UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

École Doctorale ICMS
(Information, Communication, Modélisation et Simulation)

Laboratoire d'Informatique Gaspard-Monge UMR 8049

Habilitation thesis

On Some Combinatorial and Algorithmic Aspects of Strings

Gabriele FICI

defended on December 3rd, 2015 with the following committee:

Directeur :	Marie-Pierre Béal	-	Université Paris-Est Marne-la-Vallée
Rapporteurs :	Christiane Frougny	-	Université Paris 8
	Gwenaël Richomme	-	Université Paul-Valéry Montpellier 3
	Wojciech Rytter	-	University of Warsaw
Examinateurs :	Maxime Crochemore	-	King's College London
	Thierry Lecroq	-	Université de Rouen
	Antonio Restivo	-	Università di Palermo

Table of Contents

Résumé

Dans ce mémoire je présente une sélection des résultats que j'ai publiés depuis l'obtention de mon doctorat en 2006. J'ai décidé de n'inclure que les sujets sur lesquelles j'ai l'intention (l'espérance) de continuer à travailler. Pour chaque sujet, je donne seulement une description des résultats obtenus, les papiers originaux étant inclus en appendice à ce mémoire.

Ce document est structuré en huit parties, comme suit :

1. *Prefix Normal Words.* Ce chapitre concerne une nouvelle classe de mots, Prefix Normal Words, qui émergent naturellement dans un problème algorithmique que nous avons introduit en 2009, le problème du Jumbled Pattern Matching. Ce chapitre couvre les résultats contenus dans [32, 74], obtenus en collaboration avec Péter Burcsi, Ferdinando Cicalese et Zsuzsanna Lipták.

2. *Least Number of Palindromes in a Word.* Avec Luca Zamboni, j'ai étudié le problème de déterminer le plus petit nombre de facteurs palindromiques dans un mot infini, sous différentes hypothèses. Ces résultats ont été publiés dans [75].

3. *Bispecial Sturmian Words.* Les mots sturmiens forment probablement la classe la plus connue de mots infinis, grace à la multitude de propriétés combinatoires qu'ils ont et les domaines dans lesquelles on les retrouve. Dans ce chapitre je présente des résultats sur l'extensibilité des mots dans le langage St des facteurs des mots sturmiens (c.à.d. les mots équilibrés binaires). Les résultats de cette partie sont contenus dans [68].

4. *Trapezoidal Words.* Les mots trapézoïdaux sont une généralisation des mots équilibrés binaires, i.e., des facteurs des mots sturmiens. Ils sont définis par la propriété d'avoir au plus $n + 1$ facteurs distincts de longueur n pour tout $n \geq 0$. Je présente des résultats sur les mots trapézoïdaux obtenus en collaboration avec Michelangelo Bucci et Alessandro De Luca [30].

5. *The Array of Open and Closed Prefixes.* Un outil qu'on a employé pour caractériser des propriétés des mots sturmiens (et plus généralement des mots trapézoïdaux) est la dichotomie ouvert/fermé. Un mot fini est fermé si son plus long bord n'a pas d'occurrence interne, sinon il est ouvert. Avec Alessandro De Luca, j'ai étudié le tableau *oc* d'un mot Sturmien, qui est le tableau dont la i-ème case est 0 ou 1 selon que le préfixe de longueur i du mot est ouvert ou fermé. Ces résultats sont contenus dans [58].

6. *Abelian Repetitions in Sturmian Words.* Il y a eu récemment une hausse d'intérêt pour les propriétés abéliennes des mots. Ce chapitre est dévoué à l'étude des répétitions abéliennes dans les mots sturmiens. Les résultas de cette partie sont présents dans [69], et ont été obtenus en collaboration avec Alessio Langiu, Thierry Lecroq, Arnaud Lefebvre, Filippo Mignosi et Élise Prieur-Gaston.

7. *Universal Lyndon Words.* Je présente dans ce chapitre les mots de Lyndon universels, qui sont des mots sur un alphabet de taille n dont la longueur vaut $n!$ et tels que tout conjugué est de Lyndon pour un des différents ordres de l'alphabet. Ces résultats sont contenus dans [38], et ont été obtenus en collaboration avec Arturo Carpi, Štěpán Holub, Jakub Opršal et Marinella Sciortino.

8. *Cyclic Complexity.* Dans ce dernier chapitre je discute d'une nouvelle mesure de complexité des mots infinis, la complexité cyclique. La complexité cyclique d'un mot infini est la fonction qui compte pour tout n le nombre de classes de conjugaison des facteurs de longueur n dans le mot. Ces résultats sont contenus dans [39], et ont été obtenus en collaboration avec Julien Cassaigne, Marinella Sciortino et Luca Q. Zamboni.

Je tiens à remercier tous mes coauteurs pour les échanges d'idées passionnants qu'on a eu pendant ces années !

Abstract

In this report I present a selection of the results I published after the obtention of my Ph.D. in 2006. I decided to include only topics on which I intend (hope) to continue working in the next future. For each topic, I provide here a sketch of the results only, the full papers being attached as an appendix of the report.

This document is structured in eight parts, as follows:

1. *Prefix Normal Words.* This chapter is concerned with a new class of binary words, Prefix Normal Words, which stemmed from an algorithmic problem we introduced in 2009, the Jumbled Pattern Matching Problem. This chapter covers the results contained in [32, 74], obtained in collaboration with Péter Burcsi, Ferdinando Cicalese and Zsuzsanna Lipták.

2. *Least Number of Palindromes in a Word.* Together with Luca Zamboni, I studied the problem of determining the least number of palindromic factors an infinite word must contain, under different hypotheses. These results have been published in [75].

3. *Bispecial Sturmian Words.* Sturmian words are probably the most famous class of infinite words, due to the myriad of combinatorial properties they have and the fields they can be found in. In this chapter I present some results on the extendibility of words in the language St of finite factors of Sturmian words (i.e., binary balanced words). The results of this part are contained in [68].

4. *Trapezoidal Words.* Trapezoidal words are a generalization of balanced binary words, i.e., factors of Sturmian words. They are defined by the property of having at most $n + 1$ distinct factors of length n for every $n \geq 0$. I present results on trapezoidal words obtained in collaboration with Michelangelo Bucci and Alessandro De Luca [30].

5. *The Array of Open and Closed Prefixes.* A tool we used for characterizing combinatorial properties of Sturmian words is the dichotomy open/closed. A finite word is closed if its longest border does not have internal occurrences, otherwise it is open. Together with Alessandro De Luca, I studied the array oc of a Sturmian word, which the array whose ith entry is 0 or 1 depending whether the prefix of length i of the word is open or closed, respectively. These results are contained in [58].

6. *Abelian Repetitions in Sturmian Words.* There has recently been an increase in interest in abelian properties of words. This chapter is devoted to the study of abelian repetitions in

Sturmian words. The results of this part are presented in [69], obtained in collaboration with Alessio Langiu, Thierry Lecroq, Arnaud Lefebvre, Filippo Mignosi and Élise Prieur-Gaston.

7. *Universal Lyndon Words.* I present in this chapter universal Lyndon words, that are words over an alphabet of size n having length $n!$ and with the property that each conjugate is Lyndon with respect to a different order on the alphabet. These results are contained in [38] and have been obtained in collaboration with Arturo Carpi, Štěpán Holub, Jakub Opršal and Marinella Sciortino.

8. *Cyclic Complexity.* In this last chapter I discuss a new measure of complexity of infinite words, cyclic complexity. The cyclic complexity of an infinite word is the function that counts for each n the number of conjugacy classes of factors of length n in the word. These results are contained in [39] and have been obtained in collaboration with Julien Cassaigne, Marinella Sciortino and Luca Q. Zamboni.

I want to thank all my coauthors for the exciting exchanges of ideas we had in these years!

Chapter 1

Prefix Normal Words

We fix the ordered alphabet $\Sigma = \{a, b\}$, with $a < b$. A word $w = w_1 \cdots w_n$ over Σ is a finite sequence of elements from Σ. Its length n is denoted by $|w|$. The empty word is denoted by ε. For any $1 \leq i \leq |w|$, the i-th symbol of a word w is denoted by w_i. As is standard, we denote by Σ^n the words over Σ of length n, and by $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ the set of finite words over Σ. Let $w \in \Sigma^*$. If $w = uv$ for some $u, v \in \Sigma^*$, we say that u is a *prefix* of w and v is a *suffix* of w. A *factor* of w is a prefix of a suffix of w (or, equivalently, a suffix of a prefix). We denote by $\mathrm{Pref}(w)$, $\mathrm{Suff}(w)$, $\mathrm{Fact}(w)$ the set of prefixes, suffixes, and factors of the word w, respectively.

For a letter $a \in \Sigma$, we denote by $|w|_a$ the number of occurrences of a in the word w. The *Parikh vector* of a word w over Σ is defined as $p(w) = (|w|_a, |w|_b)$. The *Parikh set* of w is $\Pi(w) = \{p(v) \mid v \in \mathrm{Fact}(w)\}$, the set of Parikh vectors of the factors of w.

Finally, given a word w over Σ and a letter $a \in \Sigma$, we denote by $P_a(w, i) = |w_1 \cdots w_i|_a$, the number of a's in the prefix of length i of w, and by $pos_a(w, i)$ the position of the i'th a in w, i.e., $pos_a(w, i) = \min\{k : |w_1 \cdots w_k|_a = i\}$. When w is clear from the context, we also write $P_a(i)$ and $pos_a(i)$. Note that in the context of succint indexing, these functions are frequently called *rank* and *select*, cf. [109]: We have $P_a(w, i) = rank_a(w, i)$ and $pos_a(w, i) = select_a(w, i)$.

Definition 1.1. *Let $w \in \Sigma^*$. We define, for each $0 \leq k \leq |w|$,*

$$F_a(w, k) = \max\{|v|_a \mid v \in \mathrm{Fact}(w) \cap \Sigma^k\},$$

the maximum number of a's in a factor of w of length k. When no confusion can arise, we also write $F_a(k)$ for $F_a(w, k)$. The function $F_b(w)$ is defined analogously by taking b in place of a.

Example 1.2. *Take $w = ababbaabaabbbaaabbab$. In Table 1.1, we give the values of F_a and F_b for w.*

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F_a	0	1	2	3	3	4	4	4	5	5	6	7	7	7	8	8	9	9	9	10	10
F_b	0	1	2	3	3	3	4	4	5	5	6	6	7	7	7	8	8	9	9	10	10

Table 1.1: The sequences F_a and F_b for the word $w = ababbaabaabbbaaabbab$.

Lemma 1.1. *Let $w \in \Sigma^*$. The function $F_a(\cdot) = F_a(w, \cdot)$ has the following property:*

$$F_a(j) - F_a(i) \leq F_a(j - i) \qquad \text{for all } 0 \leq i \leq j \leq |w|.$$

In fact, for every word w there is a word w' which realizes the function $F_a(w)$ as its prefix function $P_a(w')$.

