Université de Liège

Faculté des Sciences
Unité de Recherche Mathematics

On the k-binomial equivalence of finite words and k-binomial complexity of infinite words

Dissertation présentée le 18 juin 2021 par

Marie Lejeune

en vue de l'obtention du grade de Docteur en Sciences

Michel Rigo
Promoteur

Julien Leroy	Manon Stipulanti	Narad Rampersad	Michaël Rao
Université de Liège	Université de Liège	University of Winnipeg	ENS Lyon
Président	*Secrétaire*	*Rapporteur*	*Rapporteur*

Robert Mercaș	Gwenaël Richomme
University of Loughborough	Université Montpellier 3

"En mathématiques, le mot évident est le mot le plus dangereux."
Eric Temple Bell

Abstract

Complexity functions are well-studied objects in combinatorics on words. They encode some information about an infinite word: they count, for every non-negative integer n, the number of factors of length n present in the infinite word. One may take variations of the classical factor complexity, by counting not all factors, but only those which are different enough one from the other. To this aim one can define an equivalence relation and count, for any n, the number of equivalence classes among all factors of length n of a given infinite word. In this thesis we are interested into the k-binomial equivalence and its associated complexity. This latter equivalence involves the notion of binomial coefficient of words, counting, given two words u and x, the number of times $\binom{u}{x}$ that x appears in u as a subword. Two words u and v are k-binomially equivalent if $\binom{u}{x} = \binom{v}{x}$ for every word x of length up to k. In this manuscript we first count the number of k-binomial equivalence classes and give an algorithm generating the 2-binomial class of a finite word. We also show that the monoid $\mathcal{A}^*/\sim_2$ is isomorphic to the submonoid, generated by $\mathcal{A}$, of the nil-2 group $N_2(\mathcal{A})$. We then compute the exact values of the k-binomial complexity of the Thue–Morse word and the Tribonacci word, and we discuss the techniques employed, in an aim of generalizing it to larger families of words. Finally, we study a variant of the classical reconstruction problem and show that, proceeding in a sequential way, knowing $\lfloor \frac{n}{2} \rfloor + 1$ well-chosen binomial coefficients is sufficient for reconstructing any binary word of length n. We also treat the case of an arbitrary alphabet and show that our bounds are better than what was known in the classical reconstruction case.

Keywords: combinatorics on words, equivalence relations, k-binomial equivalence, complexity functions, factor complexity, k-binomial complexity, growth order, nil-2 group, reconstruction problem for words

Résumé

Les fonctions de complexité sont des objets souvent étudiés en combinatoire des mots. Elles permettent d'encoder beaucoup d'information sur un mot infini : elles comptent, pour chaque naturel n, le nombre de facteurs de longueur n qui apparaissent dans le mot infini. Il est possible de considérer des variations de la complexité factorielle classique en comptant non pas tous les facteurs, mais seulement ceux qui sont assez différents les uns des autres. C'est dans ce but que nous présentons une relation d'équivalence et comptons, pour chaque n, le nombre de classes d'équivalence parmi les facteurs de longueur n du mot infini donné. Dans cette thèse, nous nous intéressons à la relation d'équivalence k-binomiale et à sa fonction de complexité associée. Ces dernières font intervenir la notion de coefficient binomial de deux mots, comptant, étant donnés deux mots u et x, le nombre $\binom{u}{x}$ de fois que le mot x apparait comme sous-mot dans u. Deux mots u et v sont alors dits k-binomialement équivalents si $\binom{u}{x} = \binom{v}{x}$ pour tous les mots x de longueur au plus k. Dans ce manuscrit, nous commençons par compter le nombre de classes d'équivalence k-binomiale et donnons un algorithme permettant de générer la classe 2-binomiale d'un mot fini donné. Nous montrons aussi que le monoïde $\mathcal{A}^*/\sim_2$ est isomorphe au sous-monoïde, généré par $\mathcal{A}$, du nil-2 groupe $N_2(\mathcal{A})$. Nous calculons ensuite les valeurs exactes des fonctions de complexité k-binomiale des mots de Thue–Morse et Tribonacci. Nous discutons des techniques employées dans le but de les généraliser afin de permettre l'étude de la complexité k-binomiale sur des familles plus larges de mots infinis. Enfin, nous étudions une variation du problème de reconstruction de mots. Nous montrons que, en procédant de façon séquentielle, il est possible de reconstruire n'importe quel mot binaire de longueur n en connaissant seulement $\lfloor \frac{n}{2} \rfloor + 1$ coefficients binomiaux bien choisis. Nous traitons aussi le cas d'un alphabet quelconque et montrons que nos bornes sont meilleures que celles connues dans le cas du problème de reconstruction classique.

Mots-clés : combinatoire des mots, relations d'équivalence, équivalence k-binomiale, fonctions de complexité, complexité factorielle, complexité k-binomiale, ordre de croissance, nil-2 groupe, problème de reconstruction des mots

Remerciements

Lorsque vient l'heure de boucler un travail de thèse, les remerciements sont une étape à la fois essentielle, afin d'enfin mettre à l'honneur tous ceux qui ont contribué à la bonne conduite du projet, et délicate, tant il est parfois difficile de trouver les mots justes que l'on souhaite adresser.

Je me dois évidemment de diriger mes premiers remerciements vers mon promoteur, Michel Rigo, pour avoir accepté de me faire confiance en me lançant dans une telle aventure, mais également pour ses conseils avisés, ses coups de pouce indispensables, et enfin son travail de relecture.

Julien Leroy et Robert Mercaș m'ont également encadrée en acceptant de faire partie de mon comité de thèse. Je les remercie pour cela et pour avoir accepté de faire partie de mon jury. Je remercie également Robert pour son accueil plus que chaleureux à Loughborough lors d'une semaine passée sur place, ainsi que pour toutes ses remarques positives et encourageantes qui ont suivi notre rencontre.

Je remercie Narad Rampersard et Michaël Rao d'avoir accepté la lourde tâche de rapporteurs. C'est un luxe de pouvoir compter sur leur expertise lorsqu'il est question de conclure un manuscrit rassemblant plusieurs années de recherche.

Je remercie Manon Stipulanti et Gwénaël Richomme de faire partie de mon jury, en espérant éveiller chez eux un intérêt prononcé pour mon travail de recherche.

J'ai eu la chance de faire partie de la grande et compétente équipe de maths discrètes du Département de Mathématique de l'Université de Liège, avec laquelle j'ai pu nourrir de nombreux échanges fructueux. Merci donc à Émilie, Julien, Manon, Adeline, Célia et France !

Une vie de chercheur en mathématique ne se contente pas d'heures passées seul dans son bureau. Elle s'illumine par des excellentes relations avec les collègues lorsque l'heure de la pause a sonné. Je remercie l'ensemble de mes collègues pour les nombreuses séances de rire collectif que nous avons partagées. En particulier, je me dois de mettre à l'honneur Sophie et Laurent D. qui, en plus d'être des excellents amis, sont des nageurs assidus.

Je remercie mes parents, mon frère et ma *Babcia* qui n'ont jamais eu cesse de croire en moi. Ils ont toujours veillé à mon éveil et à m'assurer la meilleure éducation

qu'il soit. J'espère aujourd'hui les rendre fiers, au regard du chemin parcouru.

Ces derniers mois, j'ai pu compter sur la présence à mes côtés d'une petite boule de poils nommée Euclide. Je suis convaincue que ses ronronnements, une fois installé à mes côtés lors de mon travail de rédaction, étaient sa manière à lui de m'encourager.

Je remercie le FNRS qui m'a permis de parcourir le monde de conférence en conférence lors des premières années de ma thèse.

Finalement, je remercie Laurent pour son soutien indéfectible et sa relecture précieuse. Il a toujours cru en moi plus que quiconque et n'a jamais manqué d'être à mes côtés durant mes moments de doute. Sa présence, ses conseils, ses remarques, nos discussions, ont permis à cette thèse de voir le jour.

Contents

Introduction

This manuscript is a modest contribution to the field of combinatorics on words, which can be seen at the junction between mathematics and computer science as part of "concrete mathematics" following the terminology of [50]. Combinatorics on words is a quite recent subject, the first book devoted to broad study of it [75] being released in 1983. However in the last forty years a huge amount of results were published and numerous reference books arose [6, 15, 76, 77], just to cite a few. This field considers words, which are finite or infinite sequences of a non-commutative monoid, and different tools permitting to study them, such as complexity functions, avoidance of particular patterns,...

Let us fix a finite alphabet $\mathcal{A}$, which is just a finite set whose elements are called letters, and define over it a binary operation called concatenation. We can inductively define finite words: words of length 1 are exactly letters of $\mathcal{A}$, and words of length $n > 1$ are exactly the results of the concatenation of two shorter words. To form a monoid we add the only word of length 0, called the empty word and denoted by ε. The set of all finite words over $\mathcal{A}$ is written $\mathcal{A}^*$. Taking a convenient notion of distance and the associated topological space we extend our definition to infinite words, seen as the limit of a sequence of finite words whose lengths grow.

Since the alphabet $\mathcal{A}$ is finite, there are only a finite number of words of a given length n. Hence in an infinite word, at least one word of length n appears several times as a "block" (these blocks are called *factors* and are defined more formally in what follows). To get a better understanding of a given infinite word $\mathbf{w}$, one can study its set of factors. The factor complexity of $\mathbf{w}$ is the non-decreasing function mapping every natural number n to the number of distinct factors of length n in $\mathbf{w}$. The first trace of such a function goes back to 1938 where the *block growth* was used as a tool to study symbolic dynamical systems [86]. The name *subword complexity* was then given forty years later in [37], but now these appellations have disappeared for the benefit of *factor complexity*. A famous result concerning this notion is the Morse–Hedlund theorem [6, 15, 87] stating that an infinite word is ultimately periodic (i.e., at some point in this word we only see a unique block repeating) if and only if there exists $n \geq 1$ such that the value of its factor complexity in n is at most n.

One can slightly modify this complexity function to count not all distinct factors appearing, but all "different enough" factors occurring in a fixed infinite word. The strategy is the following: define an equivalence relation $\sim$ on finite words in such a way that $u \sim v$ if and only if u and v are similar enough. We can define different equivalence relations depending on what we want to call "similar enough", but one common condition is that if two words are equivalent then they have the same length. We then define the complexity function of an infinite word $\mathbf{w}$, associated with the equivalence relation $\sim$, as the function mapping to every natural number n the number of equivalence classes, for $\sim$, among the set of factors of length n of $\mathbf{w}$. Let us present several important equivalence relations. First note that the classical factor complexity can be obtained using this process and taking the trivial equivalence relation $\sim_=$: two words are equivalent if and only if they are equal.

Abelian equivalence has been investigated for quite a long time; two words u and v are Abelian equivalent if and only if for any letter $a \in \mathcal{A}$, the number of occurrences of a in u equals the number of occurrences of a in v. In the sixties Erdös raised the combinatorial question whether Abelian squares (concatenation of two words that are Abelian equivalent) can be avoided by an infinite word whose alphabet is of size four [24, 30, 106].

Motivated by a generalization of Parikh's theorem [94], Karhumäki considered a more strict condition: let us denote $|u|_x$ the number of times that the word x occurs in u as a factor. k-Abelian equivalence ($k \in \mathbb{N}$) was defined as follows [58, 59]: two words u and v are k-Abelian equivalent, noted $\sim_{k,\mathrm{ab}}$, if and only if $|u|_x = |v|_x$ for all words x of length at most k. Note that the more k increases, the more similar words u and v have to look to be equivalent, since $\sim_{k+1,\mathrm{ab}}$ is a refinement of $\sim_{k,\mathrm{ab}}$: if $u \sim_{k+1,\mathrm{ab}} v$ then $u \sim_{k,\mathrm{ab}} v$.

As we have seen before, x is a factor of u if we can write $u = u_1 \cdots u_n$, $x = x_1 \cdots x_\ell$ (all u_i and x_i being letters of $\mathcal{A}$) and if there exists $j \leq n - \ell + 1$ such that $u_j \cdots u_{j+\ell-1} = x_1 \cdots x_\ell$. It means that all letters of x have to appear consecutively in u. If we relax this last affirmation, we obtain what we call a *subword* of u. In this case there exist $i_1 < \cdots < i_\ell$ such that $1 \leq i_1$, $i_\ell \leq n$ and $u_{i_1} \cdots u_{i_\ell} = x_1 \cdots x_\ell$. Similarly to what we did for factors, we can count the number of occurrences of x as a subword of u (that is, the number of convenient sequences $i_1 < \cdots < i_\ell$) and denote by the binomial coefficient $\binom{u}{x}$ this quantity. So, given a finite word u, the knowledge of some binomial coefficients $\binom{u}{x}$ provides information about u. It leads to the following definition of k-binomial equivalence ($k \in \mathbb{N}$), first introduced by Rigo and Salimov six years ago [109]: two words u and v are k-binomially equivalent, noted $\sim_k$, if $\binom{u}{x} = \binom{v}{x}$ for all words x of length at most k. Since the definition is similar to the one of $\sim_{k,\mathrm{ab}}$, the same behavior holds: $u \sim_{k+1} v$ implies $u \sim_k v$. Note that one can't generally compare $\sim_{k,\mathrm{ab}}$ with $\sim_k$: on the one hand, words *abacab* and *acabab* are 2-Abelian

equivalent but not 2-binomially equivalent, while on the other hand, words *abba* and *baab* are 2-binomially equivalent but not 2-Abelian equivalent.

