \text{NbC}(R) = 2 \geq |L(x)| = |R(x)|$;
- $\text{NbC}(fE) = \text{NbC}(f)$ and thus, since $E(x)$ is a letter, $\text{NbC}(fE) \geq |fE(x)|$;
- $\text{NbC}(fL) = \text{NbC}(fR) = \text{NbC}(f) + |f(a)| \geq |f(x)| + |f(a)| = |fL(x)| = |fR(x)|$.

The proof of the third part follows by induction on the number of morphisms in the decomposition of f over $\text{Exch}(A) \cup \{L, R\}$. \square

Proof of validity of Algorithms 11 and 12. Let f_1, \dots, f_n be morphisms in $\text{Exch}(A) \cup \{L\}$.

If $n = 1$, the algorithms act correctly.

Assume that $n \geq 2$. We denote $f = f_1 \dots f_{n-1}$ and $k = \text{NbC}(f)$. Let (h_0, \dots, h_{k-1}) be the output of one of the two algorithms applied on f_1, \dots, f_{n-1} .

By construction, for any integer i , $1 \leq i \leq k-1$, h_i is the first right conjugate of h_{i-1} . Thus for any morphism ϕ , $h_i\phi$ is the first right conjugate of $h_{i-1}\phi$ (see Lem. 2.2(4)). It follows that $(h_0\phi, \dots, h_{k-1}\phi)$ is a list of k consecutive conjugates of $f_1 \dots f_n\phi$.

If f_n is an exchange morphism, since $\text{NbC}(ff_n) = k$, by Lemma 6.1, the list of conjugates of $f_1 \dots f_n$ is $(h_0f_n, \dots, h_{k-1}f_n)$.

Assume now $f_n = L$. Since h_0L has no left conjugates (it is epistandard), $(h_0L, \dots, h_{k-1}L)$ is the list of the first k conjugates of $f_1 \dots f_n$. In a similar way $(h_0R, \dots, h_{k-1}R)$ is the list of the last k conjugates. The number of conjugates of $f_1 \dots f_n$ is $\text{NbC}(f) + |f(a)|$. We denote as in Algorithm 11 and 12, $j = |f(a)|$. By Lemma 6.1, $j < k$. It follows that $(h_0L, \dots, h_{k-1}L, h_{k-j}R, \dots, h_{k-1}R)$ and $(h_0L, \dots, h_{j-1}L, h_0R, \dots, h_{k-1}R)$ are both the list of the $k+j$ conjugates of ff_n .

The proof of validity ends by induction. \square

As announced in the example, we have:

Theorem 6.2. *From a given input,*

- (1) *Algorithm 11 computes the same output as Algorithms 7 and 9;*
- (2) *Algorithm 12 computes the same output as Algorithms 8 and 10.*

Proof. We prove only Part 1 of this proposition. The second part is similar. We already know that Algorithms 7 and 9 give the same output. Let us compare this output with the one of Algorithm 11. We act by induction on n .

If $n = 1$, the result is true: the output is (f_1) where f_1 is the input.

If $n \geq 2$, we denote $f = f_1 \dots f_{n-1}$ and $k = \text{NbC}(f)$. Let (h_0, \dots, h_{k-1}) be the output of Algorithm 11 applied on f_1, \dots, f_{n-1} . By inductive hypothesis, (h_0, \dots, h_{k-1}) is also the list of decompositions of the k conjugates of $f_1 \dots f_{n-1}$ obtained with Algorithm 7 or 9.

Note that, whatever is a morphism f_n in $\text{Exch}(A) \cup \{L, R\}$ when we apply successively k times Algorithm 1 to obtain successive conjugates of $f_1 \dots f_n$, we obtain the list $(h_0f_n, \dots, h_{k-1}f_n)$.

Thus if f_n is an exchange morphism, we obtain the same output with Algorithms 7 and 11.