Theorem 1.2. *Let $w \in \Sigma^*$. Then there exists a unique word w' s.t. for all $0 \leq k \leq |w|$, $F_a(w, k) = F_a(w', k) = P_a(w', k)$. We call this word w' the* prefix normal form *of w (with respect to a), and denote it $\mathrm{PNF}_a(w)$. Analogously, there exists a unique word w'', such that for all $0 \leq k \leq |w|$, $F_b(w, k) = F_b(w'', k) = P_b(w'', k)$, the prefix normal form of w with respect to b, denoted $\mathrm{PNF}_b(w)$.*

Example 1.3. *Let $w = ababbaabaabbbaaabbab$. The prefix normal forms of w are the words*

$$\mathrm{PNF}_a(w) = aaababbabaabbbababbab,$$

and

$$\mathrm{PNF}_b(w) = bbbaababababaababababa.$$

The operators PNF_a and PNF_b are idempotent operators, that is, if $u = \mathrm{PNF}_x(w)$ then $\mathrm{PNF}_x(u) = u$, for any $x \in \Sigma$. Also, for any $w \in \Sigma^*$ and $x \in \Sigma$, it holds that $\mathrm{PNF}_x(w) = \mathrm{PNF}_x(\tilde{w})$, where $\tilde{w} = w_n w_{n-1} \cdots w_1$ is the reversal of w.

The prefix normal forms of a word allow one to determine the Parikh vectors of the factors of the word, as we will show in Theorem 1.4. We first recall the following lemma from [45], where we say that a Parikh vector q *occurs* in a word w if w has a factor v with $p(v) = q$.

Lemma 1.3 (Interval Lemma [45])**.** *Let $w \in \Sigma^*$. Fix $1 \leq k \leq |w|$. If the Parikh vectors $(x_1, k - x_1)$ and $(x_2, k - x_2)$ both occur in w, then so does $(y, k - y)$ for any $x_1 \leq y \leq x_2$.*

The lemma can be proved with a simple sliding window argument.

Theorem 1.4. *Let w, w' be words over Σ. Then $\Pi(w) = \Pi(w')$ if and only if $\mathrm{PNF}_a(w) = \mathrm{PNF}_a(w')$ and $\mathrm{PNF}_b(w) = \mathrm{PNF}_b(w')$.*

There is a simple geometrical construction for computing the prefix normal forms of a word w, and hence, by Theorem 1.4, the set $\Pi(w)$ of Parikh vectors occurring in w. An example of this construction is given in Fig. 1.1.

Draw in the Euclidean plane the word w by linking, for every $0 \leq i \leq |w|$, the points (i, j), where j is the difference between the number of a's and the number of b's in the prefix of w of length i. That is, draw w by representing each letter a by an upper unit diagonal and each letter b by a lower unit diagonal, starting from the origin $(0, 0)$.

Then draw all the suffixes of w in the same way, always starting from the origin. The region of the plane so delineated is in fact the region of points (x, y) such that there exists a factor v of w such that $x = |v| = |v|_a + |v|_b$ and $y = |v|_a - |v|_b$. Hence $(|v|_a, |v|_b) = (\frac{x+y}{2}, \frac{x-y}{2}) = p(v)$ belongs to $\Pi(w)$.

The region is connected by Lemma 1.3, in the sense that all internal points belong to $\Pi(w)$. The prefix normal forms $\mathrm{PNF}_a(w)$ and $\mathrm{PNF}_b(w)$ are obtained by connecting the upper and the lower points of the region, respectively.

We take a closer look at those words which are in prefix normal form, which we refer to as *prefix normal words*. For simplicity of exposition, from now on we only refer to prefix normality with respect to the letter a.

Definition 1.4. *A* prefix normal word *is a word $w \in \Sigma^*$ such that for every $0 \leq k \leq |w|$, $F_a(w, k) = P_a(w, k)$. That is, a word such that $w = \mathrm{PNF}_a(w)$. We denote by $L_a \subset \Sigma^*$ the language of prefix normal words.*

Lemma 1.5. *Let $w \in L_a$. Then $wa \in L_a$ if and only if for every $0 \leq k < |w|$ the suffix of w of length k has less a's than the prefix of w of length $k + 1$.*

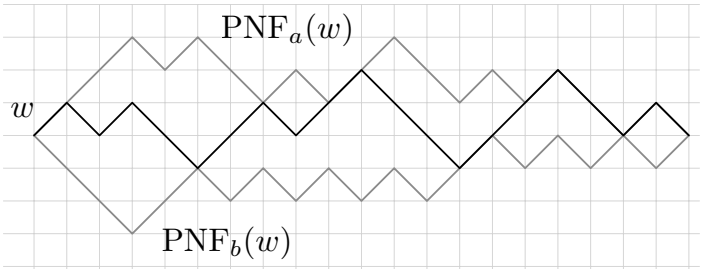

Figure 1.1: The word $w = ababbaabaabbbaaabbab$, its prefix normal forms $\mathrm{PNF}_a(w) = aaababbabaabbababbab$ and $\mathrm{PNF}_b(w) = bbbaababababaababababa$, and the region delineated by $\Pi(w)$, the Parikh set of w.

The language L_a is not context-free. The proof is an easy modification of the proof that Berstel and Boasson gave for the fact that the language of binary Lyndon words is not context-free [20].

Theorem 1.6. *L_a is not context-free.*

We now explore the relationship between the language L_a of prefix normal words and some known classes of words defined by means of lexicographic properties.

A *Lyndon word* is a word which is lexicographically (strictly) smaller than any of its proper non-empty suffixes. Equivalently, w is a Lyndon word if it is the (strictly) smallest, in the lexicographic order, among its conjugates, i.e., for any factorization $w = uv$, with u, v non-empty words, one has that the word vu is lexicographically greater than w [99]. Note that, by definition, a Lyndon word is primitive, i.e., it cannot be written as $w = u^k$ for a $u \in \Sigma^*$ and $k > 1$. Let us denote by *Lyn* the set of Lyndon words over Σ. One has that $Lyn \not\subseteq L_a$ and $L_a \not\subseteq Lyn$. For example, the word $w = abab$ belongs to L_a but is not a Lyndon word since it is not primitive. An example of Lyndon word which is not in prefix normal form is $w = aabbabaabbb$.

A power of a Lyndon word is called a *prime word* [92] or *necklace* (see [23] for more details and references on this definition).

Let us denote by *PL* the set of prefixes of powers of Lyndon words, also called sesquipowers (or fractional powers) of Lyndon words [42], or *preprime words* [92], or also *pre-necklaces* [117]. It is easy to see that *PL* is in fact the set of prefixes of Lyndon words plus the powers of the letter b.

Theorem 1.7. *Every prefix normal word is a pre-necklace. That is, $L_a \subset PL$.*

The languages L_a and *PL*, however, do not coincide. The shortest word in *PL* that does not belong to L_a is $w = aabbabaa$. Below we give the table of the number of words in L_a of each length, up to 16, compared with that of pre-necklaces. This latter sequence is listed in Neil Sloane's On-Line Encyclopedia of Integer Sequences [122].

The prefix normal form PNF_a induces an equivalence relation on Σ^*, namely $u \equiv_{\mathrm{PNF}_a} v$ if and only if $\mathrm{PNF}_a(u) = \mathrm{PNF}_a(v)$. In Table 1.3, we give all prefix normal words of length 4, and their equivalence classes.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$L_a \cap \Sigma^n$	2	3	5	8	14	23	41	70	125	218	395	697	1273	2279	4185	7568
$PL \cap \Sigma^n$	2	3	5	8	14	23	41	71	127	226	412	747	1377	2538	4720	8800

Table 1.2: The number of words in L_a and in PL for each length up to 16.

PNF_a	class	card.
$aaaa$	$\{aaaa\}$	1
$aaab$	$\{aaab, baaa\}$	2
$aaba$	$\{aaba, abaa\}$	2
$aabb$	$\{aabb, baab, bbaa\}$	3
$abab$	$\{abab, baba\}$	2
$abba$	$\{abba\}$	1
$abbb$	$\{abbb, babb, bbab, bbba\}$	4
$bbbb$	$\{bbbb\}$	1

Table 1.3: The classes of words of length 4 having the same prefix normal form.

An interesting question is how to characterize two words that have the same prefix normal form. The classes of this equivalence do not seem to follow regular patterns. For example, the words $aababaaba$, $aabbaaba$, $abaababa$, $abaabbaa$, $ababaaba$, $abababaa$ all have the same prefix normal form $aababaaba$, so that no simple statement about the lengths of the runs of the two letters seems to provide a characterization of the classes. This example also shows that the prefix normal form of a word w is in general more complicated that just a rotation of w or of its reversal (which is in fact the case for small lengths).

Recall that for word length $n \leq 16$, we listed the number of equivalence classes in Table 1.2. The sizes of the equivalence classes seem to exhibit an irregular behaviour. We report in Table 1.4, for each of the 70 equivalence classes for words of length 8, the prefix normal form and the number of words in the class. Furthermore, we report the cardinality of the largest class of words for each length up to 16 (Table 1.5).

Prefix normal words appeared naturally studying the Jumbled Pattern Matching Problem in the binary case. We next describe the problem and the results we obtained. For more details see [13, 32, 33].

The problem consists is answering the question whether a query Parikh vector q appears in a given text s (decision version), or where it occurs (occurrence version). An occurrence of q is defined as an occurrence of a substring t of s with Parikh vector q. The problem can be viewed as an approximate pattern matching problem: We are looking for an occurrence of a jumbled version of a query string t, i.e. for the occurrence of a substring t' which has the same Parikh vector. In the following, let n be the length of the text s, m the length of the query q (defined as the length of a string t with Parikh vector q), and σ the size of the alphabet.

The above problem (both decision and occurrence versions) can be solved with a simple

PNF_a	card.	PNF_a	card.	PNF_a	card.	PNF_a	card.
aaaaaaaa	1	*aaabaabb*	6	*aabababa*	6	*abababba*	2
aaaaaaab	2	*aaababaa*	2	*aabababb*	9	*abababbb*	4
aaaaaaba	2	*aaababab*	6	*aababbaa*	2	*ababbaba*	1
aaaaaabb	3	*aaababba*	4	*aababbab*	8	*ababbabb*	6
aaaaabaa	2	*aaababbb*	8	*aababbba*	4	*ababbbab*	4
aaaaabab	4	*aaabbaaa*	1	*aababbbb*	10	*abababba*	2
aaaaabba	2	*aaabbaab*	4	*aabbaabb*	3	*abababbb*	6
aaaaabbb	4	*aaabbaba*	2	*aabbabab*	4	*abbabbab*	2
aaaabaaa	2	*aaabbabb*	6	*aabbabba*	3	*abbabbba*	2
aaaabaab	4	*aaabbbaa*	2	*aabbabbb*	8	*abbabbbb*	5
aaaababa	3	*aaabbbab*	4	*aabbbaab*	2	*abbbabbb*	4
aaaababb	6	*aaabbbba*	2	*aabbbaba*	2	*abbbbabb*	3
aaaabbaa	2	*aaabbbbb*	6	*aabbbabb*	6	*abbbbbab*	2
aaaabbab	4	*aabaabaa*	1	*aabbbbaa*	1	*abbbbbba*	1
aaaabbba	2	*aabaabab*	4	*aabbbbab*	4	*abbbbbbb*	8
aaaabbbb	5	*aabaabba*	2	*aabbbbba*	2	*bbbbbbbb*	1
aaabaaab	2	*aabaabbb*	4	*aabbbbbb*	7		
aaabaaba	4	*aababaab*	2	*abababab*	2		

Table 1.4: The cardinalities of the 70 classes of words of length 8 having the same prefix normal form. There are 7 classes of length 1, 24 classes of length 2, 5 classes of length 3, 16 classes of length 4, 2 classes of length 5, 9 classes of length 6, 1 class of length 7, 4 classes of length 8, 1 class of length 9 and 1 class of length 10.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\max \lvert[w]\rvert$	1	2	3	4	5	6	8	10	12	18	24	30	40	60	80	111

Table 1.5: The maximum cardinality of a class of words having the same prefix normal form. Neil Sloane kindly added this sequence to his On-line Encyclopedia of Integer Sequences [123].

sliding window based algorithm, in $O(n)$ time and $O(\sigma)$ additional storage space. This is worst case optimal with respect to the case of one query. However, when we expect to search for many queries in the same string, the above approach leads to $O(Kn)$ runtime for K queries. To the best of our knowledge, no faster approach is known. This is in stark contrast to the classical exact pattern matching problem, where all *exact* occurrences of a query pattern of length m are sought in a text of length n. In that case, for one query, any naive approach leads to $O(nm)$ runtime, while quite involved ideas for preprocessing and searching are necessary to achieve an improved runtime of $O(n+m)$, as do the Knuth-Morris-Pratt [93], Boyer-Moore [26] and Boyer-Moore-type algorithms (see, e.g., [11, 82]). However, when many queries are expected, the text can be preprocessed to produce a data structure of size linear in n, such as a suffix tree, suffix array, or suffix automaton, which then allows to answer individual queries in time linear in the length of the pattern (see any textbook on string algorithms, e.g. [100, 124]).