Following what we announced, we define the following complexity functions: Abelian (resp., k-Abelian, k-binomial) complexity function of $\mathbf{w}$ maps to every $n \in \mathbb{N}_0$ the number of Abelian (resp., k-Abelian, k-binomial) equivalence classes among all factors of $\mathbf{w}$ of length n. The first traces of these complexities can be found in [29, 58, 107, 109]. On the one hand, many things are known about (k-)Abelian complexity of several well-known words, see [26, 51, 55] for Thue–Morse and its generalized versions, [95] for 2-automatic sequences, [16] for binary uniform morphic words, or finally [25, 78, 79, 107, 122] for other well-known particular words such as the Tribonacci word, the Cantor sequence,... On the other hand, only two main results were established for k-binomial equivalence. They can be found in [109] and we recall them here.

The first one says that, for any $k \geq 2$, the k-binomial complexity function of any *Sturmian* word equals its factor complexity. Sturmian words can be defined [76, Chapter 2] as the infinite words whose factor complexity equals $p(n) = n + 1$.

The second interesting result states that any fixed point of a Parikh-constant morphism (a morphism whose images are Abelian equivalent) has a k-binomial complexity ($k \geq 2$) bounded by a constant depending on k. Fixed points of Parikh-constant morphisms represent a large class of words and will be discussed in Chapter 3.

The fact that only these two results are established for k-binomial equivalence motivates its study, hence this manuscript is partially devoted to the analysis of several well-known infinite words and their associated k-binomial complexity, in an aim of developing general techniques that could be employed for larger families of words. The motivations to better understand these notions are multiple.

In formal language theory, Simon k-congruence considers as similar two words having the same set of subwords of length up to k (not taking into account the multiplicities of occurrences). This notion is in particular related to piecewise testable languages [57, 118]. Recent algorithmic developments have been done in that direction [48].

Closely related to binomial coefficients, the Parikh vector of a word u is encoding all quantities $|u|_a$, $a \in \mathcal{A}$. One can introduce Parikh matrices extending the notion of Parikh vector. In these matrices, only some binomial coefficients for subwords of length up to k appear. There is a vast literature on the corresponding equivalence, i.e., when two words have the same Parikh matrix [82, 83, 84]. In particular, Salomaa studied the k-spectrum of a word: two words having the same k-spectrum are k-binomially equivalent [111].

Related to numeration systems and digital functions, several properties of generalized Pascal triangles have been investigated, leading to the fractal structures like

Sierpiński gaskets [72].

Finally, there exists a strong link with algebra (Lyndon words appear in Lie algebras) [105], p-adic topology [13] and non-commutative extension of Mahler's theorem on interpolation series [99].

Let us now present the articulation of this thesis. In the first chapter we state the context and recall most common definitions. Consulted reference books are [6, 15, 75, 76, 77]. We then formally introduce the notion of binomial coefficient of words and the different equivalence relations (and their associated complexities) discussed here above. We give several tools for simplifying the computation of a binomial coefficient and we discuss the (non-)existence of a variation of the Morse-Hedlund theorem for k-binomial complexity.

The second chapter concerns k-binomial equivalence and a description of its equivalence classes. We first have a glimpse into automata and regular languages theory, before studying two particular languages: $\mathsf{LL}(\sim_k, \mathcal{A})$ is defined to be the set of least lexicographical elements of each $\sim_k$-equivalence class, while $\mathsf{Sing}(\sim_k, \mathcal{A})$ denotes the set of words of $\mathcal{A}^*$ that are alone (i.e., singletons) in their $\sim_k$-equivalence class. The main goal of this chapter is to show that these two languages are not context-free, hence not regular. It follows the trails of Whiteland who proved in his thesis [125] that, when considering k-Abelian equivalence $\sim_{k,\mathrm{ab}}$, languages $\mathsf{LL}(\sim_{k,\mathrm{ab}}, \mathcal{A})$ and $\mathsf{Sing}(\sim_{k,\mathrm{ab}}, \mathcal{A})$ are regular. The result concerning the k-binomial case can be seen as an indicator of the complexity of the studied problem. Indeed, non-context-free languages are difficult to characterize. Our reasoning is divided into several sections. We discuss the easy case ($k = 2$ and $\#\mathcal{A} = 2$) in Section 2.2 and then concentrate in Sections 2.3 to 2.5 on the general case $k \geq 2$ and $\#\mathcal{A} \geq 3$. Motivated by [44] where an algorithm deciding if $u \sim_k v$ or not, we present in Subsection 2.3.2, in the case $k = 2$, an algorithm allowing to easily compute the $\sim_2$-equivalence class of a word u, by building a tree whose nodes encode words and whose edges represent a single swap between two adjacent letters. Moreover we show in Subsection 2.3.3 that the monoid $\mathcal{A}^*/\sim_2$ is isomorphic to the submonoid, generated by $\mathcal{A}$, of the nil-2 group $N_2(\mathcal{A})$. As for Parikh matrices, things become more complex for $k \geq 3$.

In the third and fourth chapters we want to study the value of the k-binomial complexity of two famous words, namely the Thue–Morse and the Tribonacci words, in an aim of obtaining generalizable techniques that could be applied to other infinite words. The Thue–Morse word $\mathbf{t}$ is probably the most classical binary infinite word [5, 101, 120] and has numerous interesting properties [3, 4, 34, 63]. It is the fixed point (starting by 0) of the morphism φ such that $\varphi(0) = 01$ and $\varphi(1) = 10$. Otherwise stated, $\mathbf{t} = \lim_{n\to+\infty} \varphi^n(0)$. It is a particular fixed point of a Parikh-constant morphism hence as we have seen, for all $k \geq 2$, its k-binomial complexity is bounded by a constant C_k. The exact value of this complexity is computed in details in Chapter 3.

We first give formulas allowing to compute $\binom{\varphi(u)}{v}$ from $\binom{u}{v}$ and differences of the type $\binom{\varphi(u)}{v} - \binom{\varphi(w)}{v}$. We then deal with the case $k = 2$ separately in Section 3.2 before going to the general technique, where we develop a theory of cutting bars, cutting sets and types of factors of $\mathbf{t}$. We finally establish the value of the k-binomial complexity of $\mathbf{t}$ ($k \geq 3$) in Section 3.4.

Chapter 4 is devoted to the study of the Tribonacci word, which is the limit $\mathbf{T} = \lim_{n\to+\infty} \tau^n(0)$, with $\tau(0) = 01$, $\tau(1) = 02$ and $\tau(2) = 0$. This word is interesting since it can be seen as the generalization of the Fibonacci word, which is a particular Sturmian word (and recall that the equality between factor complexity and 2-binomial complexity was established for all Sturmian words in [109]). It was conjectured that the same equality holds for $\mathbf{T}$. Unfortunately classical combinatorial techniques do not seem to work here, so we made an extensive use of the concepts of templates and their ancestors, similarly to what can be found in [1, 2, 31]. In Section 4.2 we define and adapt the notions of (realizable) templates and ancestors to our purpose, before bounding the number of them in Section 4.3 and establishing the main result, that is, the equality between the two complexities, in Section 4.4. Similarly to what has already be done before [1, 2, 31, 74, 103], our proof is a computer-assisted one and we hope that the algorithm could be applied to other families of infinite words.

We conclude this thesis by a slightly different subject, in the sense that it is not related to k-binomial complexity anymore. However it still involves the main tool of our work, binomial coefficients. The classical *reconstruction problem* can be stated in numerous distinct fields: given a sufficient amount of information about substructures of a hidden discrete structure, can one uniquely determine this structure? And, in particular, what are the fragments needed to recover the whole structure? Such problems have been extensively studied for example for matrices [81], graphs [52, 60, 92] or words [35, 36, 56, 62, 80, 118]. Given an unknown word u of length n and a multiset of subwords of u (in an equivalent way, given the values of several binomial coefficients $\binom{u}{v}$), can one uniquely determine u? We study here a slight variant of the problem: we ask queries of the type "What is the value of $\binom{u}{v_i}$?" for different words $v_1, v_2, \ldots, v_k$ in a sequential way, meaning that the answer to the i^{th} question can influence the choice of v_{i+1}. We would like to ask few questions, so we are interested, for a fixed $n \in \mathbb{N}$, in the minimal k such that any word of length n can be uniquely determined by asking at most k queries. We show in Section 5.2 that $k = \lfloor \frac{n}{2} \rfloor + 1$ for a binary alphabet. We then consider the general case in Section 5.3 by trying to reconstruct every binary projection of u using the results of the previous section, and we show in Subsection 5.3.1 that a word can always be reconstructed from its binary projections. We finally prove that the obtained bounds on k are better than the best bounds known for the classical reconstruction problem on words [62]. I had the opportunity with this subject to work with researchers from other universities such as the Kiel University or the University

of Göttingen.

This manuscript presents the content of the original papers [41, 66, 67, 68, 69, 71]. Among these, paper [66] has been presented at *Developments in Language Theory* 2019 in Warsaw while paper [68] has been presented at *WORDS* 2019 in Loughborough.

1 Preliminaries

This chapter aims at stating notation and the background for the rest of the manuscript. The first section concerns very basic notions from combinatorics on words such as alphabets, words, factors, subwords, morphisms and purely morphic words. The next section introduces binomial coefficients for words and several basic related results. Finally, Sections 1.3 and 1.4 present (k-)Abelian and k-binomial equivalences as well as their associated complexity functions. This last section finishes by a state of the art.

Contents

1.1 Back to basics

We will denote by $\mathbb{N}$ the set of natural numbers $\{1, 2, \ldots\}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. We will use the following notation: for any $n \in \mathbb{N}$, $[n] = \{1, \ldots, n\}$, $[n]_0 = [n] \cup \{0\}$ and if $i \leq n$, $[n]_{\geq i} = [n] \setminus [i-1]$. The following classical definitions can be found in [15] or [75] for example.

Definition 1.1.1. An *alphabet* is a finite set whose elements are called *symbols* or *letters*. A *finite (resp., infinite) word* over the alphabet $\mathcal{A}$ is a finite (resp., infinite) sequence over $\mathcal{A}$. An infinite word is sometimes called a *sequence*. The *empty word* is the unique word composed of zero letters. It is denoted by ε. The *length* of a word u is the number $|u|$ of letters composing it. If u is a finite word, $|u| \in \mathbb{N}_0$. If $\mathbf{u}$ is an infinite word, $|\mathbf{u}| = \infty$.

In this text, words will usually be denoted by lowercase letters. Moreover, to stress the fact that a word is an infinite word, it will be in bold font.

We denote by $\mathcal{A}^*$ (resp., $\mathcal{A}^{\mathbb{N}}$) the set of all finite (resp., infinite) words over $\mathcal{A}$. We note $\mathcal{A}^\infty$ the set $\mathcal{A}^* \cup \mathcal{A}^{\mathbb{N}}$. Moreover, for every $n \in \mathbb{N}_0$, let

$$\mathcal{A}^n = \{u \in \mathcal{A}^* : |u| = n\},$$

$$\mathcal{A}^{\leq n} = \{u \in \mathcal{A}^* : |u| \leq n\} \quad \text{and} \quad \mathcal{A}^{<n} = \{u \in \mathcal{A}^* : |u| < n\}.$$

Definition 1.1.2. The *concatenation* is a binary operation

$$\cdot : (\mathcal{A}^* \times \mathcal{A}^\infty) \to \mathcal{A}^\infty$$

which associates to the pair (u, v) the word $u \cdot v$ obtained by concatenating the sequences of letters of u and v.

The word $u \cdot v$ is also simply written uv. The word uu is written u^2 and by induction, $u^n = u^{n-1} \cdot u$ for any $n \in \mathbb{N}$. By convention, $u^0 = \varepsilon$ and $u^1 = u$.

The set $\mathcal{A}^*$ equipped with the concatenation and the neutral element ε forms a monoid.

Notation 1.1.3. A word $u \in \mathcal{A}^n$ will often be denoted as $u_1 \cdots u_n$. If nothing else is specified, it is understood that $u_i \in \mathcal{A}$ for every $i \in [n]$. A word $\mathbf{u} \in \mathcal{A}^{\mathbb{N}}$ will often be denoted as $\mathbf{u}_1\mathbf{u}_2 \cdots$. Similarly, it is understood that $\mathbf{u}_i \in \mathcal{A}$ for every $i \in \mathbb{N}$.

Definition 1.1.4. Let $u = u_1 \cdots u_n \in \mathcal{A}^n$ be a finite word. A *subword* of u is a subsequence $u_{i_1}u_{i_2} \cdots u_{i_\ell}$ of $u_1 \cdots u_n$ with $1 \leq i_1 < i_2 < \cdots < i_\ell \leq n$. If $i_1 + \ell - 1 = i_\ell$ (i.e., if the subsequence is made with consecutive letters), the subword is called a *factor* of u. If in addition $i_1 = 1$, the factor is called a *prefix* of u, and if $i_\ell = n$, the factor is called a *suffix* of u. If $\ell < n$, the factor is different from u and called a *proper* factor[1] of u.