Now assume $f_n = L$. The first conjugates of $f_1 \dots f_n$ are $h_0L, \dots, h_{k-1}L$. The last conjugates of $f_1 \dots f_n$ are $h_{k-j}R, \dots, h_{k-1}R$ where $j = |f_1 \dots f_{n-1}(a)|$ as in Algorithm 11. To end the proof we have to show that when we apply Algorithm 3 with input f_1, \dots, f_n , and $p = k$, we get the decomposition $h_{k-j}R$. When we do this application, we initialize NL at 1 and NC at $\text{NbC}(f_1 \dots f_n)$. By Lemma 6.1, $\text{NbC}(f_1 \dots f_n) = \text{NbC}(f_1 \dots f_{n-1}) + |f_1 \dots f_{n-1}(a)| = k + j$. It follows that the initial value of t is $j - 1$. When executing the “for” block in Algorithm 3 with $i = n$, we get $g_n = R$ and t is unchanged. Moreover, after that, the algorithm continues as if the input is f_1, \dots, f_{n-1} and $p = k - j$. Indeed $\text{NbC}(f_1 \dots f_{n-1}) = \text{NbC}(f_1 \dots f_n) - j$ and, by Proposition 3.2 $\text{NbL}(f_1 \dots f_{n-1}) = \text{NbL}(f_1 \dots f_n) - j$. So by inductive hypothesis, the obtained decomposition is $h_{k-j}R$. \square

Acknowledgements. This paper solves questions initially asked by P. Séébold. I thank him for his remarks and his encouragements. Thanks also to an anonymous referee for his interesting remarks.

REFERENCES

- [1] P. Arnoux and G. Rauzy, Représentation géométrique de suites de complexités $2n + 1$. *Bull. Soc. Math. France* **119** (1991) 199-215.
- [2] J. Berstel and P. Séébold, Sturmian words, Chap. 2, edited by M. Lothaire. Cambridge Mathematical Library, *Algebraic Combinatorics on Words* **90** (2002).
- [3] V. Berthé and L. Vuillon, Tilings and rotations on the torus: A two dimensional generalization of Sturmian sequences. *Discrete Math.* **223** (2000) 27-53.
- [4] M.G. Castelli, F. Mignosi and A. Restivo, Fine and Wilf’s theorem for three periods and a generalization of Sturmian words. *Theoret. Comput. Sci.* **218** (1999) 83-94.
- [5] X. Droubay, J. Justin and G. Pirillo, Episturmian words and some constructions of de Luca and Rauzy. *Theoret. Comput. Sci.* **255** (2001) 539-553.
- [6] P. Hubert, Suites équilibrées. *Theoret. Comput. Sci.* **242** (2000) 91-108.
- [7] J. Justin, On a paper by Castelli, Mignosi, Restivo. *RAIRO: Theoret. Informatics Appl.* **34** (2000) 373-377.
- [8] J. Justin, Episturmian words and morphisms (results and conjectures), edited by H. Crapo and D. Senato. Springer-Verlag, *Algebraic Combinatorics and Comput. Sci.* (2001) 533-539.
- [9] J. Justin and G. Pirillo, Episturmian words and Episturmian morphisms. *Theoret. Comput. Sci.* **276** (2002) 281-313.
- [10] J. Justin and L. Vuillon, Return words in Sturmian and Episturmian words. *RAIRO: Theoret. Informatics Appl.* **34** (2000) 343-356.
- [11] F. Levé and P. Séébold, Conjugation of standard morphisms and a generalization of singular words, in *Proc. of the 9th international conference Journées Montoises d’Informatique Théorique*. Montpellier, France (2002).
- [12] M. Morse and G.A. Hedlund, Symbolic Dynamics II: Sturmian trajectories. *Amer. J. Math.* **61** (1940) 1-42.
- [13] G. Rauzy, Suites à termes dans un alphabet fini, in *Séminaire de théorie des Nombres de Bordeaux*. Exposé 25 (1983).
- [14] G. Richomme, *Conjugacy and Episturmian morphisms*, Technical Report 2001-03. LaRIA, *Theoret. Comput. Sci.* (to appear).

- [15] P. Séébold, Fibonacci morphisms and Sturmian words. *Theoret. Comput. Sci.* **88** (1991) 365-384.
- [16] P. Séébold, On the conjugation of standard morphisms. *Theoret. Comput. Sci.* **195** (1998) 91-109.
- [17] Z.X. Wen and Y. Zhang, Some remarks on invertible substitutions on three letter alphabet. *Chin. Sci. Bulletin* **44** (1999) 1755-1760.

Communicated by J. Berstel.

Received October, 2002. Accepted January, 2003.