Jumbled pattern matching has been used as a filtering step in approximate pattern matching

algorithms [87], but rarely considered in its own right.

The authors of [34] present an algorithm for finding all occurrences of a Parikh vector in a runlength encoded text. The algorithm's time complexity is $O(n' + \sigma)$, where n' is the length of the runlength encoding of s. Obviously, if the string is not runlength encoded, a preprocessing phase of time $O(n)$ has to be added. However, this may still be feasible if many queries are expected. To the best of our knowledge, this is the only algorithm that has been presented for the problem we treat here.

An efficient algorithm for computing all Parikh fingerprints of substrings of a given string was developed in [10]. Parikh fingerprints are Boolean vectors where the k'th entry is 1 if and only if a_k appears in the string. The algorithm involves storing a data point for each Parikh fingerprint, of which there are at most $O(n\sigma)$ many. This approach was adapted in [65] for Parikh vectors and applied to identifying all repeated Parikh vectors within a given length range; using it to search for queries of arbitrary length would imply using $\Omega(P(s))$ space, where $P(s)$ denotes the number of different Parikh vectors of substrings of s. This is not desirable, since, for arbitrary alphabets, there are non-trivial strings of any length with quadratic $P(s)$ [46].

In [32] we presented two novel algorithms which perform significantly better than the simple window algorithm, in the case where many queries arrive.

For the binary case, we presented an algorithm which answers *decision queries* in $O(1)$ time, using a data structure of size $O(n)$. The data structure is constructed in $\Theta(n^2)$ time.

For general alphabets, we presented an algorithm (that we called Jumping Algorithm) with expected sublinear runtime which uses $O(n)$ space to answer *occurrence queries*. We gave two different variants of the algorithm. The first one uses a very simple data structure (an inverted table) and answers queries in time $O(\sigma J \log(\frac{n}{J} + m))$, where J denotes the number of iterations of the main loop of the algorithm. We showed that the expected value of J for the case of random strings and patterns is $O(\frac{n}{\sqrt{m}\sqrt{\sigma \log \sigma}})$, yielding an expected runtime of $O(n(\frac{\sigma}{\log \sigma})^{1/2} \frac{\log m}{\sqrt{m}})$, per query

The second variant of the algorithm uses wavelet trees [79] and has query time $O(\sigma J)$, yielding an overall expected runtime of $O(n(\frac{\sigma}{\log \sigma})^{1/2} \frac{1}{\sqrt{m}})$, per query.

Our simulations on random strings and real biological strings confirm the sublinear behavior of the algorithms in practice. This is a significant improvement over the simple window algorithm w.r.t. expected runtime, both for a single query and repeated queries over one string.

The Jumping Algorithm is reminiscent of the Boyer-Moore-like approaches to the classical exact string matching problem [11, 26, 82].

In the following, we give a formal description of the problem and present a solution for the decision version in the binary case, which is deeply related to the prefix normal words. The Jumping Algorithm will not be described here and can be found in [32].

Given a finite ordered alphabet $\Sigma = \{a_1, \ldots, a_\sigma\}, a_1 \leq \ldots \leq a_\sigma$. Recall that for a string $s \in \Sigma^*$, $s = s_1 \cdots s_n$, the *Parikh vector* $p(s) = (p_1, \ldots, p_\sigma)$ of s defines the multiplicities of the characters in s, i.e. $p_i = |\{j \mid s_j = a_i\}|$, for $i = 1, \ldots, \sigma$. For a Parikh vector p, the *length* $|p|$ denotes the length of a string with Parikh vector p, i.e. $|p| = \sum_i p_i$. An *occurrence* of a Parikh vector p in s is an occurrence of a substring t with $p(t) = p$. (An occurrence of t is a pair of

positions $0 \leq i \leq j \leq n$, such that $s_i \cdots s_j = t$.) A Parikh vector that occurs in s is called a sub-Parikh vector of s. The prefix of length i is denoted $pr(i) = pr(i, s) = s_1 \cdots s_i$, and the Parikh vector of $pr(i)$ as $prv(i) = prv(i, s) = p(pr(i))$.

For two Parikh vectors $p, q \in \mathbb{N}^\sigma$, we define $p \leq q$ and $p + q$ component-wise: $p \leq q$ if and only if $p_i \leq q_i$ for all $i = 1, \ldots, \sigma$, and $p + q = u$ where $u_i = p_i + q_i$ for $i = 1, \ldots, \sigma$. Similarly, for $p \leq q$, we set $q - p = v$ where $v_i = q_i - p_i$ for $i = 1, \ldots, \sigma$.

> **Jumbled Pattern Matching (JPM).** Let $s \in \Sigma^*$ be given, $|s| = n$. For a Parikh vector $q \in \mathbb{N}^\sigma$ (the query), $|q| = m$, find all occurrences of q in s. The *decision version* of the problem is where we only want to know whether q occurs in s.

We assume that K many queries arrive over time, so some preprocessing may be worthwhile.

Note that for $K = 1$, both the decision version and the occurrence version can be solved worst-case optimally with a simple window algorithm, which moves a fixed size window of size m along string s. Maintain the Parikh vector c of the current window and a counter r which counts indices i such that $c_i \neq q_i$. Each sliding step costs either 0 or 2 update operations of c, and possibly one increment or decrement of r. This algorithm solves both the decision and occurrence problems and has running time $\Theta(n)$, using additional storage space $\Theta(\sigma)$.

Precomputing, sorting, and storing all sub-Parikh vectors of s would lead to $\Theta(n^2)$ storage space, since there are non-trivial strings with a quadratic number of Parikh vectors over arbitrary alphabets [46]. Such space usage is inacceptable in many applications.

For small queries, the problem can be solved exhaustively with a linear size indexing structure such as a suffix tree, which can be searched down to length $m = |q|$ (of the substrings), yielding a solution to the decision problem in time $O(\sigma^m)$. For finding occurrences, report all leaves in the subtrees below each match; this costs $O(M)$ time, where M is the number of occurrences of q in s. Constructing the suffix tree takes $O(n)$ time, so for $m = o(\log n)$, we get a total runtime of $O(n)$, since $M \leq n$ for any query q.

The algorithm for the decision problem in the binary case we presented in [32] makes use of the nice property of binary strings stated in Lemma 1.3.

Let $\Sigma = \{a, b\}$ and let $\mathrm{pmin}(m)$ (resp. $\mathrm{pmax}(m)$) denote the minimum (resp. maximum) number of a's in a substring of length m. Assume that the algorithm has access to the values $\mathrm{pmin}(m)$ and $\mathrm{pmax}(m)$ for $m = 1, \ldots, n$; then, when a query $q = (x, y)$ arrives, it answers YES if and only if $x \in [\mathrm{pmin}(x + y), \mathrm{pmax}(x + y)]$. The query time is $O(1)$.

The table of the values $\mathrm{pmin}(m)$ and $\mathrm{pmax}(m)$ can be easily computed in a preprocessing step in time $\Theta(n^2)$ by scanning the string with a window of size m, for each m. Alternatively, lazy computation of the table is feasible, since for any query q, only the entry $m = |q|$ is necessary. Therefore, it can be computed on the fly as queries arrive. Then, any query will take time $O(1)$ (if the appropriate entry has already been computed), or $O(n)$ (if it has not). After n queries of the latter kind, the table is completed, and all subsequent queries can be answered in $O(1)$ time. If we assume that the query lengths are uniformly distributed, then this can be viewed as a coupon collector problem where the coupon collector has to collect one copy of each length m.

Then the expected number of queries needed before having seen all n coupons is $nH_n \approx n \ln n$ (see e.g. [66]). The algorithm will have taken $O(n^2)$ time to answer these $n \ln n$ queries.

The assumption of the uniform length distribution may not be very realistic; however, even if it does not hold, we never take more time than $O(n^2 + K)$ for K many queries. Since any one query may take at most $O(n)$ time, our algorithm never performs worse than the simple window algorithm. Moreover, for those queries where the table entries have to be computed, we can even run the simple window algorithm itself and report all occurrences, as well. For all others, we only give decision answers, but in constant time.

The size of the data structure is $2n = O(n)$. The overall running time for either variant is $\Theta(K + n^2)$. As soon as the number of queries is $K = \omega(n)$, both variants outperform the simple window algorithm, whose running time is $\Theta(Kn)$.

Example 1. Let $s = ababbaabaabbbaaabbab$. Below, we give the table of pmin and pmax for s. This example shows that the locality of pmin and pmax is preserved only in adjacent levels. As an example, the value $\text{pmax}(3) = 3$ corresponds to the substring aaa appearing only at position 14, while $\text{pmax}(5) = 4$ corresponds to the substring $aabaa$ appearing only at position 6.

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
pmin	0	0	0	1	2	2	3	3	4	4	5	5	6	7	7	8	8	9	9	10
pmax	1	2	3	3	4	4	4	5	5	6	7	7	7	8	8	9	9	9	10	10

Chapter 2

Least Number of Palindromes in a Word

In recent years, there has been an increasing interest in the importance of palindromes in mathematics, theoretical computer science and theoretical physics. In particular, one is interested in infinite words containing arbitrarily long palindromes. This stems from their role in the modeling of *quasicrystals* in theoretical physics (see for instance [55,80]) and also diophantine approximation in number theory (e.g., see [1,2,3,4,31,76,115,116]).

In [62], X. Droubay, J. Justin and G. Pirillo observed that any finite word w of *length* $|w|$ contains at most $|w|+1$ distinct palindromes (including the empty word). Such words are 'rich' in palindromes, in the sense that they contain the maximum number of different palindromic factors. Accordingly, we say that a finite word w is *rich* if it contains exactly $|w|+1$ distinct palindromes, and we call an infinite word *rich* if all of its factors are rich. In an independent work, P. Ambrož, C. Frougny, Z. Masáková and E. Pelantová [9] have considered the same class of words, which they call *full words* (following earlier work of S. Brlek, S. Hamel, M. Nivat, and C. Reutenauer [27]). Since [62], there is an extensive number of papers devoted to the study of rich words and their generalizations (see [7,9,14,27,28,29,59,78]).