We write $\mathrm{Fac}(u)$ (resp., $\mathrm{Pref}(u)$, $\mathrm{Suff}(u)$) the set of factors (resp., prefixes, suffixes) of u. Similarly, we write $\mathrm{PPref}(u)$ (resp., $\mathrm{PSuff}(u)$) the set of proper prefixes (resp., proper suffixes) of u. We also denote by $\mathrm{Fac}_n(u)$ the set of factors of length n of u:

$$\mathrm{Fac}_n(u) = \mathrm{Fac}(u) \cap \mathcal{A}^n.$$

[1]Here we consider that the empty word is a proper factor, it is not always the case in the literature.

By convention, $\varepsilon \in \mathrm{Fac}(u)$ for any $u \in \mathcal{A}^*$. We can naturally extend these definitions to infinite words.

Definition 1.1.5. Let $\mathbf{u} = \mathbf{u}_1\mathbf{u}_2\cdots \in \mathcal{A}^{\mathbb{N}}$. A word $v \in \mathcal{A}^*$ is a *subword* (resp., *factor*, *prefix*) of $\mathbf{u}$ if it is a subword (resp., factor, prefix) of the finite word $\mathbf{u}_1\mathbf{u}_2\cdots\mathbf{u}_n$ for some $n \in \mathbb{N}$.

1.1.1 Introducing a distance

Infinite words can sometimes be seen as the limit of a sequence of finite words. We thus have to introduce a distance and a convergence notion.

Notation 1.1.6. Let $u, v \in \mathcal{A}^*$. We denote by $\Lambda(u,v)$ the longest common prefix of u and v, that is, the word $w \in \mathcal{A}^*$ such that $w \in \mathrm{Pref}(u) \cap \mathrm{Pref}(v)$ and $wa \notin \mathrm{Pref}(u) \cap \mathrm{Pref}(v)$ for any $a \in \mathcal{A}$.

Note that $|\Lambda(u,v)| \leq \min(|u|,|v|)$. We can now introduce a distance on $\mathcal{A}^\infty$.

Definition 1.1.7. Let $u, v \in \mathcal{A}^*$. The *distance* between u and v is given by

$$d(u,v) := \begin{cases} 0, & \text{if } u = v; \\ 2^{-|\Lambda(u,v)|}, & \text{otherwise.} \end{cases}$$

It is easy to check that $d(\cdot,\cdot)$ is a good definition; indeed it verifies the following properties:

- (symmetry) $d(u,v) = d(v,u)$ for all $u, v \in \mathcal{A}^*$;
- (separation) $d(u,v) = 0$ if and only if $u = v$;
- (triangle inequality) $d(u,v) \leq d(u,w) + d(w,v)$ for all $u, v, w \in \mathcal{A}^*$.

Moreover, this distance is *ultrametric*, which means that, for all $u, v, w \in \mathcal{A}^*$,

$$d(u,v) \leq \max\{d(u,w), d(v,w)\}.$$

Note that this last property implies the triangle inequality.

This distance can naturally be extended on $\mathcal{A}^\infty$, by taking the convention that $|\Lambda(\mathbf{u},\mathbf{v})| = \infty$ if $\mathbf{u} = \mathbf{v}$ are infinite words, and that $2^{-\infty} = 0$. From this distance, a convergence notion can be introduced: the sequence of words $(u_n)_{n\geq 1} \in (\mathcal{A}^\infty)^{\mathbb{N}}$ converges to $\mathbf{u}$ if and only if

$$\lim_{n\to+\infty} d(u_n, \mathbf{u}) = 0.$$

If $v \in \mathcal{A}^*$, the infinite word v^ω is the limit of the sequence $(v^n)_{n\geq 1}$. Such a word is called a *purely periodic* word.

1.1.2 Morphisms and fixed points

As in many algebraic structures, morphisms of monoids can be defined.

Definition 1.1.8. Let $\mathcal{A}$ and $\mathcal{B}$ be two alphabets. An application

$$\sigma : \mathcal{A}^* \to \mathcal{B}^*$$

is a *morphism* if the image of any word is the concatenation of the images of its letters. Otherwise stated, for every word $u = u_1 \cdots u_n \in \mathcal{A}^n$,

$$\sigma(u) = \sigma(u_1) \cdots \sigma(u_n).$$

Therefore, a morphism is completely defined by the set $\{\sigma(a) : a \in \mathcal{A}\}$. A morphism is *non-erasing* if, for all $a \in \mathcal{A}$, $\sigma(a) \neq \varepsilon$.

Let $\mathbf{u} \in \mathcal{A}^{\mathbb{N}}$ be an infinite word and let $\sigma : \mathcal{A}^* \to \mathcal{A}^*$ be a morphism. We say that $\mathbf{u}$ is a *fixed point of the morphism* σ if $\lim_{n\to+\infty} \sigma(\mathbf{u}_1 \cdots \mathbf{u}_n) = \mathbf{u}$. By abuse of notation we will write $\sigma(\mathbf{u}) = \mathbf{u}$.

Proposition 1.1.9. *Let $\sigma : \mathcal{A}^* \to \mathcal{B}^*$ be a morphism. If*

- *there exists $a \in \mathcal{A}$ and $u \in \mathcal{A}^*$ such that $\sigma(a) = au$ (i.e., the image of a starts with a);*
- *the limit*

$$\lim_{n\to+\infty} |\sigma^n(a)|$$

tends to infinity;

then the sequence of words

$$(\sigma^n(a))_{n\geq 1}$$

converges to an infinite word from $\mathcal{A}^{\mathbb{N}}$, which is a fixed point of the morphism σ.

Not all fixed points of morphisms are obtained such as in the previous proposition: take for example $\sigma : a \mapsto ab, b \mapsto b$ and $\mathbf{u} = bab^\omega$.

A word is called *purely morphic* if it can be obtained as the fixed point of a morphism. As an example, one can take the well-known Thue–Morse word, defined as the fixed point, starting with 0, of the following morphism:

$$\varphi : \{0,1\}^* \to \{0,1\}^* : \begin{cases} 0 & \mapsto & 01; \\ 1 & \mapsto & 10. \end{cases}$$

The Thue–Morse word $\mathbf{t}$ is thus the limit $\lim_{n\to+\infty} \varphi^n(0)$.

We can trivially extend the notion of set of factors (resp., (proper) prefixes, (proper) suffixes) to a set of words: if $S \subset \mathcal{A}^*$, then for example

$$\mathrm{Fac}(S) = \bigcup_{u\in S} \mathrm{Fac}(u).$$

It is thus meaningful to define the set of factors (resp., (proper) prefixes, (proper) suffixes) of a morphism as the set of factors (resp., (proper) prefixes, (proper) suffixes) of its images of the letters of the alphabet. Hence for example:

$$\mathrm{PPref}(\sigma) = \bigcup_{a \in \mathcal{A}} \mathrm{PPref}(\sigma(a)).$$

1.1.3 Factor complexity

Definition 1.1.10. Let $\mathbf{w}$ be an infinite word. Its *factor complexity function* (or simply *factor complexity*) is the function

$$p_{\mathbf{w}} : \mathbb{N}_0 \to \mathbb{N} : n \mapsto \#\mathrm{Fac}_n(\mathbf{w}),$$

which associates to every $n \in \mathbb{N}_0$ the number of distinct factors of $\mathbf{w}$ of length n.

The factor complexity was introduced in 1938 by Morse and Hedlund [86], under the name of *block growth*, as a tool to study symbolic dynamical systems. The name *subword complexity* was given in 1975 by Ehrenfeucht, Lee, and Rozenberg [37]. Now we use more the notion of factor complexity, since subwords are non-necessarily contiguous factors. Factor complexity is a well-studied object, see for example [15, Chapter 4] where a whole chapter is devoted to this notion. One of the most famous results is the Morse–Hedlund theorem, which characterizes ultimately periodic words in terms of a bounded factor complexity function; for a reference, see [6, 87] or [15, Section 4.3].

Definition 1.1.11. An *ultimately periodic* infinite word $\mathbf{w}$ is a word of the form

$$\mathbf{w} = u \cdot v^{\omega},$$

where u, v are finite words. If $u = \varepsilon$, the word $\mathbf{w}$ is *purely periodic*. A word that is not ultimately periodic is *aperiodic*.

Theorem 1.1.12 (Morse–Hedlund)**.** *Let $\mathbf{w}$ be an infinite word on the alphabet $\mathcal{A}$. The following conditions are equivalent:*

1. *The word $\mathbf{w}$ is ultimately periodic;*
2. *The function $p_{\mathbf{w}}$ is bounded by a constant;*
3. *There exists $n \in \mathbb{N}$ such that $p_{\mathbf{w}}(n) < n + \#\mathcal{A} - 1$.*

This result gives a meaning to the following definition.

Definition 1.1.13. A word $\mathbf{w}$ is *Sturmian* if its factor complexity is $p_{\mathbf{w}}(n) = n + 1$ for every $n \in \mathbb{N}_0$.

Hence all Sturmian words are built on binary alphabets. Moreover, these words are the aperiodic words having the least possible values for their factor complexity. There exist other equivalent definitions for Sturmian words. We refer the interested reader to [76, Chapter 2].

1.2 Binomial coefficients

Factors and subwords of a word are important objects. We can ask the following question: does the word v appears as a factor/subword in u? But we could be more precise and ask how many times this subword occurs. Let $u = u_1 \cdots u_n$. If $u_{i_1} \cdots u_{i_\ell} = v$ and $u_{i'_1} \cdots u_{i'_\ell} = v$ with $1 \leq i_1 < \cdots < i_\ell \leq n$, $1 \leq i'_1 < \cdots < i'_\ell \leq n$ and such that $(i_1, \ldots, i_\ell) \neq (i'_1, \ldots, i'_\ell)$, we consider that v occurs as a subword of u two times.

We denote by $|u|_v$ the number of times that v appears as a factor in u, and by $\binom{u}{v}$, called the binomial coefficient of u and v, the number of times that v appears as a subword in u. Here is a formal definition.

Definition 1.2.1. Let $u = u_1 \cdots u_n \in \mathcal{A}^n$ and $v \in \mathcal{A}^\ell$. The *binomial coefficient* of u and v is

$$\binom{u}{v} = \#\{(i_1, \ldots, i_\ell) \in \mathbb{N}^\ell \mid 1 \leq i_1 < \cdots < i_\ell \leq n \text{ and } u_{i_1} \cdots u_{i_\ell} = v\}.$$

Similarly, the number of occurrences of v as a factor of u is

$$|u|_v = \#\{(i_1, \ldots, i_\ell) \in \mathbb{N}^\ell \mid 1 \leq i_1 < \cdots < i_\ell \leq n,\ u_{i_1} \cdots u_{i_\ell} = v \text{ and } i_1 + \ell - 1 = i_\ell\}.$$

As a convention we take $|u|_\varepsilon = \binom{u}{\varepsilon} = 1$ for any $u \in \mathcal{A}^*$.

Note that since every factor is a subword, $|u|_v \leq \binom{u}{v}$ for all words u and v. Let $k \in \mathbb{N}$; the *k-deck* of $u = u_1 \cdots u_n$ is the multiset[2]

$$\{u_{i_1} \cdots u_{i_k} \mid 1 \leq i_1 < \cdots < i_k \leq n\}$$

of subwords of u of length k. The multiplicity of every $v \in \mathcal{A}^k$ in the k-deck of u is thus $\binom{u}{v}$. The 1-deck is the simplest k-deck of a word, and it encodes the number of occurrences of every letter of the alphabet. It is often given in another way, called the *Parikh vector*, or *abelianization*[3] of a word $u \in \mathcal{A}$. Fix an order $\{a_1 < \cdots < a_\ell\}$ on $\mathcal{A}$. Then the Parikh vector of u is the vector

$$\underline{\Psi}(u) = (|u|_{a_1}, \ldots, |u|_{a_\ell})^\intercal.$$

[2] A *multiset* is just a set where elements can be repeated with a finite integer multiplicity.

[3] The term Parikh vector comes from Parikh's theorem in formal language theory stating that if L is a context-free language, then the set of Parikh vectors of its elements is a semi-linear set [94]. This terminology is now consecrated even though abelianization should be more suited in a mathematical context.

We take the convention in the manuscript to write vectors in bold font, as usual, but also underlined, in an aim of avoiding confusion with infinite words.

We will need a special terminology[4] in this thesis, to emphasize the upper or the lower word in the binomial coefficient. The binomial coefficient can be seen as a two-variable application

$$\binom{\cdot}{\cdot} : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{N}_0.$$

By fixing any of the two variables, we obtain partial applications

$$\binom{u}{\cdot} : \mathcal{A}^* \to \mathbb{N}_0 : v \mapsto \binom{u}{v} \quad \text{and} \quad \binom{\cdot}{v} : \mathcal{A}^* \to \mathbb{N}_0 : u \mapsto \binom{u}{v}.$$

We call any image of the application $\binom{u}{\cdot}$ a binomial coefficient *applied on* (or simply *on*) u. Similarly, we call any image of the application $\binom{\cdot}{v}$ a binomial coefficient *using* v, or *in* v.

Binomial coefficients have been extensively studied, see for example [75, Section 6.3]. They have been successfully used in several applications: p-adic topology [13], non-commutative extension of Mahler's theorem on interpolation series [99], formal language theory [57], Parikh matrices, and a generalization of Sierpiński triangle [72]. The following proposition lists several direct but fundamental properties of them, see [109] for example.

Proposition 1.2.2. *1. For any $u \in \mathcal{A}^*$ and any $a \in \mathcal{A}$, $\ell \in \mathbb{N}_0$, we have*

$$\binom{u}{a^\ell} = \binom{|u|_a}{\ell},$$

where the right member is the classical binomial coefficient on natural numbers.

2. For any $u, v \in \mathcal{A}^*$ and any $a \in \mathcal{A}$, we have

$$\binom{ua}{va} = \binom{u}{va} + \binom{u}{v} \quad \text{and} \quad \binom{au}{av} = \binom{u}{av} + \binom{u}{v}.$$

3. For any $u, v \in \mathcal{A}^*$ and any $a, b \in \mathcal{A}$ such that $a \neq b$, we have

$$\binom{ua}{vb} = \binom{u}{vb} \quad \text{and} \quad \binom{au}{bv} = \binom{u}{bv}.$$

4. For any $u, v \in \mathcal{A}^*$ and any $w = w_1 \cdots w_n \in \mathcal{A}^n$, we have

$$\binom{uv}{w} = \sum_{i=0}^{n} \binom{u}{w_1 \cdots w_i} \binom{v}{w_{i+1} \cdots w_n},$$

where $w_1 \cdots w_0 = w_{n+1} \cdots w_n = \varepsilon$.

[4] No particular terminology was found in the literature.

1.2.1 Computing a binomial coefficient using Lyndon words

Reutenauer obtained in [105] a formula allowing to compute a binomial coefficient $\binom{u}{v}$ by involving binomial coefficients using particular words, called Lyndon words.

Let $<$ be a total ordering on the alphabet $\mathcal{A}$. Then, we denote by $<_{\text{lex}}$ the *lexicographical ordering* on $\mathcal{A}^*$ induced by $<$: we have $u <_{\text{lex}} v$ if and only if

- u is a prefix of v and $u \neq v$, or;
- there exists $i \in [\min(|u|, |v|) - 1]$ such that $u_1 \cdots u_i$ is a common prefix to u and v and $u_{i+1} < v_{i+1}$.

We denote by $\prec$ the *radix order* (sometimes also called *genealogical order*) on $\mathcal{A}^*$ induced by $<$. Otherwise stated, $u \prec v$ if and only if

- $|u| < |v|$, or;
- $|u| = |v|$ and $u <_{\text{lex}} v$.

Definition 1.2.3. Let $<$ be a total ordering on the alphabet $\mathcal{A}$. A word $u \in \mathcal{A}^*$ is a *Lyndon word* if and only if for all $v, w \in \mathcal{A}^+$ such that $u = vw$, we have $u <_{\text{lex}} wv$.

To express Reutenauer's formula we need to introduce two definitions.

Notation 1.2.4. Let $x, y \in \mathcal{A}^*$ be two words of length n_x and n_y respectively. Let $I_x = \{i_{x,1}, i_{x,2}, \ldots, i_{x,n_x}\}$ and $I_y = \{i_{y,1}, i_{y,2}, \ldots, i_{y,n_y}\}$ be two subsets of $\mathbb{N}$ of cardinalities n_x and n_y respectively, such that there exists $n \in \mathbb{N}$ for which $I_x \cup I_y = [n]$. We denote by $w(I_x, I_y)$ the word w such that both equalities $w_{i_{x,1}} w_{i_{x,2}} \cdots w_{i_{x,n_x}} = x$ and $w_{i_{y,1}} w_{i_{y,2}} \cdots w_{i_{y,n_v}} = y$ hold.

Note that $w(I_x, I_y)$ is always well defined when $I_x \cap I_y = \emptyset$, but it is not the case if $i_{x,j} = i_{y,k}$ but $x_j \neq y_k$ for some $j \leq n_x, k \leq n_y$. In that case, we take the convention that $w(I_x, I_y) = \varepsilon$.

Definition 1.2.5. Let $n_x, n_y \in \mathbb{N}$, $x \in \mathcal{A}^{n_x}$, and $y \in \mathcal{A}^{n_y}$. Set $n = n_x + n_y$. The *shuffle* of x and y is the polynomial $x \sqcup\!\sqcup y = \sum_{(I_x, I_y)} w(I_x, I_y)$ where the sum has to be taken over all pairs (I_x, I_y) of sets that are partitions of $[n]$ such that $\#I_x = n_x$ and $\#I_y = n_y$.

The infiltration is a variant of the shuffle in which equal letters can be merged.

Definition 1.2.6. Let $n_x, n_y \in \mathbb{N}$, $x \in \mathcal{A}^{n_x}$, and $y \in \mathcal{A}^{n_y}$. Set $n = n_x + n_y$. The *infiltration* of x and y is the polynomial $x \downarrow y = \sum_{(I_x, I_y)} w(I_x, I_y)$, where the sum has to be taken over all pairs (I_x, I_y) of sets of cardinality n_x and n_y respectively, for which there exists $n' \in [n]$ such that $I_x \cup I_y = [n']$.

Based on Definitions 1.2.5 and 1.2.6, we are able to give a formula to compute a given binomial coefficient only from binomial coefficients in Lyndon words. This formula appears implicitly in [105, Theorem 6.4].

Lemma 1.2.7. *[105, Corollary 6.2] Let $v \in \mathcal{A}^{\geq 2}$ be a non-Lyndon word. There exist $x, y \in \mathcal{A}^+$ such that $v = xy$ and such that every word appearing in the polynomial $x \sqcup\!\sqcup y$ is lexicographically less than or equal to v.*

Theorem 1.2.8. *Let $v \in \mathcal{A}^{\geq 2}$ be a non-Lyndon word and let $x, y \in \mathcal{A}^+$ be as in the previous lemma. Then, for any word $u \in \mathcal{A}^*$, we have*

$$\binom{u}{v} = \frac{1}{(x \sqcup\!\sqcup y, v)} \left[\binom{u}{x}\binom{u}{y} - \sum_{w \in \mathcal{A}^+, w \neq v} (x \downarrow y, w) \binom{u}{w} \right],$$

where (P, w) denotes the coefficient of the word w in the polynomial P.

One may apply recursively this formula until only Lyndon words are considered.

Corollary 1.2.9. *Let $u \in \mathcal{A}^*$, and let $v \in \mathcal{A}^{\geq 2}$ be a non-Lyndon word. It is possible to express the binomial coefficient $\binom{u}{v}$ with a formula only involving binomial coefficients on u using Lyndon words.*

Proof. Let x, y be two words satisfying Lemma 1.2.7 for v. All words appearing in $x \sqcup\!\sqcup y$ are either equal to v, or lexicographically smaller than v. Therefore, all words appearing in $x \downarrow y$ are either equal to v, or genealogically smaller than v. We can apply Theorem 1.2.8 inductively on non-Lyndon words from $x \downarrow y$ appearing in the formula and since $\prec$ is a total order on $\mathcal{A}^*$, we will obtain in a finite number of steps, a finite formula involving only binomial coefficients using Lyndon words. □

Example 1.2.10. Considering for instance $x = ab$ and $y = aab$ gives the polynomials

$$\begin{aligned} x \sqcup\!\sqcup y &= abaab + 3aabab + 6aaabb, \\ x \downarrow y &= x \sqcup\!\sqcup y + 4aabb + 3aaab + abab + 2aab + 5\varepsilon. \end{aligned}$$

Since every word in $ab \sqcup\!\sqcup aab$ is less than or equal to $abaab$ for $<_{\text{lex}}$, we have, for any $u \in \mathcal{A}^*$,

$$\begin{aligned} \binom{u}{abaab} = \binom{u}{ab}\binom{u}{aab} - 3\binom{u}{aabab} - 6\binom{u}{aaabb} - 4\binom{u}{aabb} \\ - 3\binom{u}{aaab} - \binom{u}{abab} - 2\binom{u}{aab}. \end{aligned}$$

The word $abab$ is not Lyndon, the process can be applied on this word recursively. We obtain with $x = y = ab$

$$\binom{u}{abab} = \frac{1}{2}\left[\binom{u}{ab}^2 - 4\binom{u}{aabb} - 2\binom{u}{aab} - 2\binom{u}{abb} - \binom{u}{ab} \right]$$

Finally, the coefficient $\binom{u}{abaab}$ can be computed just knowing the values of the coefficients in $ab, aab, abb, aaab, aabb, aaabb, aabab$.

Note that the formulas quickly become long, however this type of result can be useful in reconstruction problems, i.e., when trying to determine a word u from several of its binomial coefficients (the least possible), as done in Chapter 5.

The following proposition gives formulas to compute the first binomial coefficients (using a word of length less than 4) using only Lyndon words.

Proposition 1.2.11. *For any $u \in \mathcal{A}^*$ and $a, b \in \mathcal{A}$ such that $a < b$,*

$$\begin{aligned}
\binom{u}{ba} &= \binom{u}{a}\binom{u}{b} - \binom{u}{ab},\\
\binom{u}{aba} &= \binom{u}{ab}\left[\binom{u}{a} - 1\right] - 2\binom{u}{aab},\\
\binom{u}{baa} &= \left[\binom{u}{a} - 1\right]\left[\frac{1}{2}\binom{u}{a}\binom{u}{b} - \binom{u}{ab}\right] + \binom{u}{aab},\\
\binom{u}{bab} &= \binom{u}{ab}\left[\binom{u}{b} - 1\right] - 2\binom{u}{abb},\\
\binom{u}{bba} &= \left[\binom{u}{b} - 1\right]\left[\frac{1}{2}\binom{u}{a}\binom{u}{b} - \binom{u}{ab}\right] + \binom{u}{abb}.
\end{aligned}$$

Proof. The result is direct using Theorem 1.2.8, Corollary 1.2.9 and noticing that

$$\begin{aligned}
b \sqcup a &= ab + ba \quad \text{and} \quad b \downarrow a = b \sqcup a + \varepsilon,\\
ab \sqcup a &= 2aab + aba \quad \text{and} \quad ab \downarrow a = ab \sqcup a + ab + \varepsilon,\\
b \sqcup aa &= baa + aba + aab \quad \text{and} \quad b \downarrow aa = b \sqcup aa + 2\varepsilon,\\
b \sqcup ab &= 2abb + bab \quad \text{and} \quad b \downarrow ab = b \sqcup ab + ab + \varepsilon,\\
bb \sqcup a &= abb + bab + bba \quad \text{and} \quad bb \downarrow a = bb \sqcup a + 2\varepsilon.
\end{aligned}$$

□

1.2.2 Computing the binomial coefficient of the image of a word

Let σ be a morphism. The goal of this subsection is to compute $\binom{\sigma(u)}{v}$ using binomial coefficients applied on u. The results come from our common paper with Leroy and Rigo [67]. The following proposition is just generalization of Proposition 1.2.2, item 4.

Proposition 1.2.12. *Let $\sigma : \mathcal{A}^* \to \mathcal{A}^*$ be a non-erasing morphism and $u, v \in \mathcal{A}^+$ be two words.*

$$\binom{\sigma(u)}{v} = \sum_{\substack{w_1,\dots,w_{|u|} \in \mathcal{A}^* \\ v = w_1 \cdots w_{|u|}}} \binom{\sigma(u_1)}{w_1} \cdots \binom{\sigma(u_{|u|})}{w_{|u|}}.$$

Note that, in the previous statement, we do not ask that $|w_i| = 1$ for all i and, in particular, we can have $w_i = \varepsilon$ for some i.

Example 1.2.13. Let us consider the following morphism:

$$\sigma : \{0,1\}^* \to \{0,1\}^* : \begin{cases} 0 & \mapsto & 01 \\ 1 & \mapsto & 11. \end{cases}$$

We want to compute $\binom{\sigma(00101)}{0101}$. We thus have to consider all possible factorizations $w_1w_2w_3w_4w_5$ of $v = 0101$. For such a factorization, we want to compute

$$\binom{\sigma(0)}{w_1}\binom{\sigma(0)}{w_2}\binom{\sigma(1)}{w_3}\binom{\sigma(0)}{w_4}\binom{\sigma(1)}{w_5}.$$

Therefore, if w_1, w_2 or w_4 is not in the set $\mathrm{Fac}(\sigma(0)) = \{\varepsilon, 0, 1, 01\}$, the contribution of this factorization to the binomial coefficient $\binom{\sigma(00101)}{0101}$ is zero. For the same reason, we must have $w_3, w_5 \in \mathrm{Fac}(\sigma(1)) = \{\varepsilon, 1, 11\}$. The only interesting factorizations of v are given in Table 1.1. To increase the readability, blank spaces are left when $w_i = \varepsilon$. Moreover, the value of $\binom{\sigma(0)}{w_1}\binom{\sigma(0)}{w_2}\binom{\sigma(1)}{w_3}\binom{\sigma(0)}{w_4}\binom{\sigma(1)}{w_5}$ is computed in any such case. Finally,

w_1	w_2	w_3	w_4	w_5	$\binom{01}{w_1}\binom{01}{w_2}\binom{11}{w_3}\binom{01}{w_4}\binom{11}{w_5}$
0	1		0	1	2
0		1	0	1	4
	0	1	0	1	4
01	0	1			2
01	0		1		1
01	0			1	2
01			0	1	2
	01		0	1	2
0	1		01		1
0		1	01		2
	0	1	01		2
01	01				1
01			01		1
	01		01		1

Table 1.1: Computing $\binom{\sigma(00101)}{0101}$ using Proposition 1.2.12.

$$\binom{\sigma(00101)}{0101} = 27.$$

The next result is a rewriting of the previous one. This time, we do not authorize the empty word ε to be part of the factorizations we are considering. We are therefore looking at factorizations $w_1 \cdots w_k$ of v with $k \leq |v|$. Since for most factorizations, $k < |u|$, a given factorization can be seen in several ways in $\sigma(u)$ (see Example 1.2.15).