In [75] we considered the opposite question: What is the least number of palindromes which can occur in an infinite word subject to certain constraints? For an infinite word ω, the set $PAL(\omega)$ of palindromic factors of ω can be finite or infinite (cf. [27]). For instance, in case ω is a Sturmian word, then $PAL(\omega)$ contains two elements of each odd length and one element of each even length. In fact, this property characterizes Sturmian words (see [56] and [63]). In contrast, the *paperfolding word* P is an example of an aperiodic uniformly recurrent word containing a finite number of palindromes:

$$P = aabaabbaaabbabbaaabaabbbaabbabbaaaba\cdots$$

It is obtained as the limit of the sequence $(P_n)_{n\geq 0}$ defined recursively by $P_0 = a$ and $P_{n+1} = P_n a \hat{P}_n$ (for $n \geq 0$), where $\hat{P}_n$ is the word obtained from $\tilde{P}_n$ by exchanging a's and b's [6]. J.-P. Allouche showed that the paperfolding word contains exactly 29 palindromes, the longest of which has length 13.

It is easy to see that any uniformly recurrent word ω which contains an infinite number of palindromes must be *closed under reversal*, that is, for every factor $u = u_1u_2\cdots u_n$ of ω, its *reversal* $\tilde{u} = u_n\cdots u_2u_1$ is also a factor of ω. The converse is not true: In fact, J. Berstel, L. Boasson, O. Carton and I. Fagnot [85] exhibited various examples of uniformly recurrent words closed under reversal and containing a finite number of palindromes. The paperfolding word is not closed under reversal, since for example it contains the factor *aaaba* but not *abaaa*.

If X is a set consisting of infinite words, we set

$$MinPal(X) = \inf\{\#PAL(\omega) \mid \omega \in X\}.$$

We first showed that without restrictions on the cardinality of the alphabet, one has that $MinPal = 4$. That is, for

$$W = \{\omega \in A^{\mathbb{N}} \mid 0 < \#A < \infty\},$$

we have $MinPal(W) = 4$. If in addition one requires that the word be aperiodic, that is, for

$$W_{\text{ap}} = \{\omega \in A^{\mathbb{N}} \mid 0 < \#A < \infty \text{ and } \omega \text{ is aperiodic}\},$$

then $MinPal(W_{\text{ap}}) = 5$. If moreover one requires that the word must be closed under reversal, that is, for

$$W_{\text{cl}} = \{\omega \in A^{\mathbb{N}} \mid 0 < \#A < \infty \text{ and } \omega \text{ is closed under reversal}\},$$

then one still has $MinPal(W_{\text{cl}}) = 5$.

In the case of binary words we showed the following:

Theorem 2.1. *Let A be a set with $\#A = 2$. Then:*

1. *$MinPal(A^{\mathbb{N}}) = 9$, where $A^{\mathbb{N}}$ denotes the set of all infinite words on A.*
2. *$MinPal(A^{\mathbb{N}}_{ap}) = 11$, where $A^{\mathbb{N}}_{ap}$ denotes the set of all aperiodic words in $A^{\mathbb{N}}$.*
3. *$MinPal(A^{\mathbb{N}}_{cl}) = 13$, where $A^{\mathbb{N}}_{cl}$ denotes the set of all words in $A^{\mathbb{N}}$ closed under reversal.*
4. *$MinPal(A^{\mathbb{N}}_{ap/cl}) = 13$, where $A^{\mathbb{N}}_{ap/cl}$ denotes the set of all aperiodic words in $A^{\mathbb{N}}$ closed under reversal.*

After the publication of [75], we observed that if one considers only words that are fixed points of primitive substitutions, then the lower bounds above remain the same (except for the two last ones). Let us give examples realizing the bounds.

- $MinPal(W) = 4$. This is realized by the fixed point of $a \mapsto abc, b \mapsto abc, c \mapsto abc$:

$$abcabcabcabcabcabcabc\cdots$$

- $MinPal(W_{\text{ap}}) = 5$. This is realized by the fixed point of $a \mapsto abc, b \mapsto aab, c \mapsto ca$:

$$abcaabcaabcabcaabcaabc\cdots$$

For binary words, i.e., elements of $A^{\mathbb{N}}$, where A is a set with $\#A = 2$, we have:

- $MinPal(A^{\mathbb{N}}) = 9$. This is realized by the fixed point of $a \mapsto aababb, b \mapsto aababb$:

$$aababbaababbaababbaababb\cdots$$

- $MinPal(A^{\mathbb{N}}_{\text{ap}}) = 11$. This is realized by the fixed point of $a \mapsto aaababb, b \mapsto aababb$ (as it has been suggested to us by Mike Müller):

$$aaababbaaababbaaababbaababb\cdots$$

Finally, we have $MinPal(A_{\mathrm{cl}}^{\mathbb{N}}) = MinPal(A_{\mathrm{ap/cl}}^{\mathbb{N}}) = +\infty$, as a consequence of the following theorem of Tan [125]:

Theorem 2.2. *Let x be a fixed point of a primitive substitution over A. Then x is closed under reversal if and only if it contains infinitely many palindromes.*

Chapter 3

Bispecial Sturmian Words

A word w is *balanced* if and only if for any u, v factors of w of the same length, and for any letter a, one has $||u|_a - |v|_a| \leq 1$, where $|z|_a$ denotes the number of a's in the word z.

Balanced words appear in several problems in Computer Science. For example, Altman, Gaujal and Hordijk [8] proved that balanced words are optimal sequences for some classes of scheduling problems, such as routing among several systems. An interesting problem arising in this context is that of constructing infinite balanced words with assigned frequencies of letters. There is a conjecture of A. S. Fraenkel [77], originally stated in the context of Number Theory, that is equivalent to the following: for any fixed $k > 2$, there is only one bi-infinite balanced word (up to letter permutation) over an alphabet of size k, in which all letters have different frequencies, and this word is periodic. The Fraenkel Conjecture has been proved true for small alphabet sizes (see [15, 128] and references therein), but the general problem remains open.

For any alphabet Σ of size at least two, there exist infinite words over Σ that are balanced and aperiodic. When $|\Sigma| = 2$, these are called infinite Sturmian words. Sturmian words are very rich from the combinatorial point of view, and because of this fact they have a lot of equivalent definitions and characterizations (see, as a classical reference, [99, Chapter 2]). The only balanced infinite words that are aperiodic and have different letter frequencies are the infinite Sturmian words (see [83]).

A finite Sturmian word is a factor of some infinite Sturmian word. The set St of Sturmian words therefore coincides with the set of binary balanced finite words.

If one considers extendibility within the set St of Sturmian words, one can define left special Sturmian words (resp. right special Sturmian words) [60] as those words w over the alphabet $\Sigma = \{a, b\}$ such that aw and bw (resp. wa and wb) are both Sturmian words. For example, the word aab is left special since $aaab$ and $baab$ are both Sturmian words, but is not right special since $aabb$ is not a Sturmian word.

Left special Sturmian words are precisely the binary words having suffix automaton[1] with minimal state complexity (cf. [67, 121]). From combinatorial considerations one has that right special Sturmian words are the reversals of left special Sturmian words.

The Sturmian words that are both left and right special are called bispecial Sturmian words. They are of two kinds: strictly bispecial Sturmian words, that are the words w such that awa, awb, bwa and bwb are all Sturmian words (e.g. aa), or non-strictly bispecial Sturmian words otherwise (e.g. ab). Strictly bispecial Sturmian words are also called central words, and have been deeply studied (see for example [37, 60]) because they constitute the kernel of the theory of Sturmian words. Non-strictly bispecial Sturmian words, instead, received less attention.

One important field in which Sturmian words arise naturally is Discrete Geometry. Indeed, infinite Sturmian words can be viewed as the digital approximations of Euclidean straight lines in the plane. It is known that given a point (p, q) in the grid $\mathbb{Z} \times \mathbb{Z}$, with $p, q > 0$, there exists a unique path that approximates from below (resp. from above) the Euclidean segment joining the origin $(0, 0)$ to the point (p, q). If one encodes horizontal and vertical unitary segments with

[1] The suffix automaton of a finite word w is the minimal deterministic finite state automaton accepting the set of suffixes of w.

the letters a and b respectively, this path is called the lower (resp. upper) Christoffel word[2] associated to the pair (p, q), and is denoted by $w_{p,q}$ (resp. $w'_{p,q}$). By elementary geometrical considerations, one has that for any $p, q > 0$, $w_{p,q} = aub$ for some word u, and $w'_{p,q} = b\tilde{u}a$, where $\tilde{u}$ is the reversal of u. If (and only if) p and q are coprime, the Christoffel words $w_{p,q}$ and $w'_{p,q}$ are primitive (that is, they are not a concatenation of copies of a shorter word).

A well known result of Jean Berstel and Aldo de Luca [21] is that a word u is a strictly bispecial Sturmian word if and only if aub is a primitive lower Christoffel word (or, equivalently, if and only if bua is a primitive upper Christoffel word). We showed in [68] that this correspondence holds in general between bispecial Sturmian words and Christoffel words. More precisely, we proved that u is a bispecial Sturmian word if and only if there exist letters x, y in $\{a, b\}$ such that xuy is a Christoffel word.

This characterization allowed us to prove an enumerative formula for bispecial Sturmian words: there are exactly $2n + 2 - \phi(n + 2)$ bispecial Sturmian words of length n, where ϕ is the Euler totient function, i.e., $\phi(n)$ is the number of positive integers smaller than or equal to n and coprime with n. Surprisingly, enumerative formulae for left special, right special and strictly bispecial Sturmian words were known [60], but to the best of our knowledge we exhibited the first proof of an enumerative formula for non-strictly bispecial (and therefore for bispecial) Sturmian words.

We then investigated the minimal forbidden words for the set of finite Sturmian words. Recall that the set of minimal forbidden words (or minimal absent words) of a factorial language is the set of words of minimal length that do not belong to the language [107]. More precisely, given a factorial language L over an alphabet Σ, a word $v = v_1 v_2 \cdots v_n$, with $v_i \in \Sigma$, is a minimal forbidden word for L if $v_1 \cdots v_{n-1}$ and $v_2 \cdots v_n$ are in L, but v is not.

Minimal forbidden words represent a powerful tool to investigate the structure of a factorial language (see [16, 17, 50]), such as the language of factors of a (finite or infinite) word, or of a set of words. They also appear in different contexts in Computer Science, such as symbolic dynamics [17], data compression (where the set of minimal forbidden words is often called an antidictionary) [51], or bio-informatics (where they are also called minimal absent words) [41].

We gave a characterization of minimal forbidden words for the language St of Sturmian words in [68]. We showed that they are precisely the words of the form ywx such that xwy is a non-primitive Christoffel word, where $\{x, y\} = \{a, b\}$. This characterization allowed us to give an enumerative formula for the set of minimal forbidden words of St: there are exactly $2(n - 1 - \phi(n))$ minimal forbidden words of length n for every $n > 1$.

[2]Some authors require that p and q be coprime in the definition of Christoffel word. Here we follow the definition given in [21] and do not require this condition.

Chapter 4

Trapezoidal Words

Finite Sturmian words have *at most* $n+1$ factors of length n, for every $n \geq 0$. However, this property does not characterize them, as shown by the word $w = aaabab$, which is not Sturmian since the factors aaa and bab violate the condition of balancedness.

The finite words defined by the property of having at most $n+1$ factors of length n, for every $n \geq 0$, are called *trapezoidal words*. The name comes from the fact that the graph of the function counting the distinct factors of each length (usually called the factor complexity) defines, for trapezoidal words, an isosceles trapezoid.

Trapezoidal words were defined by de Luca [57] by means of combinatorial parameters on the repeated factors of finite words. For a finite word w over a finite alphabet Σ, let K_w be the minimal length of an unrepeated suffix of w and let R_w be the minimal length for which w does not contain right special factors, i.e., factors having occurrences followed by distinct letters. de Luca proved that for any word w of length $|w|$ one always has

$$R_w + K_w \leq |w| \tag{4.1}$$

and studied the case in which the equality holds. This is in fact the case in which w is trapezoidal, since one can prove that a word w is trapezoidal if and only if

$$R_w + K_w = |w|. \tag{4.2}$$

Then, de Luca proved that finite Sturmian words verify (4.2) and are therefore trapezoidal.

The non-Sturmian trapezoidal words were later characterized by D'Alessandro [53]. Since a non-Sturmian word is not balanced, it must contain a pair of factors having the same length and not verifying the condition of balancedness. Such a pair is called a *pathological pair*. Let w be a non-Sturmian word and (f, g) its pathological pair of minimal length (it can be proved that this is in fact unique). D'Alessandro proved that w is trapezoidal if and only if $w = pq$ for some $p \in \mathrm{Suff}(\{\tilde{z}_f\}^*)$ and $q \in \mathrm{Pref}(\{z_g\}^*)$, where $\tilde{z}_f$ is the reversal of the fractional root z_f of f and z_g is the fractional root of g. This characterization is quite involved and will be discussed in detail later.

The characterization by D'Alessandro allows us to give an enumerative formula for trapezoidal words. Indeed, it is known that the number of Sturmian words of length n is equal to

$$S(n) = 1 + \sum_{i=1}^{n} (n - i + 1)\phi(i)$$

(cf. [97,103]), where ϕ is the Euler totient function (see the previous chapter for the definition). In [30] we proved that the number of non-Sturmian trapezoidal words of length n is

$$T(n) = \sum_{i=0}^{\lfloor (n-4)/2 \rfloor} 2(n - 2i - 3)\phi(i + 2),$$

and thus the number of trapezoidal words of length n is $S(n) + T(n)$.

We then distinguish trapezoidal words into two distinct classes, accordingly with the defini-

tion below.

Definition 4.1. *Let w be a finite word over an alphabet Σ. We say that w is* closed *if it is empty or it has a factor (different from the word itself) occurring exactly twice in w, as a prefix and as a suffix, that is, with no internal occurrences. A word which is not closed is called* open.

For example, the words *aabbaa* and *ababa* are closed, whereas the word *aabbaaa* is open.

The notion of closed word is closely related to the concept of *complete return* to a factor u in a word w, as considered in [78]. A complete return to u in w is any factor of w having exactly two occurrences of u, one as a prefix and one as a suffix. Hence w is closed if and only if it is a complete return to one of its factors; such a factor is clearly both the longest repeated prefix and the longest repeated suffix of w.

The notion of closed word is also equivalent to that of *periodic-like* word [36]. A word w is periodic-like if and only if its longest repeated prefix does not appear in w followed by different letters, i.e., is not right special.

We derived some properties of open and closed trapezoidal words in [30]. We proved that the set of closed trapezoidal words is contained in that of Sturmian words. We then characterized the special factors of closed trapezoidal words by showing that every closed trapezoidal word w contains a bispecial factor u such that u is a central word and the left (resp. right) special factors of w are the prefixes (resp. suffixes) of u. We showed that trapezoidal palindromes (hence, Sturmian palindromes) are closed words, and that a closed trapezoidal word is a Sturmian palindrome if and only if its longest repeated prefix is a palindrome.

We introduced *semicentral words*, that are trapezoidal words in which the longest repeated prefix, the longest repeated suffix, the longest left special factor and the longest right special factor all coincide. We proved that a trapezoidal word w is semicentral if and only if it is of the form $w = uxyu$, for a central word u and two different letters x, y.

We then studied the prefixes of the Fibonacci infinite word f, and showed that the sequence of the numbers of consecutive prefixes of f that are closed (resp. open) is the sequence of Fibonacci numbers.

In [58], we extended this study to the set of characteristic Sturmian words, as described in the next section.

Chapter 5

The Array of Open and Closed Prefixes

In the previous section, we dealt with trapezoidal words, also with respect to the property of being *closed* (also known as periodic-like [36]) or open. Factors of Sturmian words are the most notable example of trapezoidal words, and we characterized the sequence of open and closed prefixes of the Fibonacci word F, a famous characteristic Sturmian word.

In [58] we investigated the sequence of open and closed prefixes of Sturmian words in general, and in particular in the standard case. More precisely, we proved that the sequence $oc(w)$ of open and closed prefixes of a word w (i.e., the sequence whose n-th element is 1 if the prefix of length n of w is closed, or 0 if it is open) characterizes every (finite or infinite) Sturmian word, up to isomorphisms of the alphabet.

In [30] we proved that the lengths of the runs (maximal subsequences of consecutive equal elements) in $oc(F)$ form the doubled Fibonacci sequence. We proved in [58] that this doubling property holds for every standard Sturmian word, and described the sequence $oc(w)$ of a standard Sturmian word w in terms of the *semicentral* prefixes of w, which are the prefixes of the form $u_n xy u_n$, where x, y are letters and $u_n xy$ is an element of the standard sequence of w. As a consequence, we showed that the word $ba^{-1}w$, obtained from a standard Sturmian word w starting with letter a by swapping its first letter, can be written as the infinite product of the words $(u_n^{-1}u_{n+1})^2$, $n \geq 0$. Since the words $u_n^{-1}u_{n+1}$ are reversals of standard words, this induces an infinite factorization of $ba^{-1}w$ in squares of reversed standard words.

For example, in the case of the Fibonacci infinite word f, we have the following factorization in squares of reversed Fibonacci finite words:

$$\begin{aligned} f &= ab \prod_{n>1} (\widetilde{f_n})^2 \\ &= ab \cdot (a \cdot a)(ba \cdot ba)(aba \cdot aba)(baaba \cdot baaba) \cdots \end{aligned}$$

Finally, we showed how the sequence of open and closed prefixes of a standard Sturmian word of slope α is related to the continued fraction expansion of α.

Chapter 6

Abelian Repetitions in Sturmian Words

The study of repetitions in words is a classical subject in Theoretical Computer Science both from the combinatorial and the algorithmic point of view. Repetitions are strictly related to the notion of periodicity. Recall that a word w of length $|w|$ has a *period* $p > 0$ if $w_i = w_{i+p}$ for any $1 \leqslant i \leqslant |w| - p$, where w_i is the symbol in position i of w. Every word w has a minimal period $p \leq |w|$. If $|w|/p \geq 2$, then w is called a *repetition* of period p and *exponent* $|w|/p$. When $|w|/p = k$ is an integer, the word w is called an *integer power*, since it can be written as $w = u^k$ for some $k > 1$, i.e., w is the concatenation of k copies of a word u of length p. If instead $|w|/p$ is not an integer, the word w is called a *fractional power*. In any case one can write $w = u^k u'$, where u' is the prefix of u such that $|w|/p = k + |u'|/|u|$. For example, the word $w = aabaaba$ is a 7/3-power since it has minimal period 3 and length 7. A classical reference on periodicity is [99, Chap. 8].

Abelian properties concerning words have been studied since the very beginning of Formal Languages and Combinatorics on Words. The notion of Parikh vector has become a standard and is often used without an explicit reference to the original 1966 Parikh's paper [110]. Abelian powers were first considered in 1961 by Erdös [64] as a natural generalization of usual powers. Research concerning abelian properties of words and languages developed afterwards in different directions. In particular, there is a recent increasing of interest on abelian properties of words linked to periodicity (see, for example, [12, 40, 61, 111, 113, 120]), and on the algorithmic search for abelian periodicities in strings [43, 49, 52, 71, 72, 73, 90, 94].

Recall that the Parikh vector $\mathcal{P}_w$ of a finite word w enumerates the cardinality of each letter of the alphabet in w. Therefore, two words have the same Parikh vector if and only if one can be obtained from the other by permuting letters.

An integer m is an abelian period of a word w if w can be written as $w = u_0 u_1 \cdots u_{j-1} u_j$ for words u_i and an integer $j \geq 2$, where for $0 < i < j$ all the u_i's have the same Parikh vector $\mathcal{P}$ whose sum of components is m and the Parikh vectors of u_0 and u_j are contained in $\mathcal{P}$ (see [47]). Every word w has a minimal abelian period $p \leq |w|$.

For example, the word $w = abcba$ has abelian period 5 (since one can set $u_0 = u_2 = \varepsilon$, where ε denotes the empty word, and $u_1 = w$) and 4 (for $u_0 = a$, $u_1 = bcba$ and $u_2 = \varepsilon$), but its minimal abelian period is 3 ($u_0 = ab$, $u_1 = cba$ and $u_2 = \varepsilon$).

We say that the word w is an *abelian repetition* of (abelian) period m and exponent $|w|/m$ if w can be written as $w = u_0 u_1 \cdots u_{j-1} u_j$, for an integer $j \geq 3$, where for $0 < i < j$ all the u_i's have the same Parikh vector $\mathcal{P}$ whose sum of components is m and the Parikh vectors of u_0 and u_j are contained in $\mathcal{P}$. Notice that the exponent of an abelian repetition is always greater than or equal to 2. When u_0 and u_j are both empty, w is called an *abelian power* (or also a *weak repetition* [52]).

For example, the word $w = abaab$ is an abelian repetition of period 2, since one can set $u_0 = a$, $u_1 = ba$, $u_2 = ab$ and $u_3 = \varepsilon$. The word $w = abcba$, instead, is not an abelian repetition.

It is well known that Sturmian words and Fibonacci words, in particular, are extremal cases for several problems related to repetitions (see for example [48, 84, 105]) and are worst-case examples for classical pattern matching algorithms, e.g. Knuth-Morris-Pratt [5, 95].

There exists a huge bibliography concerning Sturmian words (see for instance the survey papers [19, 22], [99, Chap. 2], [112, Chap. 6] and references therein). In particular, there is a vaste bibliography concerning the maximal exponent of classical repetitions in Sturmian words (see for example [18, 24, 35, 54, 88, 96, 127] and references therein) which stems from the seminal work on the Fibonacci word presented in [104].

In [102], a bijection between factors of Sturmian words and subintervals of the unitary segment is described. We showed in [70] that this bijection preserves abelian properties of factors.

In the same paper, we proved that a Sturmian word of slope α contains an abelian power of period m and exponent $k \geq 2$ if and only if $\{m\alpha\} < \frac{1}{k}$ or $\{-m\alpha\} < \frac{1}{k}$, where $\{x\}$ is the fractional part of x. As a consequence, we have that the maximal exponent of an abelian power of period m in a Sturmian word of slope α is equal to the largest integer k such that $\min(\{m\alpha\}, \{-m\alpha\}) < 1/k$. Furthermore, we proved that the Sturmian word $s_{\alpha,\rho}$ with slope α and intercept ρ contains an abelian power of period m and exponent $k \geq 2$ starting at position n if and only if $\{\rho + n\alpha\} < 1 - k\{m\alpha\}$ or $\{\rho + n\alpha\} > k\{-m\alpha\}$.

If k_m (resp. k'_m) denotes the maximal exponent of an abelian power (resp. repetition) of period m, we proved that $\limsup k_m/m = \limsup k'_m/m \geq \sqrt{5}$ for any Sturmian word, and the equality holds for the Fibonacci word. We further proved that the above superior limits are finite if and only if the development in continued fraction of α has bounded partial quotients, if and only if s_α is β-power free for some real number β.