Theorem 1.2.14. *Let $\sigma : \mathcal{A}^* \to \mathcal{A}^*$ be a non-erasing morphism and $u, v \in \mathcal{A}^+$ be two words.*

$$\binom{\sigma(u)}{v} = \sum_{k=1}^{|v|} \sum_{\substack{w_1,\ldots,w_k \in \mathcal{A}^+ \\ v = w_1 \cdots w_k}} \sum_{a_1,\ldots,a_k \in \mathcal{A}} \binom{\sigma(a_1)}{w_1} \cdots \binom{\sigma(a_k)}{w_k} \binom{u}{a_1 \cdots a_k}.$$

Proof. The word v occurs as a subword of $\sigma(u)$ if and only if there exists $k \geq 1$ such that v can be factorized into $w_1 \cdots w_k$ where, for all i, w_i is a non-empty subword occurring in $\sigma(a_i)$ for some letter a_i and such that $a_1 \cdots a_k$ is a subword of u. □

Example 1.2.15. Let us continue the previous example. Since every block w_i will be used in a coefficient of the form $\binom{\sigma(a_i)}{w_i}$, the only interesting factorizations will be such that $w_i \in \mathrm{Fac}(\sigma(0)) \cup \mathrm{Fac}(\sigma(1))$ for all i, and are given below:

$$01 \cdot 01, \quad 01 \cdot 0 \cdot 1, \quad 0 \cdot 1 \cdot 01, \quad 0 \cdot 1 \cdot 0 \cdot 1.$$

Moreover, if $w_i = 0$ or $w_i = 01$, the only letter $a_i \in \mathcal{A}$ interesting to pick up is 0. Therefore, we can compute

$$\begin{aligned}
\binom{\sigma(00101)}{0101} = &\binom{\sigma(0)}{01}\binom{\sigma(0)}{01}\binom{00101}{00} \\
&+ \sum_{a_3 \in \mathcal{A}} \binom{\sigma(0)}{01}\binom{\sigma(0)}{0}\binom{\sigma(a_3)}{1}\binom{00101}{00a_3} \\
&+ \sum_{a_2 \in \mathcal{A}} \binom{\sigma(0)}{0}\binom{\sigma(a_2)}{1}\binom{\sigma(0)}{01}\binom{00101}{0a_20} \\
&+ \sum_{a_2,a_4 \in \mathcal{A}} \binom{\sigma(0)}{0}\binom{\sigma(a_2)}{1}\binom{\sigma(0)}{0}\binom{\sigma(a_4)}{1}\binom{00101}{0a_20a_4} = 27.
\end{aligned}$$

As a corollary, we obtain that the difference between two binomial coefficients of exact images by a morphism σ can be expressed using differences between binomial coefficients on u and u'.

Corollary 1.2.16. *With the above notation, if u and u' are two words of the same length, we have*

$$\begin{aligned}
&\binom{\sigma(u)}{v} - \binom{\sigma(u')}{v} \\
&\qquad = \sum_{k=1}^{|v|} \sum_{\substack{w_1,\ldots,w_k \in \mathcal{A}^+ \\ v = w_1 \cdots w_k}} \sum_{a_1,\ldots,a_k \in \mathcal{A}} \binom{\sigma(a_1)}{w_1} \cdots \binom{\sigma(a_k)}{w_k} \left[\binom{u}{a_1 \cdots a_k} - \binom{u'}{a_1 \cdots a_k} \right].
\end{aligned}$$

1.3 Equivalence relations

Let us recall that an *equivalence relation* on a set S is a binary relation

$$\sim : \ S \times S$$

that verifies the following properties:

- (reflexive property) $s \sim s$ for any $s \in S$;
- (symmetric property) $s \sim t \Rightarrow t \sim s$ for any $s, t \in S$;
- (transitive property) $s \sim t, t \sim p \Rightarrow s \sim p$ for any $s, t, p \in S$.

We denote by $[s]_\sim$ the *equivalence class* of s, that is, the set of all elements from S that are equivalent to s. Moreover, if S is a monoid equipped with operation $\cdot$ and if

$$\forall s_1, s_2, t_1, t_2 \in S, \ s_1 \sim s_2 \ \text{ and } \ t_1 \sim t_2 \ \Rightarrow s_1 \cdot t_1 \sim s_2 \cdot t_2,$$

then $\sim$ is called a *congruence (relation)*.

In this section, we will define several equivalence relations on the set of words over an arbitrary alphabet $\mathcal{A}$. The "simplest" equivalence relation is the equality.

Abelian equivalence of words has been investigated for quite a long time; for example in the sixties Erdös raised the question whether Abelian squares (concatenation of two words that are Abelian equivalent) can be avoided by an infinite word over an alphabet of size 4 [24, 30, 106]. A generalization of Abelian equivalence is k-Abelian equivalence where one counts factors of length at most k [58, 59]. Then, an independent but similar generalization of Abelian equivalence was introduced in 2015 by Rigo and Salimov [109]. Let us state the following definitions.

Definition 1.3.1. Let u and v be two finite words over the alphabet $\mathcal{A}$. These words are *Abelian equivalent*, noted $u \sim_{\text{ab}} v$, if

$$|u|_a = |v|_a$$

for all $a \in \mathcal{A}$.

Note that to be Abelian equivalent, two words have to be of the same length. An equivalent definition is that two words are Abelian equivalent if and only if one of them is obtained from the other one by permuting the letters. Moreover, $u \sim_{\text{ab}} v$ if and only if their Parikh vectors are equal.

Definition 1.3.2. Let u and v be two finite words over the alphabet $\mathcal{A}$ and let $k \in \mathbb{N}$. These words are *k-Abelian equivalent*, noted $u \sim_{k,\text{ab}} v$, if

$$|u|_x = |v|_x$$

for all $x \in \mathcal{A}^{\leq k}$. These words are *k-binomially equivalent*, noted $u \sim_{k,\text{bin}} v$, (or simply $u \sim_k v$) if

$$\binom{u}{x} = \binom{v}{x}$$

for all $x \in \mathcal{A}^{\leq k}$.

Remark 1.3.3. For any letter $a \in \mathcal{A}$ and for any $u \in \mathcal{A}^*$, we have $|u|_a = \binom{u}{a}$. Therefore, any two words u and v such that $u \sim_{k,\text{ab}} v$ or $u \sim_{k,\text{bin}} v$ ($k \in \mathbb{N}$) are such that $u \sim_{\text{ab}} v$. Hence, $|u| = |v|$.

Due to the definition, if there is a $k \in \mathbb{N}$ such that $u \sim_{k,\text{ab}} v$, then $u \sim_{k',\text{ab}} v$ for all $k' \leq k$. The same observation holds for k-binomial equivalence.

The next example shows that, *a priori*, there is no comparison possible between $\sim_{k,\text{ab}}$ and $\sim_{k,\text{bin}}$ for a fixed k.

Example 1.3.4. Let $\mathcal{A} = \{a, b, c\}$. We have

$$abacab \sim_{2,\text{ab}} acabab$$

but

$$abacab \not\sim_{2,\text{bin}} acabab$$

because $\binom{abacab}{ab} = 4$ but $\binom{acabab}{ab} = 5$. On the alphabet $\{a, b\}$, we have $abba \sim_{2,\text{bin}} baab$ but $abba \not\sim_{2,\text{ab}} baab$ since $|abba|_{bb} = 1$ but $|baab|_{bb} = 0$.

Lemma 1.3.5. *For any $k \in \mathbb{N}$, k-binomial equivalence is a congruence.*

Proof. Let $u_1, u_2, v_1, v_2 \in \mathcal{A}^*$ such that $u_1 \sim_k u_2$ and $v_1 \sim_k v_2$. Thanks to Proposition 1.2.2(4), we have, for any $t \in \mathcal{A}^{\leq k}$,

$$\binom{u_1 v_1}{t} = \sum_{\substack{xy=t \\ x,y\in\mathcal{A}^*}} \binom{u_1}{x}\binom{v_1}{y} = \sum_{\substack{xy=t \\ x,y\in\mathcal{A}^*}} \binom{u_2}{x}\binom{v_2}{y} = \binom{u_2 v_2}{t}.$$

□

The result is also true for k-Abelian equivalence, but we won't use this property in the present thesis. It suffices to notice that $u \sim_{2,\text{ab}} v$ implies that u and v both start by the same letter and finish by the same letter.

Proposition 1.3.6 (Cancellation property)**.** *Let u, v, w be three words over $\mathcal{A}$. We have*

$$(u \sim_k v \Leftrightarrow u\,w \sim_k v\,w) \quad \textit{and} \quad (u \sim_k v \Leftrightarrow w\,u \sim_k w\,v).$$

Proof. Since $\sim_k$ is a congruence, we only have to prove that the condition is sufficient. Assume that $u \nsim_k v$. There exists a shortest word t, of length at most k, such that

$$\binom{u}{t} \neq \binom{v}{t}.$$

We compute

$$\binom{uw}{t} = \sum_{\substack{rs=t \\ r,s\in\mathcal{A}^*}} \binom{u}{r}\binom{w}{s} = \binom{u}{t} + \sum_{\substack{rs=t \\ r,s\in\mathcal{A}^+}} \binom{u}{r}\binom{w}{s} + \binom{w}{t}. \tag{1.1}$$

In the above formula, $\binom{u}{r} = \binom{v}{r}$ for all r shorter than t. Hence, we get exactly the same decomposition for $\binom{vw}{t}$ except for the first term. Thus,

$$\binom{uw}{t} - \binom{vw}{t} = \binom{u}{t} - \binom{v}{t} \neq 0.$$

This means that $uw \nsim_k vw$. Proceed similarly for the second equivalence or observe that

$$\binom{\widetilde{x}}{\widetilde{y}} = \binom{x}{y}$$

where $\widetilde{x}$ is the mirror[5] of x. □

We finish with a practical result that will be used in Chapter 3.

Proposition 1.3.7. *Let $k \geq 2$. Let $u, v, u', v' \in \mathcal{A}^*$ be such that $u \sim_{k-1} u'$ but $u \not\sim_k u'$, and $v \sim_k v'$. Then, $uv \not\sim_k u'v'$.*

Proof. There exists $t \in \mathcal{A}^k$ such that $\binom{u}{t} \neq \binom{u'}{t}$. Thus

$$\begin{aligned}
\binom{uv}{t} &= \binom{u}{t} + \sum_{\substack{rs=t \\ r,s\in\mathcal{A}^+}} \binom{u}{r}\binom{v}{s} + \binom{v}{t} \\
&= \binom{u}{t} + \sum_{\substack{rs=t \\ r,s\in\mathcal{A}^+}} \binom{u'}{r}\binom{v'}{s} + \binom{v'}{t} \\
&\neq \binom{u'}{t} + \sum_{\substack{rs=t \\ r,s\in\mathcal{A}^+}} \binom{u'}{r}\binom{v'}{s} + \binom{v'}{t} = \binom{u'v'}{t}.
\end{aligned}$$

□

In the next chapter, we will study the k-binomial equivalence and the number of equivalence classes.

[5]The mirror (or reversal) of a finite word $x_1 \cdots x_n$ is the word $x_n \cdots x_1$.

1.4 Complexity functions

There is a direct link between an equivalence relation on finite words and the complexity function of an infinite word. First of all, a *complexity function* of an infinite word $\mathbf{w} \in \mathcal{A}^{\mathbb{N}}$ is a function whose domain is $\mathbb{N}_0$ and whose range is included in $\mathbb{N}$, establishing a link between natural numbers n and factors of $\mathbf{w}$ of length n.

1.4.1 Factor complexity function

The most famous complexity function is the factor complexity, as defined in Definition 1.1.10. It can be rewritten using an equivalence relation. Let us denote by $\sim_=$ the trivial equality equivalence relation: two words are equivalent if and only if they are equal. The factor complexity of an infinite word $\mathbf{w}$ is the function

$$p_{\mathbf{w}} : n \mapsto \#\left(\mathrm{Fac}_n(\mathbf{w}) / \sim_=\right).$$

1.4.2 Other complexity functions

Similarly, we can introduce Abelian, k-Abelian and k-binomial complexity functions, following what was done in [29, 107], [58] and [109].