We then focused on the Fibonacci word. We proved that the maximal exponent of an abelian power of period F_j—the jth Fibonacci number—in the Fibonacci word is equal to $\lfloor \phi F_j + F_{j-1} \rfloor$, and that for every F_j, $j > 1$, the longest prefix of the Fibonacci word that is an abelian repetition of period F_j has length $F_j(F_{j+1}+F_{j-1}+1)-2$ if j is even or $F_j(F_{j+1}+F_{j-1})-2$ if j is odd. These results allowed us to give an exact formula for the smallest abelian periods of the Fibonacci finite words. More precisely we proved that, for $j \geq 3$, the Fibonacci word f_j, of length F_j, has abelian period equal to $F_{\lfloor j/2 \rfloor}$ if $j = 0, 1, 2 \mod 4$, or to $F_{1+\lfloor j/2 \rfloor}$ if $j = 3 \mod 4$.

Chapter 7

Universal Lyndon Words

A word is called Lyndon if it is lexicographically smaller than all of its conjugate words (other than itself). Lyndon words are an important and well studied object in Combinatorics. Recall, for example, the fact that every Lyndon word is unbordered, or the existence of a unique factorization of any word into a non-decreasing sequence of Lyndon words [98]. The definition of Lyndon word implicitly assumes a lexicographic order. Therefore, for different orders, we typically obtain several distinct Lyndon conjugates of the same word. The motivation of this research, published in [38], is to push the idea to its limits, and ask whether there is a *universal Lyndon word*, that is, a word of length $n!$ over n letters such that for each of its conjugates there exists an order with respect to which this conjugate is Lyndon.

Such a word resembles similar objects known in the literature as universal cycles. A *universal cycle* [44] is a circular word containing every object of a particular type exactly once as a factor. Probably the most prominent example of universal cycles are de Bruijn cycles, which are circular words of length 2^n containing every binary word of length n exactly once.

The set represented by a universal Lyndon word is the set of all total orders on n letters or, equivalently, all permutations of n letters. The most convenient way is to represent the order $a_1 < a_2 < \cdots < a_n$ by its "shorthand encoding", which is the word $a_1 a_2 \cdots a_{n-1}$. Jackson [86] showed that the corresponding universal cycles exist for every n and can be obtained from an Eulerian graph in a manner similar to the generation of de Bruijn cycles. Ruskey and Williams [118] gave efficient algorithms for constructing shorthand universal cycles for permutations. The results presented in this chapter can be seen as a generalization of this concept. Indeed, it is easy to note that every shorthand universal cycle for permutations is a universal Lyndon word (see [81] for more details), but the opposite is not true—that is, there exist universal Lyndon words such that the Lyndon conjugate for some order $a_1 < a_2 < \cdots < a_n$ does not start with $a_1 a_2 \cdots a_{n-1}$.

We studied the structural properties of universal Lyndon words and gave combinatorial characterizations. We then developed a method for generating all the universal Lyndon words. This method is based on the notion of Hamiltonian lex-code.

In what follows, we let Σ_n denote the alphabet $\{1, 2, \ldots, n\}$, $n > 0$.

Definition 7.1. *A universal Lyndon word (ULW) of degree n is a word over Σ_n that has length $n!$ and such that all its conjugates are Lyndon words.*

We consider universal Lyndon words up to rotation, i.e., as cyclic words.

Example 7.2. *The only universal Lyndon word of degree* 1 *is* 1*, and the only universal Lyndon word of degree* 2 *is* 12*. There are three universal Lyndon words of degree* 3*, namely* 212313*,* 323121 *and* 131232*. Note that these words are pairwise isomorphic (i.e., one can be obtained from another by renaming letters). There are* 492 *universal Lyndon words of degree* 4*. There are* 41 *if we consider them up to isomorphism.*

The following proposition gives a sufficient condition for a word being a ULW. Recall that a cyclic factor of a word w is a factor of w when considered as a circular word.

Proposition 7.1. *Let $n \geq 2$, and w be a word over Σ_n such that every permutation of $n-1$ elements of Σ_n appears as a cyclic factor in w exactly once. Then w is a universal Lyndon word.*

A consequence of Proposition 7.1 is that we can use Jackson graphs [1] to show that there exist universal Lyndon words for each degree.

Proposition 7.2. *There exist universal Lyndon words of degree n for every $n > 0$.*

A universal Lyndon word that is an Eulerian cycle of a Jackson graph is called a *universal cycle* [86], or *shorthand universal cycle for permutations* [118], but here we will call it a *universal Lyndon word of Jackson type*, or simply a *Jackson universal Lyndon word*.

However, there are universal Lyndon words that are not of Jackson type. In fact, the converse of Proposition 7.1 is not true. For instance, $w = 123431242314132421343214$ is a universal Lyndon word of degree 4 but it does not contain any of 142, 143, 241, 243, 341, 342 as a factor.

We now give a combinatorial characterization of universal Lyndon words.

Theorem 7.3. *Let w be a word over Σ_n. Then w is a universal Lyndon word if and only if for every cyclic factor u of w, one has*

$$|w|_u^c = (n - |\mathrm{Alph}(u)|)! \tag{7.1}$$

where $|w|_u^c$ is the number of occurrences of u as a cyclic factor of w.

Corollary 7.4. *The reversal of a ULW is a ULW.*

Note that the fact that the set of universal Lyndon words is closed under reversal is not an immediate consequence of the definition. This property is not true for Lyndon words, e.g. the word 112212 is Lyndon but its reversal is not.

We now exhibit a method for constructing all the universal Lyndon words. This method is based on particular prefix codes, whose definition is given below.

Definition 7.3. *A set $X \subseteq \Sigma_n^*$ is a* lex-code *of degree n if:*

1. *for any $x \in X$, there exists a unique ordering of Σ_n such that x is the lexicographical minimum of X;*
2. *if u is a proper prefix of some word of X, then u is a prefix of at least two distinct words of X.*

A lex-code X of degree n is Hamiltonian *if the relation*

$$S_X = \{(x, y) \in X \times X \mid \exists a \in \Sigma, x \text{ is a prefix of } ay\}$$

has a Hamiltonian digraph.

[1] The *Jackson graph of degree n* is a directed graph in which the nodes are the words over Σ_n that are permutations of $n-2$ letters, and there is an edge from node u to node v if and only if the suffix of length $n-3$ of u is equal to the prefix of length $n-3$ of v and the first letter of u is different from the last letter of v [86]

Notice that Condition 1 in the previous definition ensures that a lex-code is a prefix code.

Let w be a universal Lyndon word. We let $MT(w)$ denote the minimal total order-defining words of w, i.e., the set of cyclic factors of w that are minimal order-defining words with respect to a total order on Σ_n. The next proposition is a direct consequence of the definitions and of the previous results.

Proposition 7.5. *Let w be a universal Lyndon word of degree n, and u a cyclic factor of w. The following conditions are equivalent:*

1. *$u \in MT(w)$;*
2. *$|\mathrm{Alph}(u)| = n-1$, and $|\mathrm{Alph}(u')| < n-1$ for each proper prefix u' of u;*
3. *there exists a unique conjugate w_i of w such that u is the shortest unrepeated prefix of w_i.*

The following theorem shows the relationships between Hamiltonian lex-codes and universal Lyndon words.

Theorem 7.6. *Let w be a ULW. Then the set $MT(w)$ is a Hamiltonian lex-code. Conversely, if $X \subseteq \Sigma_n^*$ is a Hamiltonian lex-code, then there exists a ULW w such that $X = MT(w)$.*

From Theorem 7.6, in order to produce a ULW, one can construct a lex-code and check whether it is Hamiltonian. Let S_n be the set of the total orders on Σ_n. All lex-codes of degree n can be obtained by a construction based on iterated refinements of a partition of S_n as follows:

1. set $X = \{\varepsilon\}$ and $C_\varepsilon = S_n$;
2. repeat the following steps until C_x is a singleton for all $x \in X$:
   (a) select $x \in X$ such that C_x contains at least two elements;
   (b) choose $\Gamma \subseteq \Sigma_n$;
   (c) for any $a \in \Gamma$, let C_{xa} be the set of the orders of C_x such that $a = \min \Gamma$;
   (d) replace X by $(X \setminus \{x\}) \cup \{xa \mid a \in \Gamma,\ C_{xa} \neq \emptyset\}$.

Chapter 8

Cyclic Complexity

The usual notion of complexity of a discrete system counts the number of distinct patterns of the same size that the system can generate. In the case of sequences (words), this is the number of distinct blocks (factors) of each given length. This measure of complexity is usually called *factor complexity* (or *block complexity*). The words with the "simplest" structure are the periodic ones. They are of the form $x = u^\omega$ (i.e., an infinite concatenation of a same finite block u) called *purely periodic*, or of the form $x = vu^\omega$, called *ultimately periodic*. The non-periodic words are called *aperiodic*. The factor complexity distinguishes between periodic and aperiodic words. In fact, the theorem of Morse and Hedlund [108] states that a word is aperiodic if and only if it has at least $n + 1$ factors of length n for every n. From this result, it is natural to consider those aperiodic words which have minimal factor complexity, i.e., those having exactly $n + 1$ distinct factors of length n for every n, that we know are called *Sturmian words*.

There exist many other measures of complexity of words in literature. For example, a lot of attention has recently been given (see for instance [25, 101, 113, 119, 126]) to the *abelian complexity*, which is the function counting the number of factors of each given length up to permutations. Other new measures of complexity of words have been introduced over the time, which are intermediate between factor and abelian complexity (e.g. maximal pattern complexity [89], k-abelian complexity [91], binomial complexity [114]) or involve different definitions that appear naturally in the study of sequences (e.g. periodicity complexity [106], minimal forbidden factor complexity [107], palindromic complexity [27], etc.) For most of these measures, Sturmian words are those aperiodic words of lowest complexity. However, they do not distinguish between two Sturmian words having different slopes.

In [39] we proposed a new measure of complexity, *cyclic complexity*, which consists in counting the factors of each given length of a word up to conjugacy. The notion of conjugacy is a basic notion in Combinatorics on Words. Two words are said conjugate if they are equal when read on a circle[1]. That is, the cyclic complexity of a word is the function counting the number of conjugacy classes of factors of each given length.

One of the main results we obtained is that cyclic complexity distinguishes between periodic and aperiodic words. In fact, we proved the following theorem.

Theorem 8.1. *A word is ultimately periodic if and only if it has bounded cyclic complexity.*

That is, the Morse-Hedlund theorem can be extended to the setting of cyclic complexity. Note that a word is (purely) periodic if and only if there exists an integer n such that all the factors of length n are conjugate. Therefore, the minimum value that the cyclic complexity of an aperiodic word can take is 2. Sturmian words have the property that the cyclic complexity takes value 2 infinitely often.

Since the Sturmian words are characterized by having $n + 1$ factors of length n for every n, the factor complexity does not distinguish between two Sturmian words with different languages of factors. In contrast, for cyclic complexity, two Sturmian words with different languages of factors have different cyclic complexity. Indeed, we proved something stronger:

[1]More formally, u and v are conjugate if and only if one can write $u = w_1w_2$ and $v = w_2w_1$ for some words w_1, w_2.

Theorem 8.2. *Let x be a Sturmian word. If a word y has the same cyclic complexity as x then, up to renaming letters, y is a Sturmian word having the same slope of x.*

That is, not only two Sturmian words with different languages of factors cannot have the same cyclic complexity, but the only words which have the same cyclic complexity of a Sturmian word x are those Sturmian words with the same slope of x.

These two results, whose proofs are not trivial, suggest that cyclic complexity can be considered as an interesting refinement of the classical notion of factor complexity and can open new perspectives in the study of complexity of discrete systems.