Definition 1.4.1. The *Abelian complexity (function)* of an infinite word $\mathbf{w}$ is the function

$$\rho_{\mathbf{w}} : n \mapsto \#\left(\mathrm{Fac}_n(\mathbf{w}) / \sim_{\mathrm{ab}}\right).$$

Let $k \in \mathbb{N}$. The *k-Abelian complexity (function)* of an infinite word $\mathbf{w}$ is the function

$$\rho_{\mathbf{w}}^{(k)} : n \mapsto \#\left(\mathrm{Fac}_n(\mathbf{w}) / \sim_{k,\mathrm{ab}}\right).$$

The *k-binomial complexity (function)* of an infinite word $\mathbf{w}$ is the function

$$b_{\mathbf{w}}^{(k)} : n \mapsto \#\left(\mathrm{Fac}_n(\mathbf{w}) / \sim_{k,\mathrm{bin}}\right).$$

The different remarks made on equivalence relations in the previous section lead to the following proposition.

Proposition 1.4.2. *Let $\mathbf{w}$ be an infinite word. We have the following sequences of inequalities: for any $n \in \mathbb{N}_0$,*

$$\rho_{\mathbf{w}}(n) \leq \rho_{\mathbf{w}}^{(2)}(n) \leq \cdots \leq \rho_{\mathbf{w}}^{(k)}(n) \leq \rho_{\mathbf{w}}^{(k+1)}(n) \leq \cdots \leq p_{\mathbf{w}}(n)$$

and

$$\rho_{\mathbf{w}}(n) \leq b_{\mathbf{w}}^{(2)}(n) \leq \cdots \leq b_{\mathbf{w}}^{(k)}(n) \leq b_{\mathbf{w}}^{(k+1)}(n) \leq \cdots \leq p_{\mathbf{w}}(n).$$

Let us conclude this chapter by making an overview of known results about these functions. First, since the factor complexity was introduced more than eighty years ago, a lot of results could be mentioned. As we have already seen, the Morse–Hedlund theorem (Theorem 1.1.12) gives a characterization of periodic words in terms of their factor complexity. Another result is due to Pansiot [93]. Recall that $f(n) \in \Theta(g(n))$ means that there exist $C, D \in \mathbb{R}$ such that for all n large enough, $Cg(n) \leq f(n) \leq Dg(n)$. Pansiot's theorem characterizes all possible behaviors of $p_{\mathbf{w}}(n)$ when $\mathbf{w}$ is a purely morphic word.

Theorem 1.4.3 (Pansiot)**.** *Let* $\mathbf{w}$ *be a purely morphic word. One of the following holds:*

- $p_{\mathbf{w}}(n) \in \Theta(1)$;
- $p_{\mathbf{w}}(n) \in \Theta(n)$;
- $p_{\mathbf{w}}(n) \in \Theta(n \log\log n)$;
- $p_{\mathbf{w}}(n) \in \Theta(n \log n)$, *or;*
- $p_{\mathbf{w}}(n) \in \Theta(n^2)$.

The initial proof given in [93] being difficult to read, a whole section is devoted to it in [15, Section 4.7]. For example, the Thue–Morse word has a factor complexity in $\Theta(n)$, as recalled in Chapter 3.

Abelian and k-Abelian complexity were studied for a large range of words, as for example for the Thue–Morse word [51] and its generalizations over larger alphabets [26, 55], the Tribonacci word [107, 122], the paperfolding word [79], the Rudin–Shapiro sequence [78], the Cantor sequence [25], binary uniform morphic words [16], 2-automatic sequences [95],...

Finally, less things are known for k-binomial complexity and this is why this manuscript is partially devoted to its study. Let us mention two important results coming from [109].

Proposition 1.4.4. *Let* $\mathbf{w}$ *be a Sturmian word. For any* $k \geq 2$ *and any* $n \in \mathbb{N}_0$*, we have*

$$b_{\mathbf{w}}^{(k)}(n) = p_{\mathbf{w}}(n) = n + 1.$$

The proof is quite simple, since due to Proposition 1.4.2, it suffices to show that two different factors of $\mathbf{w}$ are non-2-binomially equivalent.

The second result concerns a whole family of words, namely *fixed points of Parikh-constant morphism*. Let $\sigma : \mathcal{A}^* \to \mathcal{A}^*$ be a morphism. If $|\sigma(a)|_c = |\sigma(b)|_c$ for any $a, b, c \in \mathcal{A}$, then σ is called a Parikh-constant morphism. An equivalent condition is that all images $\sigma(a)$ have the same Parikh vector.

Proposition 1.4.5. *Let* $\mathbf{w}$ *be a fixed point of a Parikh-constant morphism and let* $k \geq 2$*. There exists a constant* $C_{\mathbf{w},k} \in \mathbb{N}$ *such that*

$$b_{\mathbf{w}}^{(k)}(n) \leq C_{\mathbf{w},k},$$

for all $n \in \mathbb{N}_0$*.*

From this result we can already deduce that the Thue–Morse word has a k-binomial complexity bounded by a constant depending on k, and thus a generalization of the Morse–Hedlund theorem does not seem to be evident in the k-binomial case: there exist infinite words that are aperiodic but have a bounded k-binomial complexity. The exact value of $b_{\mathbf{t}}^{(k)}(n)$ will be computed in details in Chapter 3.

2 | k-binomial equivalence classes of finite words

The goal of this chapter is to study the equivalence classes induced by the k-binomial equivalence. Section 2.2 deals with the easiest case, that is $k = 2$ and $\#\mathcal{A} = 2$. We then study 2-binomial equivalence on alphabets of at least three letters, and provide in Subsection 2.3.2 an algorithm generating the 2-binomial equivalence class of a word. For $k \geq 2$ and an alphabet of 3 or more symbols, the language made of lexicographically least elements of every k-binomial equivalence class and the language of singletons, both defined in Subsection 2.1.5, are shown to be non-context-free (see Section 2.5 and more especially Theorem 2.5.3). The latter notion will be defined in Subsection 2.1.3. As a consequence of our discussions, we also prove in Subsection 2.3.3 that the submonoid generated by the generators of the free nil-2 group (also called free nilpotent group of class 2) on m generators is isomorphic to the quotient of the free monoid $\{1, \ldots, m\}^*$ by the 2-binomial equivalence. Most of the important results of this chapter are taken from [69] by Marie Lejeune, Michel Rigo and Matthieu Rosenfeld. They appeared in International Journal of Algebra and Computation. The work is based on similar question asked for k-Abelian equivalence by Whiteland [125]. However, in the k-Abelian case, the two studied languages are shown to be regular.

Contents

2.1 Classical notions

This chapter will deal with several types of languages. A language is just a set of words. Formally, a *language over* $\mathcal{A}$ is a subset of $\mathcal{A}^*$. One may consider different subclasses of languages. We will introduce those that will be useful in this work. They involve the notion of automaton.

2.1.1 Automata

Even if in combinatorics on words, automata are a classical notion, let us briefly recall what an automaton is. The interested reader can consult Chapter 1 of the following reference books [15, 77, 110].

Definition 2.1.1. A *deterministic finite automaton (DFA)* is a 5-tuple $M = (Q, \mathcal{A}, \delta, q_0, F)$ consisting of

- a finite set of *states* Q;
- a finite *alphabet* $\mathcal{A}$;
- a *transition function* $\delta : Q \times \mathcal{A} \to Q$;
- an *initial state* $q_0 \in Q$;
- a set of *final states* $F \subseteq Q$.

If $q \in Q$ is a state which is not final (i.e., $q \notin F$) and having no outgoing transition (i.e., $\delta(q, a) = q$ for all $a \in \mathcal{A}$), then q is called a *sink state*.

Let $u = u_1 \cdots u_n \in \mathcal{A}^*$ be a word. We say that the word u is *accepted* by the automaton M if there exists a sequence of states $(p_i)_{i=0}^n$ of Q such that

- $p_0 = q_0$;

- $p_{i+1} = \delta(p_i, u_{i+1})$ for all $i \in \{0, \dots, n-1\}$;
- $p_n \in F$.

The *language accepted by* M, denoted $\mathcal{L}(M)$, is the set of words of $\mathcal{A}^*$ accepted by M.

We often represent DFAs in the following manner: states are drawn into circles, and the transition $\delta(p, a) = q$ is represented by an arrow from p to q and labelled by a. Moreover, the initial state q_0 is distinguished by an incoming arrow without any label, and final states are circled twice. For more readability, automata are trim: we do not represent sink states and their incoming transitions.

Example 2.1.2. Figure 2.1 represents a DFA

$$(\{0,1,2\}, \{a,b,c\}, \delta, 0, \{0\})$$

where

$$\delta : \begin{cases} (0,a) & \mapsto & 1; \\ (0,b) & \mapsto & 0; \\ (0,c) & \mapsto & 2; \\ (1,a) & \mapsto & 0; \\ (1,b) & \mapsto & 1; \\ (1,c) & \mapsto & 2; \\ (2,a) & \mapsto & 2; \\ (2,b) & \mapsto & 2; \\ (2,c) & \mapsto & 2. \end{cases}$$

Since 2 is a sink state, it is not represented. The accepted language is the following set:

$$\{u \in \{a,b,c\}^* \ : \ |u|_c = 0 \text{ and } |u|_a \in 2\mathbb{N}_0\}.$$

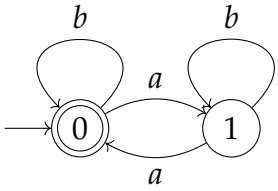


Figure 2.1: A DFA accepting all words from $\{a,b,c\}$ having no c and an even number of a.

2.1.2 Regular languages

We recall two definitions of regular languages. The reader can find more details in [38] or [110].

Definition 2.1.3. A language $L \subseteq \mathcal{A}^*$ is *recognizable* (or *regular*) if it is accepted by a deterministic finite automaton.

An equivalent definition is that a recognizable language can be represented by a rational expression and every rational expression represent a recognizable language.

Definition 2.1.4. A *rational expression* over the alphabet $\mathcal{A}$ is an expression representing a set of words from $\mathcal{A}^*$ consisting of constants and rational operations, defined as follows. Constants are ε, representing the empty word, and expressions of the type `a` representing letters from $\mathcal{A}$. Then, rational operations are the following:

- (union) If `R` and `S` are rational expressions representing languages A and B, then `R + S` is a rational expression representing the language $A \cup B$;
- (concatenation) If `R` and `S` are rational expressions representing languages A and B, then `RS` is a rational expression representing the language $A \cdot B = \{uv : u \in A, v \in B\}$;
- (Kleene star) If `R` is a rational expression representing the language A, then `R`* is a rational expression representing the language $\{\varepsilon\} \cup \{u_1 \cdots u_n : n \in \mathbb{N}, u_i \in A \, \forall i\}$.

Example 2.1.5. The language of words from $\{a, b, c\}$ that contain no c and an even number of a is represented by the following rational expression:

$$\texttt{(b}^*\texttt{ab}^*\texttt{ab}^*\texttt{)}^*.$$

Proposition 2.1.6. *[110, Propositions 2.1 and 2.7] A language is recognizable by a finite automaton if and only if it is represented by a rational expression.*

2.1.3 Context-free languages

Context-free languages are more general than regular languages in the following sense: every regular language is context-free, but the converse does not hold. We first need to introduce the notion of context-free grammar. More details can be found in [6].

A context-free grammar is a type of rewriting system on two kinds of symbols: variables and terminals. Whenever a variable appears, it can be rewritten using one of the rewriting rules. Terminals, however, are never rewritten, and when the rewriting rules produce a string of all terminals, the process stops.

Definition 2.1.7. A *context-free grammar* is a 4-tuple

$$G = (V, \mathcal{A}, P, S),$$

where

- V is a finite set of elements, called *variables*;
- $\mathcal{A}$ is a finite alphabet, whose letters are called *terminals*;
- P is a finite set of rewriting rules, called *productions*;
- S is a particular variable from V, called the *start symbol*.

A production is a pair of the form $(R, \alpha) \in V \times (V \cup \mathcal{A})^*$, often written in the following form: $R \longrightarrow \alpha$.

If $R \longrightarrow \alpha$ is a production of G and if $\beta R \gamma$ is an element of $(V \cup \mathcal{A})^*$, then we may replace R with α in $\beta R \gamma$, to obtain $\beta \alpha \gamma$. In this case we write

$$\beta R \gamma \Longrightarrow \beta \alpha \gamma.$$

If $\alpha_1 \Longrightarrow \alpha_2 \Longrightarrow \cdots \Longrightarrow \alpha_n$ for some $n \in \mathbb{N}$, we write $\alpha_1 \overset{n-1}{\Longrightarrow} \alpha_n$. If $\alpha \overset{n}{\Longrightarrow} \beta$ for some $n \geq 0$, we write $\alpha \overset{*}{\Longrightarrow} \beta$.

We can now define context-free languages.

Definition 2.1.8. The *language generated by a context-free grammar* $G = (V, \mathcal{A}, P, S)$ is defined to be the set of terminal words that can be produced thanks to G:

$$\mathcal{L}(G) = \{u \in \mathcal{A}^* \,:\, S \overset{*}{\Longrightarrow} u\}.$$

A *context-free language* is a language generated by a context-free grammar.

Let us illustrate the concept on an example.