Finally, note that factor complexity, abelian complexity and cyclic complexity can all be viewed as actions of different subgroups of the symmetric group on the indices of a finite word (respectively, the trivial subgroup, the whole symmetric group and the cyclic subgroup). Since factor and abelian complexity are very well studied, looking at other subgroups of the symmetric group seems a very natural way of investigation.

Bibliography

[1] B. Adamczewski and J.-P. Allouche. Reversals and palindromes in continued fractions. *Theoret. Comput. Sci.*, 380(3):220–237, 2007.

[2] B. Adamczewski and Y. Bugeaud. On the Littlewood conjecture in simultaneous Diophantine approximation. *J. London Math. Soc. (2)*, 73(2):355–366, 2006.

[3] B. Adamczewski and Y. Bugeaud. Palindromic continued fractions. *Ann. Inst. Fourier (Grenoble)*, 57(5):1557–1574, 2007.

[4] B. Adamczewski and Y. Bugeaud. Transcendence measures for continued fractions involving repetitive or symmetric patterns. *J. Eur. Math. Soc. (JEMS)*, 12(4):883–914, 2010.

[5] A.V. Aho. Algorithms for Finding Patterns in Strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 257–300. Elsevier Science Publishers B. V., Amsterdam, the Netherlands, 1990.

[6] J.-P. Allouche. Schrödinger operators with rudin-shapiro potentials are not palindromic. *Journal of Mathematical Physics*, 38(4):1843–1848, 1997.

[7] J.-P. Allouche, M. Baake, J. Cassaigne, and David Damanik. Palindrome complexity. *Theoret. Comput. Sci.*, 292(1):9–31, 2003. Selected papers in honor of Jean Berstel.

[8] E. Altman, B. Gaujal, and A. Hordijk. Balanced sequences and optimal routing. *J. ACM*, 47(4):752–775, 2000.

[9] P. Ambrož, Z. Masáková, E. Pelantová, and C. Frougny. Palindromic complexity of infinite words associated with simple Parry numbers. *Ann. Inst. Fourier (Grenoble)*, 56(7):2131–2160, 2006. Numération, pavages, substitutions.

[10] A. Amir, A. Apostolico, G. M. Landau, and G. Satta. Efficient text fingerprinting via Parikh mapping. *J. Discrete Algorithms*, 1(5-6):409–421, 2003.

[11] A. Apostolico and R. Giancarlo. The Boyer-Moore-Galil string searching strategies revisited. *SIAM J. Comput.*, 15(1):98–105, 1986.

[12] S.V. Avgustinovich, J. Karhumäki, and S. Puzynina. On abelian versions of Critical Factorization Theorem. *RAIRO Theor. Inform. Appl.*, 46:3–15, 2012.

[13] G. Badkobeh, G. Fici, S. Kroon, and Zs. Lipták. Binary jumbled string matching for highly run-length compressible texts. *Inf. Process. Lett.*, 113(17):604–608, 2013.

[14] P. Baláži, Z. Masáková, and E. Pelantová. Factor versus palindromic complexity of uniformly recurrent infinite words. *Theoret. Comput. Sci.*, 380(3):266–275, 2007.

[15] J. Bark and P.P. Varjú. Partitioning the positive integers to seven Beatty sequences. *Indag. Math.*, 14(2):149 – 161, 2003.

[16] M.-P. Béal, F. Mignosi, and A. Restivo. Minimal Forbidden Words and Symbolic Dynamics. In *13th Annual Symposium on Theoretical Aspects of Computer Science, STACS '96*, volume 1046 of *Lecture Notes in Comput. Sci.*, pages 555–566. Springer, 1996.

[17] M.-P. Béal, F. Mignosi, A. Restivo, and M. Sciortino. Forbidden words in symbolic dynamics. *Adv. in Appl. Math.*, 25(2):163–193, 2000.

[18] J. Berstel. On the index of Sturmian words. In J. Karhumäki, H. Maurer, G. Păun, and G. Rozenberg, editors, *Jewels are Forever*, pages 287–294. Springer Berlin Heidelberg, 1999.

[19] J. Berstel. Sturmian and episturmian words (a survey of some recent results). In S. Bozapalidis and G. Rohonis, editors, *CAI 2007*, volume 4728 of *Lecture Notes in Comput. Sci.*, pages 23–47. Springer, 2007.

[20] J. Berstel and L. Boasson. The set of Lyndon words is not context-free. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 63:139–140, 1997.

[21] J. Berstel and A. de Luca. Sturmian Words, Lyndon Words and Trees. *Theoret. Comput. Sci.*, 178(1-2):171–203, 1997.

[22] J. Berstel, A. Lauve, C. Reutenauer, and F. Saliola. *Combinatorics on Words: Christoffel Words and Repetition in Words*, volume 27 of *CRM monograph series*. American Mathematical Society, 2008.

[23] J. Berstel and D. Perrin. The origins of combinatorics on words. *Eur. J. Comb.*, 28:996–1022, 2007.

[24] V. Berthé, C. Holton, and L. Q. Zamboni. Initial powers of Sturmian sequences. *Acta Arith.*, pages 315–347, 2006.

[25] F. Blanchet-Sadri and N. Fox. On the asymptotic abelian complexity of morphic words. In *Developments in Language Theory, 17th International Conference, DLT 2013*, volume 7907 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2013.

[26] R.S. Boyer and J.S. Moore. A fast string searching algorithm. *Commun. ACM*, 20(10):762–772, 1977.

[27] S. Brlek, S. Hamel, M. Nivat, and C. Reutenauer. On the palindromic complexity of infinite words. *Internat. J. Found. Comput. Sci.*, 15:293–306, 2004.

[28] M. Bucci, A. De Luca, A. Glen, and L. Q. Zamboni. A connection between palindromic and factor complexity using return words. *Adv. in Appl. Math.*, 42(1):60–74, 2009.

[29] M. Bucci, A. De Luca, A. Glen, and L. Q. Zamboni. A new characteristic property of rich words. *Theoret. Comput. Sci.*, 410(30-32):2860–2863, 2009.

[30] M. Bucci, A. De Luca, and G. Fici. Enumeration and Structure of Trapezoidal Words. *Theoretical Computer Science*, 468:12–22, 2013.

[31] Y. Bugeaud and M. Laurent. Exponents of Diophantine approximation and Sturmian continued fractions. *Ann. Inst. Fourier (Grenoble)*, 55(3):773–804, 2005.

[32] P. Burcsi, F. Cicalese, G. Fici, and Zs. Lipták. Algorithms for Jumbled Pattern Matching in Strings. *Int. J. Found. Comput. Sci.*, 23:357–374, 2012.

[33] P. Burcsi, F. Cicalese, G. Fici, and Zs. Lipták. On approximate jumbled pattern matching in strings. *Theory Comput. Syst.*, 50(1):35–51, 2012.

[34] A. Butman, R. Eres, and G. M. Landau. Scaled and permuted string matching. *Inf. Process. Lett.*, 92(6):293–297, 2004.

[35] A. Carpi and A. de Luca. Special factors, periodicity, and an application to Sturmian words. *Acta Inform.*, 36(12):983–1006, 2000.

[36] A. Carpi and A. de Luca. Periodic-like words, periodicity and boxes. *Acta Inform.*, 37:597–618, 2001.

[37] A. Carpi and A. de Luca. Central Sturmian Words: Recent Developments. In *Developments in Language Theory, 9th International Conference, DLT 2005*, volume 3572 of *Lecture Notes in Comput. Sci.*, pages 36–56. Springer, 2005.

[38] A. Carpi, G. Fici, Š. Holub, J. Opršal, and M. Sciortino. Universal Lyndon Words. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science*, volume 8634 of *Lecture Notes in Computer Science*, pages 135–146. Springer Verlag Berlin Heidelberg, 2014.

[39] J. Cassaigne, G. Fici, M. Sciortino, and L. Q. Zamboni. Cyclic Complexity of Words. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science*, volume 8634 of *Lecture Notes in Computer Science*, pages 159–170. Springer Verlag Berlin Heidelberg, 2014.

[40] J. Cassaigne, G. Richomme, K. Saari, and L.Q. Zamboni. Avoiding Abelian powers in binary words with bounded Abelian complexity. *Int. J. Found. Comput. Sci.*, 22(4):905–920, 2011.

[41] S. Chairungsee and M. Crochemore. Using minimal absent words to build phylogeny. *Theoret. Comput. Sci.*, 450:109–116, 2012.

[42] J.M. Champarnaud, G. Hansel, and D. Perrin. Unavoidable sets of constant length. *Internat. J. Algebra Comput.*, 14:241–251, 2004.

[43] M. Christou, M. Crochemore, and C. S. Iliopoulos. Identifying all abelian periods of a string in quadratic time and relevant problems. *Int. J. Found. Comput. Sci.*, 23(6):1371–1384, 2012.

[44] F. Chung, P. Diaconis, and R. Graham. Universal cycles for combinatorial structures. *Discrete Mathematics*, 110:43 – 59, 1992.

[45] F. Cicalese, G. Fici, and Zs. Lipták. Searching for jumbled patterns in strings. In *Proc. of the Prague Stringology Conference 2009 (PSC 2009)*, pages 105–117. Czech Technical University in Prague, 2009.

[46] M. Cieliebak, T. Erlebach, Zs. Lipták, J. Stoye, and E. Welzl. Algorithmic complexity of protein identification: combinatorics of weighted strings. *Discrete Applied Mathematics*, 137(1):27–46, 2004.

[47] S. Constantinescu and L. Ilie. Fine and Wilf's theorem for abelian periods. *Bulletin of the European Association for Theoretical Computer Science*, 89:167–170, 2006.

[48] M. Crochemore, L. Ilie, and W. Rytter. Repetitions in strings: Algorithms and combinatorics. *Theoret. Comput. Sci.*, 410(50):5227–5235, 2009.

[49] M. Crochemore, C. S. Iliopoulos, T. Kociumaka, M. Kubica, J. Pachocki, J. Radoszewski, W. Rytter, Wo. Tyczynski, and T. Walen. A note on efficient computation of all abelian periods in a string. *Inf. Process. Lett.*, 113(3):74–77, 2013.

[50] M. Crochemore, F. Mignosi, and A. Restivo. Minimal forbidden words and factor automata. In *Mathematical Foundations of Computer Science, 23rd International Symposium, MFCS '98*, volume 1450 of *Lecture Notes in Comput. Sci.*, pages 665–673. Springer, 1998.

[51] M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi. Text compression using antidictionaries. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 261–270. Springer, 1999.

[52] L. J. Cummings and W. F. Smyth. Weak repetitions in strings. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 24:33–48, 1997.

[53] F. D'Alessandro. A combinatorial problem on Trapezoidal words. *Theoret. Comput. Sci.*, 273(1-2):11–33, 2002.

[54] D. Damanik and D. Lenz. The index of Sturmian sequences. *European J. Combin.*, 23(1):23–29, 2002.

[55] D. Damanik and L.Q. Zamboni. Combinatorial properties of Arnoux-Rauzy subshifts and applications to Schrödinger operators. *Rev. Math. Phys.*, 15:745–763, 2003.

[56] A. de Luca. Sturmian words: structure, combinatorics, and their arithmetics. *Theoret. Comput. Sci.*, 183(1):45–82, 1997.

[57] A. de Luca. On the combinatorics of finite words. *Theoret. Comput. Sci.*, 218:13–39, 1999.