Example 2.1.9. Let G be the following grammar: $G = (\{S\}, \{a, b\}, P, S)$ with $P = \{S \to aSb, S \to ab\}$. Then, the language generated by G is $\{a^n b^n \,:\, n \geq 1\}$. Indeed, it is easy to see that every word generated by G must be of this form. Moreover, all such words are generated by G since, for any $n \geq 1$,

$$S \Longrightarrow aSb \Longrightarrow a^2Sb^2 \Longrightarrow \cdots \Longrightarrow a^{n-1}Sb^{n-1} \Longrightarrow a^n b^n.$$

As announced before, context-free languages are more general than regular languages.

Proposition 2.1.10. *Any context-free language is a regular language. The converse does not hold.*

The proof of this result is not in the scope of this work. It is shown that context-free languages are exactly languages accepted by pushdown automata, that form a more general class of automata. The interested reader can find the proof in [9, Theorem 5.1].

2.1.4 Growth function of a language

Definition 2.1.11. Let $L \subset \mathcal{A}^*$ be a language. The *growth function* of L is the function mapping every n to the number of words of length n in L:

$$g_L : n \in \mathbb{N}_0 \mapsto \#(L \cap \mathcal{A}^n) \in \mathbb{N}_0.$$

The growth function can be used to divide languages into subclasses.

Definition 2.1.12. Let $L \subseteq \mathcal{A}^*$ be a language. If there exists a polynomial $P(x)$ such that $g_L(n) \leq P(n)$ for all $n \in \mathbb{N}_0$, then L has *polynomial growth*. If there exists a real number $r > 1$ such that $g_L(n) \geq r^n$ for infinitely many $n \in \mathbb{N}_0$, then L has *exponential growth*.

Context-free languages do not accept intermediate growths, as stated in the following proposition.

Proposition 2.1.13. *[54] A context-free language has either polynomial growth, or exponential growth.*

The previous proposition allows us to divide context-free languages into two subclasses: those having polynomial growth, and those having exponential growth. Languages of the first subclass share an alternative property.

Definition 2.1.14. A language $L \subseteq \mathcal{A}^*$ is *bounded* if there exist words $u_1, \ldots, u_\ell \in \mathcal{A}^*$ such that $L \subseteq u_1^* \ldots u_\ell^*$.

Proposition 2.1.15. *A context-free language is bounded if and only if it has polynomial growth.*

This result has been independently discovered at least six times [121, 64, 53, 104, 54, 17]. Moreover, several of these results lead to the proof of Proposition 2.1.13.

2.1.5 Two particular languages

Let $(\mathcal{A}, <)$ be an ordered alphabet and let $\sim$ be an equivalence relation on $\mathcal{A}^*$. In what follows we will be particularly interested in two types of subsets of $\mathcal{A}^*$ with respect to $\sim$. We let

$$\mathsf{LL}(\sim, \mathcal{A}) = \{u \in \mathcal{A}^* \mid \forall v \in [u]_\sim : u \leq_{\text{lex}} v\}$$

denote the language of lexicographically least elements of every $\sim$-equivalence class. So there is a one-to-one correspondence between $\mathsf{LL}(\sim, \mathcal{A})$ and $\mathcal{A}^*/\sim$. We let

$$\mathsf{Sing}(\sim, \mathcal{A}) = \{u \in \mathcal{A}^* \mid \#[u]_\sim = 1\}$$

denote the language consisting of the so-called $\sim$-*singletons*, i.e., the elements whose $\sim$-equivalence class is restricted to a single element. Clearly, we have $\mathsf{Sing}(\sim, \mathcal{A}) \subseteq \mathsf{LL}(\sim, \mathcal{A})$. These sets have been extensively studied in [23]. Based on an operation of k-switching, the following result is given.

Theorem 2.1.16. *Let $k \geq 1$ and let $\mathcal{A}$ be a m-letter alphabet. For k-Abelian equivalence, both languages $\mathsf{LL}(\sim_{k,\mathrm{ab}}, \mathcal{A})$ and $\mathsf{Sing}(\sim_{k,\mathrm{ab}}, \mathcal{A})$ are regular.*

We want to answer the following question: are the languages $\mathsf{LL}(\sim_k, \mathcal{A})$ and $\mathsf{Sing}(\sim_k, \mathcal{A})$ regular?

2.2 2-binomial equivalence over a 2-letter alphabet

Let $\mathcal{A}$ be the ordered alphabet $\{1, 2\}$ in this section. In this case, the answer is positive and can be easily found by making a parallel between these languages and the set of M-unambiguous words.

Definition 2.2.1. The *Parikh matrix* associated with a word $u \in \{1, 2\}^*$ is the 3×3 matrix given by

$$P(u) = \begin{pmatrix} 1 & |u|_1 & \binom{u}{12} \\ 0 & 1 & |u|_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

See [83] for a longer introduction on Parikh matrices. For $a, b \in \{1, 2\}$, $\binom{u}{ab}$ can be deduced from $P(u)$. Indeed, we have $\binom{u}{aa} = \binom{|u|_a}{2}$ and if $a \neq b$,

$$\binom{u}{aa} + \binom{u}{ab} + \binom{u}{ba} + \binom{u}{bb} = \binom{|u|_a + |u|_b}{2}. \tag{2.1}$$

It is thus clear that $u \sim_2 v$ if and only if $P(u) = P(v)$. We can therefore make use of the following theorem of Fossé and Richomme [42]. If two words u and v over an arbitrary alphabet $\mathcal{B}$ can be factorized as $u = xabybaz$ and $v = xbayabz$ with $a, b \in \mathcal{B}$, $x, y, z \in \mathcal{B}^*$, we write $u \equiv v$. The reflexive and transitive closure of this relation is denoted by $\equiv^*$: let $u, v \in \mathcal{A}^*$, we write $u \equiv^* v$ if $u = v$, if $u \equiv v$ or if there exist $n \in \mathbb{N}$, $u_1, \ldots, u_n \in \mathcal{A}^*$ such that $u \equiv u_1$, $u_i \equiv u_{i+1}$ for all $i \in [n-1]$ and $u_n \equiv v$.

Theorem 2.2.2. *Let u, v be two words over $\{1, 2\}$. The following assertions are equivalent:*

- *the words u and v have the same Parikh matrix;*
- *the words u and v are 2-binomially equivalent;*
- $u \equiv^* v$.

Consequently, the language $\mathsf{Sing}(\sim_2, \{1,2\})$ is composed of words avoiding patterns $12u21$ and $21u12$, $u \in \mathcal{A}^*$. This language is regular since a regular expression is given by

$$1^*2^* + 2^*1^* + 1^*21^* + 2^*12^* + 1^*212^* + 2^*121^*.$$

A non-deterministic finite automaton accepting $\mathsf{Sing}(\sim_2, \{1,2\})$ was given in [112].

Let us now consider the language $\mathsf{LL}(\sim_2, \{1,2\})$.

Remark 2.2.3. From [109], we know that

$$\#\mathsf{LL}(\sim_2, \{1,2\}) = \#\left(\{1,2\}^n / \sim_2\right) = \frac{n^3 + 5n + 6}{6}.$$

Note that this is exactly the OEIS sequence `A000125` of *cake numbers*, i.e., the maximal number of pieces resulting from n planar cuts through a cube.

Proposition 2.2.4. *The language* $\mathsf{LL}(\sim_2, \{1,2\})$ *is regular.*

Proof. As a consequence of Theorem 2.2.2, if a word u belongs to $\mathsf{LL}(\sim_2, \{1,2\})$, it cannot be of the form $x21y12z$ because otherwise, the word $x12y21z$ belongs to the same class and is lexicographically less. Consequently,

$$\mathsf{LL}(\sim_2, \{1,2\}) \subseteq \{1,2\}^* \setminus \{1,2\}^*21\{1,2\}^*12\{1,2\}^*.$$

It suffices to check that the language in the right-hand side has exactly $(n^3 + 5n + 6)/6$ words of length n. Indeed, in this case with the previous remark the inclusion would become an equality and thus $\mathsf{LL}(\sim_2, \{1,2\})$ will be regular. Let us denote by $C(n)$ the number of words of length n from

$$L := \{1,2\}^* \setminus \{1,2\}^*21\{1,2\}^*12\{1,2\}^*,$$

and set for any n, $L_n = L \cap \mathcal{A}^n$. We show by induction that $C(n) = (n^3 + 5n + 6)/6$. This is evident for $n \leq 4$. Then, let us fix n and assume that $\#L_n = C(n)$. Any word of L_{n+1} is such that its prefix of length n is in L_n. In addition, any word $u \in L_n$ gives $u \cdot 1 \in L_{n+1}$ in all cases, and $u \cdot 2 \in L_{n+1}$ if and only if

- the last letter of u is 2, or;
- the last letter of u is 1, but there is no occurrence of 21 in u in positions[1] 1 to $n-2$ (u can finish by 21 since the presence of 212 is not forbidden in L).

Hence $C(n+1) = 2C(n) - D(n)$, where $D(n)$ denotes the number of words of

$$(1^*2^* \cap \mathcal{A}^{i-1})21(2^*1^* \cap \mathcal{A}^{n-i-2})1, \quad i \in [n-2].$$

[1] We say that 21 occurs in u in position i if $u_i u_{i+1} = 21$.

It gives

$$D(n) = \sum_{i=1}^{n-2} i(n-1-i)$$

and after some computations we get

$$C(n+1) = \frac{(n+1)^3 + 5(n+1) + 6}{6},$$

hence the conclusion. □

2.3 2-binomial equivalence over a m-letter alphabet

Theorem 2.2.2 does not hold for ternary or larger alphabets. Indeed, the two words 1223312 and 2311223 are 2-binomially equivalent but both words belong to $\mathsf{Sing}(\equiv, \{1,2,3\})$ which means that $1223312 \not\equiv^* 2311223$. However, we still have that $u \equiv^* v$ implies $u \sim_2 v$. It is therefore meaningful to study $\sim_2$ over larger alphabets and to describe the 2-binomial equivalence classes.

The first few terms of $(\#(\{1,2,3\}^n / \sim_2))_{n\in\mathbb{N}_0}$ are given by

$$1, 3, 9, 27, 78, 216, 568, 1410, \ldots.$$

This sequence also appears in the Sloane's encyclopedia as entry `A140348` which is the growth function for the submonoid generated by the generators of the free nil-2 group on three generators (if we stick to the terminology of this entry in the encyclopedia). In this section, we make explicit the connection between these two notions (see Theorem 2.3.16).

2.3.1 Free nil-2 group on m generators

Let us first present the well-known notion of free groups. The following definitions and results are extracted from [97]. The interested reader can also consult [8, Chapter 10]. Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be an alphabet. Let $\mathcal{A}^{-1} = \{a_1^{-1}, \ldots, a_n^{-1}\}$ be a set of elements, distinct from each other and from the elements of $\mathcal{A}$. We can think of the set $\mathcal{A} \cup \mathcal{A}^{-1}$ as an alphabet too. We call *inverse pairs* words of the form $a_i a_i^{-1}$ or $a_i^{-1} a_i$, $i \in [n]$. A word of $(\mathcal{A} \cup \mathcal{A}^{-1})^*$ is said to be *reduced* if it contains no inverse pair. Let us denote by F_n the set of reduced words of $(\mathcal{A} \cup \mathcal{A}^{-1})^*$. Let us add an operation on F_n to form a group: we define the product of two elements of F_n as the concatenation of these two words, followed by deleting all the inverse pairs. Hence the product is closed on F_n. Then F_n is a group, where the identity is the empty word ε and the inverse of $u_1 \cdots u_\ell$ is $u_\ell^{-1} \cdots u_1^{-1} \in (\mathcal{A} \cup \mathcal{A}^{-1})^*$, with the convention that for any $a_i^{-1} \in \mathcal{A}^{-1}$, $(a_i^{-1})^{-1} = a_i \in \mathcal{A}$. Any group isomorphic to F_n for a positive n is called

a *free group on n generators*. As stated by the following theorem, every group can be seen as the quotient group of a free group.

Theorem 2.3.1. *[97, Theorem 1] Every (finitely generated) group is the quotient group of a (finitely generated) free group by a normal subgroup.*

Let us now introduce the notion of presentation of a group, which we will use to describe the particular group we are interested in.

Definition 2.3.2. Let G be a group. A *presentation of* G is a set of generators $\mathcal{A}$ of a free group F_n and a set of relators R, words in F_n, such that the quotient group of F_n by the normal closure[2] of R is isomorphic to G. We write $G =< \mathcal{A} \mid R >$.

Remark 2.3.3. It can be easily deduced from Theorem 2.3.1 that every presentation defines a group and that every group has a presentation.

Another way of thinking about group G, given by a presentation $< \mathcal{A} \mid R >$, is as a set of equivalence classes. If $r \in (\mathcal{A} \cup \mathcal{A}^{-1})^*$ is a relator of R, we want in the group G the relation $r = \varepsilon$. So if any word contains the factor r it is equivalent to the word with the occurrence of r deleted. Group G can be defined as the set $(\mathcal{A} \cup \mathcal{A}^{-1})^*$ with equivalence classes given by insertions or deletions of inverse pairs, along with the equivalences defined by the relators of R.