[58] A. De Luca and G. Fici. Open and Closed Prefixes of Sturmian Words. In *Proceedings of the 9th International Conference on Words*, volume 8079 of *Lecture Notes in Computer Science*, pages 132–142. Springer Berlin Heidelberg, 2013.

[59] A. de Luca, A. Glen, and L. Q. Zamboni. Rich, Sturmian, and trapezoidal words. *Theoret. Comput. Sci.*, 407(1-3):569–573, 2008.

[60] A. de Luca and F. Mignosi. Some combinatorial properties of Sturmian words. *Theoret. Comput. Sci.*, 136(2):361–385, 1994.

[61] M. Domaratzki and N. Rampersad. Abelian primitive words. *Int. J. Found. Comput. Sci.*, 23(5):1021–1034, 2012.

[62] X. Droubay, J. Justin, and G. Pirillo. Episturmian words and some constructions of de Luca and Rauzy. *Theoret. Comput. Sci.*, 255(1-2):539–553, 2001.

[63] X. Droubay and G. Pirillo. Palindromes and Sturmian words. *Theoret. Comput. Sci.*, 223(1-2):73–85, 1999.

[64] P. Erdös. Some unsolved problems. *Magyar Tud. Akad. Mat. Kutato. Int. Kozl.*, 6:221–254, 1961.

[65] R. Eres, G. M. Landau, and L. Parida. Permutation pattern discovery in biosequences. *Journal of Computational Biology*, 11(6):1050–1060, 2004.

[66] W. Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 1968.

[67] G. Fici. Special Factors and the Combinatorics of Suffix and Factor Automata. *Theoret. Comput. Sci.*, 412(29):3604–3615, 2011.

[68] G. Fici. On the Structure of Bispecial Sturmian Words. *Journal of Computer and System Sciences*, 80(4):711–719, 2014.

[69] G. Fici, A. Langiu, T. Lecroq, A. Lefebvre, F. Mignosi, and É. Prieur-Gaston. Abelian Repetitions in Sturmian Words. In O. Carton and M.-P. Béal, editors, *Proceedings of DLT 2013*, volume 7907 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2013.

[70] G. Fici, A. Langiu, T. Lecroq, A. Lefebvre, F. Mignosi, and É. Prieur-Gaston. Abelian Repetitions in Sturmian Words. In *Proceedings of the 17th International Conference on Developments in Language Theory*, volume 7907 of *Lecture Notes in Computer Science*, pages 227–238. Springer Berlin Heidelberg, 2013.

[71] G. Fici, T. Lecroq, A. Lefebvre, and É. Prieur-Gaston. Computing Abelian Periods in Words. In J. Holub and J. Žd'árek, editors, *Proceedings of the Prague Stringology Conference 2011*, pages 184–196. Czech Technical University in Prague, 2011.

[72] G. Fici, T. Lecroq, A. Lefebvre, and É. Prieur-Gaston. Algorithms for Computing Abelian Periods of Words. *Discrete Applied Mathematics*, 163:287–297, 2014.

[73] G. Fici, T. Lecroq, A. Lefebvre, É. Prieur-Gaston, and W. F. Smyth. Quasi-Linear Time Computation of the Abelian Periods of a Word. In J. Holub and M. Bálik, editors, *Proceedings of the Prague Stringology Conference 2012*, pages 103–110. Czech Technical University in Prague, 2012.

[74] G. Fici and Zs. Lipták. On prefix normal words. In *Proc. of the 15th Intern. Conf. on Developments in Language Theory (DLT 2011)*, volume 6795 of *LNCS*, pages 228–238. Springer, 2011.

[75] G. Fici and L. Q. Zamboni. On the least number of palindromes contained in an infinite word. *Theoretical Computer Science*, 481:1–8, 2013.

[76] S. Fischler. Palindromic prefixes and Diophantine approximation. *Monatsh. Math.*, 151(1):11–37, 2007.

[77] A. S. Fraenkel. Complementing and exactly covering sequences. *J. Comb. Theory, Ser. A*, 14(1):8–20, 1973.

[78] A. Glen, J. Justin, S. Widmer, and L. Q. Zamboni. Palindromic richness. *European J. Combin.*, 30:510–531, 2009.

[79] R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–850. ACM/SIAM, 2003.

[80] A. Hof, O. Knill, and B. Simon. Singular continuous spectrum for palindromic schrdinger operators. *Communications in Mathematical Physics*, 174(1):149–159, 1995.

[81] A. E. Holroyd, F. Ruskey, and A. Williams. Shorthand universal cycles for permutations. *Algorithmica*, 64(2):215–245, 2012.

[82] R. Nigel Horspool. Practical fast searching in strings. *Softw., Pract. Exper.*, 10(6):501–506, 1980.

[83] P. Hubert. Suites équilibrées. *Theoret. Comput. Sci.*, 242(1-2):91–108, 2000.

[84] C. S. Iliopoulos, D. Moore, and W. F. Smyth. A Characterization of the Squares in a Fibonacci String. *Theoret. Comput. Sci.*, 172(1-2):281–291, 1997.

[85] O. Carton J. Berstel, L. Boasson and I. Fagnot. Infinite words without palindrome. *Technical Report, CoRR abs/0903.2382*, 2009.

[86] B. W. Jackson. Universal cycles of k-subsets and k-permutations. *Discrete Mathematics*, 117(1-3):141–150, 1993.

[87] P. Jokinen, J. Tarhio, and E. Ukkonen. A comparison of approximate string matching algorithms. *Software Practice and Experience*, 26(12):1439–1458, 1996.

[88] J. Justin and G. Pirillo. Fractional powers in Sturmian words. *Theoret. Comput. Sci.*, 255(1–2):363–376, 2001.

[89] T. Kamae and L. Q. Zamboni. Maximal pattern complexity for discrete systems. *Ergodic Theory Dynam. Systems*, 22(4):1201–1214, 2002.

[90] A. Karaman. Weak repetitions in Sturmian strings. *Theoret. Comput. Sci.*, 290(3):2137–2146, 2003.

[91] J. Karhumäki, A. Saarela, and L. Q. Zamboni. On a generalization of abelian equivalence and complexity of infinite words. *J. Comb. Theory, Ser. A*, 120(8):2189–2206, 2013.

[92] D. E. Knuth. *Generating All Tuples and Permutations. The Art of Computer Programming, Vol. 4, Fascicle 2.* Addison-Wesley, 2005.

[93] D. E. Knuth, J. H. Morris Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

[94] T. Kociumaka, J. Radoszewski, and W. Rytter. Fast algorithms for abelian periods in words and greatest common divisor queries. In *STACS 2013*, volume 20 of *LIPIcs*, pages 245–256. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[95] R. Kolpakov and G. Kucherov. Finding Maximal Repetitions in a Word in Linear Time. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 596–604. IEEE Computer Society, 1999.

[96] D. Krieger. On critical exponents in fixed points of non-erasing morphisms. *Theoretical Computer Science*, 376(1–2):70–88, 2007.

[97] E. P. Lipatov. A classification of binary collections and properties of homogeneity classes. (Russian). *Problemy Kibernet.*, 39:67–84, 1982.

[98] M. Lothaire. *Combinatorics on words.* Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1997.

[99] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, Cambridge, U.K., 2002.

[100] M. Lothaire. *Applied Combinatorics on Words*. Encyclopedia of Mathematics and its Applications. Cambridge Univ. Press, 2005.

[101] B. Madill and N. Rampersad. The abelian complexity of the paperfolding word. *Discrete Math.*, 313(7):831–838, 2013.

[102] F. Mignosi. Infinite Words with Linear Subword Complexity. *Theoret. Comput. Sci.*, 65(2):221–242, 1989.

[103] F. Mignosi. On the number of factors of Sturmian words. *Theoret. Comput. Sci.*, 82:71–84, 1991.

[104] F. Mignosi and G. Pirillo. Repetitions in the Fibonacci infinite word. *RAIRO Theor. Inform. Appl.*, 26:199–204, 1992.

[105] F. Mignosi and A. Restivo. Characteristic Sturmian words are extremal for the critical factorization theorem. *Theoret. Comput. Sci.*, 454(0):199 – 205, 2012.

[106] F. Mignosi and A. Restivo. A new complexity function for words based on periodicity. *Internat. J. Algebra Comput.*, 23(4):963–988, 2013.

[107] F. Mignosi, A. Restivo, and M. Sciortino. Words and forbidden factors. *Theoret. Comput. Sci.*, 273(1-2):99–117, 2002.

[108] M. Morse and G. A. Hedlund. Symbolic dynamics. *Amer. J. Math.*, 60:1–42, 1938.

[109] G. Navarro and V. Mäkinen. Compressed full-text indexes. *ACM Comput. Surv.*, 39(1), 2007.

[110] R. J. Parikh. On context-free languages. *J. Assoc. Comput. Mach.*, 13(4):570–581, 1966.

[111] S. Puzynina and L. Q. Zamboni. Abelian returns in Sturmian words. *J. Comb. Theory, Ser. A*, 120(2):390–408, 2013.

[112] N. Pytheas Fogg. *Substitutions in Dynamics, Arithmetics and Combinatorics*, volume 1794 of *Lecture Notes in Math.* Springer, 2002.

[113] G. Richomme, K. Saari, and L.Q. Zamboni. Abelian complexity of minimal subshifts. *J. Lond. Math. Soc.*, 83(1):79–95, 2011.

[114] M. Rigo and P. Salimov. Another generalization of abelian equivalence: Binomial complexity of infinite words. In *Combinatorics on Words, 9th International Conference, WORDS 2013*, volume 8079 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 2013.

[115] D. Roy. Approximation to real numbers by cubic algebraic integers. II. *Ann. of Math. (2)*, 158(3):1081–1087, 2003.

[116] D. Roy. Approximation to real numbers by cubic algebraic integers. I. *Proc. London Math. Soc. (3)*, 88(1):42–62, 2004.

[117] F. Ruskey, C. Savage, and T. M. Y. Wang. Generating necklaces. *J. Algorithms*, 13(3):414–430, 1992.

[118] F. Ruskey and A. Williams. An explicit universal cycle for the (n-1)-permutations of an n-set. *ACM Transactions on Algorithms*, 6(3), 2010.

[119] A. Saarela. Ultimately constant abelian complexity of infinite words. *J. Autom. Lang. Comb.*, 14(3-4):255–258, 2010.

[120] A.V. Samsonov and A.M. Shur. On Abelian repetition threshold. *RAIRO Theor. Inform. Appl.*, 46:147–163, 2012.

[121] M. Sciortino and L. Q. Zamboni. Suffix Automata and Standard Sturmian Words. In *Developments in Language Theory, 11th International Conference, DLT 2007*, volume 4588 of *Lecture Notes in Comput. Sci.*, pages 382–398. Springer, 2007.

[122] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. Available electronically at `http://oeis.org`. Sequence A062692.

[123] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. Available electronically at `http://oeis.org`. Sequence A238110.

[124] W. F. Smyth. *Computing Patterns in Strings*. Pearson Addison Wesley (UK), 2003.

[125] B. Tan. Mirror substitutions and palindromic sequences. *Theoret. Comput. Sci.*, 389(1-2):118–124, 2007.

[126] O. Turek. Abelian complexity and abelian co-decomposition. *Theoret. Comput. Sci.*, 469:77–91, 2013.

[127] D. Vandeth. Sturmian words and words with a critical exponent. *Theoret. Comput. Sci.*, 242(1–2):283–300, 2000.

[128] L. Vuillon. Balanced words. *Bull. Belg. Math. Soc. Simon Stevin*, 10(5):787–805, 2003.