Remark 2.3.4. Let G be a group of presentation $< \mathcal{A} \mid \{r\} >$. Let $r_1 \cdots r_\ell = r$. As we have seen before, relator r defines the relation $r = \varepsilon$ that we want to hold true in G. Hence, $< \mathcal{A} \mid r = \varepsilon >$ is a valid presentation for G, but for every $i \in [\ell - 1]$, G also admits

$$< \mathcal{A} \mid r_1 \cdots r_i = r_\ell^{-1} \cdots r_{i+1}^{-1} >$$

as a presentation.

Definition 2.3.5. The *commutator* of two elements x, y belonging to a multiplicative group $(G, \cdot)$ is $[x, y] = x^{-1}y^{-1}xy$. Hence, the following relations hold

$$xy = yx[x, y] \quad \forall x, y \in G.$$

A *group of nilpotency class* 2, or *nil*-2 group for short, is a group G for which the commutators belong to the center[3] $Z(G)$, i.e.,

$$[x, y]z = z[x, y] \quad \forall x, y, z \in G. \tag{2.2}$$

[2]The normal closure of R is the intersection of all normal subgroups of F_n that contain elements of R. It is a normal subgroup.

[3]The center of a group is the set of elements that commute with the whole group.

Let now $\mathcal{A}$ be the alphabet $\{1, \ldots, m\}$ in this section. The free nil-2 group on m generators has thus a presentation

$$N_2(\mathcal{A}) = \langle \mathcal{A} \mid [x,y]z = z[x,y] \ (x,y,z \in \mathcal{A}) \rangle .$$

As an example, making use of these relations, let us show two elements of the free group on $\{1,2,3\}$ that are equivalent in $N_2(\{1,2,3\})$:

$$12321 = (12[2,1])[1,2]321 = 21[1,2]321 = 213(21[1,2]) = 21312.$$

Since $N_2(\mathcal{A})$ is the quotient of the free monoid $\left(\mathcal{A} \cup \mathcal{A}^{-1}\right)^*$ under the congruence relations generated by $xx^{-1} = \varepsilon$ and (2.2), we will consider the natural projection denoted by

$$\pi : \left(\mathcal{A} \cup \mathcal{A}^{-1}\right)^* \to N_2(\mathcal{A}). \tag{2.3}$$

In Subsection 2.3.2, we provide an algorithmic description of any 2-binomial class. We make use of this description in Subsection 2.3.3 to show that the monoid $\mathcal{A}^*/\sim_2$ is isomorphic to the submonoid, generated by $\mathcal{A}$, of the nil-2 group $N_2(\mathcal{A})$.

2.3.2 A nice tree generating the $\sim_2$ class of a word

Let u be a word over $\mathcal{A}$ and ℓ be the lexicographically least element in its Abelian equivalence class, i.e.,

$$\ell = 1^{|u|_1} 2^{|u|_2} \cdots m^{|u|_m}.$$

We present an algorithm that, given u, produces a finite sequence of words $\mathcal{L}_u = (\ell_i)_i$ starting with $\ell_0 = \ell$, ending with u and such that two consecutive words in the sequence differ only by exchanging two adjacent symbols. This algorithm will be used to build a graph that will next be restricted to a rooted tree encoding sequences $\mathcal{L}_u$ and whose edges are labelled by words from $\mathcal{A}^2$. Moreover it will have the property that two words u and v are 2-binomially equivalent if and only if they appear in the same tree at the same level with the same number of edges of each type. More details will be given in Definitions 2.3.10, 2.3.11 and Proposition 2.3.12.

Recall that we denote $\Lambda(x,y)$ the longest common prefix of two finite words x and y. We also use the following notation coming from the definition of inverse pairs in the free group: if $x = ya$, $a \in \mathcal{A}$, then $xa^{-1} = y$. The algorithm is given in Figure 2.2.

Let us give some additional explanations. The idea is that ℓ_{i+1} is obtained from ℓ_i by a single swap of two adjacent letters $ab \mapsto ba$ with $a < b$, in such a way that $|\Lambda(\ell_i, u)| \leq |\Lambda(\ell_{i+1}, u)|$.

$\mathrm{L}(u)$:

1: $\ell = 1^{|u|_1}2^{|u|_2}\cdots m^{|u|_m}$
2: $L = [\ell]$
3: $p = \Lambda(\ell, u)$
4: **while** $p \neq u$ **do**
5: d = letter following p in u
6: x = word s.t. $\ell = px$
7: y = word s.t. $u = pdy$
8: v = longest prefix of x containing no letter d
9: w = word s.t. $x = vdw$
10: **for** $i \in [|v|]$ **do**
11: c' = last letter of v
12: $v = vc'^{-1}$
13: $w = c'w$
14: $\ell = pvdw$
15: $L = L + [\ell]$
16: **end for**
17: $p = \Lambda(\ell, u)$
18: **end while**
19: **return** L

Figure 2.2: An algorithm producing the sequence $\mathcal{L}_u$.

Assume that we have already obtained $\ell_0, \ldots, \ell_i$ and let $p = \Lambda(\ell_i, u)$. If $p = u$, then we are done. Otherwise, let us denote by d the letter following p in u. There exist $x, y \in \mathcal{A}^*$ such that $\ell_i = px, u = pdy$. Since d appears in x, let us consider its leftmost occurrence. There exist $v, w \in \mathcal{A}^*$ such that $x = vdw$ with $|v|_d = 0$. Moreover, since $p = \Lambda(\ell_i, u)$ and by definition of d, $|v| \geq 1$. It can easily be shown by induction that the word cx is the least lexicographical word of its Abelian class. It follows that, if c denotes the first letter of v, $c < d$ and v only contains letters less than d.

To move the letter d in front of the word v, perform $|v|$ swaps of the form $c'd \mapsto dc'$ with $c' < d$ (c' is a letter occurring in v), as done in lines 13 to 15. This defines $\ell_{i+1}, \ldots, \ell_{i+|v|}$. More precisely, $\ell_{i+j} = pv_1 \cdots v_{|v|-j}dv_{|v|-j+1} \cdots v_{|v|}w$ for $j \in [|v|]$. Note that $|\Lambda(\ell_{i+|v|}, u)|$ is at least $|\Lambda(\ell_i, u)| + 1$. Indeed, with these swaps, the $(|p|+1)^{\text{th}}$ letter of $\ell_{i+|v|}$ is now a d, as in u.

Remark 2.3.6. The letter d has been swapped several times until it reaches a position that corresponds to its position in u. We stress the fact that afterwards, any other

letter that will be swapped by the algorithm will not affect the prefix pd. This remark will allow us to define a graph $\mathcal{G}_u$ that will then be restricted to a tree.

To obtain the complete sequence $\mathcal{L}_u$, we iterate the previous process until we reach u.

Example 2.3.7. Take the word $u = 1223312$. If we apply the above algorithm, we get the sequence

$$\mathcal{L}_u = (\ell_0 = 1122233,\ \ell_1 = 1212233,\ \ell_2 = 1221233,\ \ell_3 = 1221323,\\ \ell_4 = 1223123,\ \ell_5 = 1223132,\ \ell_6 = 1223312).$$

Using (2.1), the next lemma is obvious.

Lemma 2.3.8. *Two Abelian equivalent words u, v are 2-binomially equivalent if and only if $\binom{u}{ab} = \binom{v}{ab}$ for all $a, b \in \mathcal{A}$ with $a < b$. Let ℓ be a word in $1^*2^* \cdots m^*$. In the set of tuples of size $m(m-1)/2$*

$$\left\{ \left(\binom{u}{12}, \ldots, \binom{u}{1m}, \binom{u}{23}, \ldots, \binom{u}{2m}, \ldots, \binom{u}{(m-1)m} \right) \;\middle|\; u \in [\ell]_{\sim_1} \right\},$$

the greatest element, for the lexicographical ordering, is achieved for $u = \ell$.

We consider the $m(m-1)/2$ coefficients $\binom{u}{ab}$ with $a < b$. Note that, in the algorithm, if ℓ_{j+1} is obtained from ℓ_j by an exchange of the form $ab \mapsto ba$, all these coefficients remain unchanged except for

$$\binom{\ell_{j+1}}{ab} = \binom{\ell_j}{ab} - 1. \tag{2.4}$$

Corollary 2.3.9. *When applying the algorithm producing the word u from the word $\ell = 1^{|u|_1}2^{|u|_2} \cdots m^{|u|_m}$, the total number of exchanges $ab \mapsto ba$, with $a < b$, is given by*

$$\binom{\ell}{ab} - \binom{u}{ab} = \binom{u}{ba}.$$

Consequently two words are 2-binomially equivalent if and only if they are Abelian equivalent and the total number of exchanges of each type $ab \mapsto ba$, $a < b$, when applying the algorithm to these two words, is the same. An equivalence class $[\cdot]_{\sim_2}$ is thus completely determined by a word $\ell = 1^{n_1}2^{n_2} \cdots m^{n_m}$ and the number of different exchanges. We obtain an algorithm generating all words of $[u]_{\sim_2}$.

Definition 2.3.10. Given a word u, define a directed graph $\mathcal{G}_u$ whose vertices are the words belonging to the Abelian equivalence class of u. There exists an edge from v to v' if and only if v' is obtained by an exchange of the type $ab \mapsto ba$, $a < b$, from v. The edge is labelled with the applied exchange. In particular, for any vertex v in $\mathcal{G}_u$, there is at least a path from $\ell_0 = 1^{|u|_1}2^{|u|_2} \cdots m^{|u|_m}$ to v.

Definition 2.3.11. Let v be a node of $\mathcal{G}_u$. If there are several paths from ℓ_0 to v in the graph, then exactly one of them follows the sequence $\mathcal{L}_v$. Thanks to Remark 2.3.6, restricting $\mathcal{G}_u$ to these paths gives a subgraph having the same set of vertices which is a tree denoted by $\mathcal{T}_u$ and whose root is ℓ_0. In particular, if v' appears in the sequence $\mathcal{L}_v$, then $\mathcal{L}_{v'}$ is a prefix of $\mathcal{L}_v$.

Let us recap in the following statement what we have done so far.

Proposition 2.3.12. *Let u be a finite word. The $\sim_2$-equivalence class of u is composed of all the nodes of $\mathcal{T}_u$ that are at level $\sum_{1\leq a<b\leq m}\binom{u}{ba}$ and such that the path from the root ℓ_0 to such a node is composed of $\binom{u}{ba}$ edges labelled by $ab \mapsto ba$, for all letters $a < b$.*

Recall that our aim is to conveniently find all the words belonging to the $\sim_2$-class of u. Instead of building the full tree $\mathcal{T}_u$, starting from the root we will only build a relevant subtree. Moreover, we conveniently add two pieces of information to help us in the construction. First, if there is an edge from $vabw$ to $vbaw$ labelled by $ab \mapsto ba$, then we underline the letter b in the destination node $v\underline{b}aw$.

Secondly, a sequence of nodes $u_0 = \ell_0, u_1, \ldots, u_n$ defining a path in $\mathcal{T}_u$ corresponds to the sequence $\mathcal{L}_{u_n}$. Thus, we highlight as in Remark 2.3.6 the prefix that will not be modified anymore if we extend the path further. This prefix is separated from the remaining part of the word by a vertical line. Two cases can occur, whether or not we continue to swap the same letter:

1. either $u_i = v|wa\underline{b}w'$ and $u_{i+1} = v|w\underline{b}aw'$, or;
2. $u_i = v|x\underline{c}yabw'$ and $u_{i+1} = vxc|y\underline{b}aw'$.

Starting from the root $|1^{|u_1|}2^{|u_2|}\cdots m^{|u_m|}$ with no underlined letter, we build the tree level by level, by trying all the possible swaps that may occur on the right hand side of the vertical line. Moreover, if a path from the root to a node has a number of edges labelled by $ab \mapsto ba$ greater than $\binom{u}{ba}$, it is useless to add the children of this node, since they will not lead to any element of $[u]_{\sim_2}$. They are therefore parts of $\mathcal{T}_u$ that will not be explored.

Example 2.3.13. Let us continue Example 2.3.7 with $u = 1223312$ on the alphabet $\{1,2,3\}$. Its $\sim_2$-equivalence class is $\{1223312, 2311223\}$. It can be read from the tree in Figure 2.3, which is a subtree of $\mathcal{T}_u$. The edges labelled by $12 \mapsto 21$ (resp., $13 \mapsto 31$, $23 \mapsto 32$) are represented in black (resp., dotted red, dashed gray).

Let us illustrate the differences between $\mathcal{G}_u$, $\mathcal{T}_u$ and the subtree in Figure 2.3. For instance, the edges from $12|12\underline{3}23$ to $2|112\underline{3}23$ and from $11223|\underline{3}2$ to $12123|\underline{3}2$ are in the graph $\mathcal{G}_u$ but have been suppressed in the tree $\mathcal{T}_u$. These nodes do not appear as consecutive words in any sequence $\mathcal{L}_{u'}$ for $u' \in [u]_{\sim_1}$.

Moreover, a dashed gray edge from $|1123\underline{2}223$ to $1123|\underline{2}3\underline{2}$ is in $\mathcal{T}_u$ but has not been computed in our figure since the path leading to this node would have three dashed gray edges but we know that we may only have $\binom{u}{32} = 2$ dashed gray edges on any path from the root.

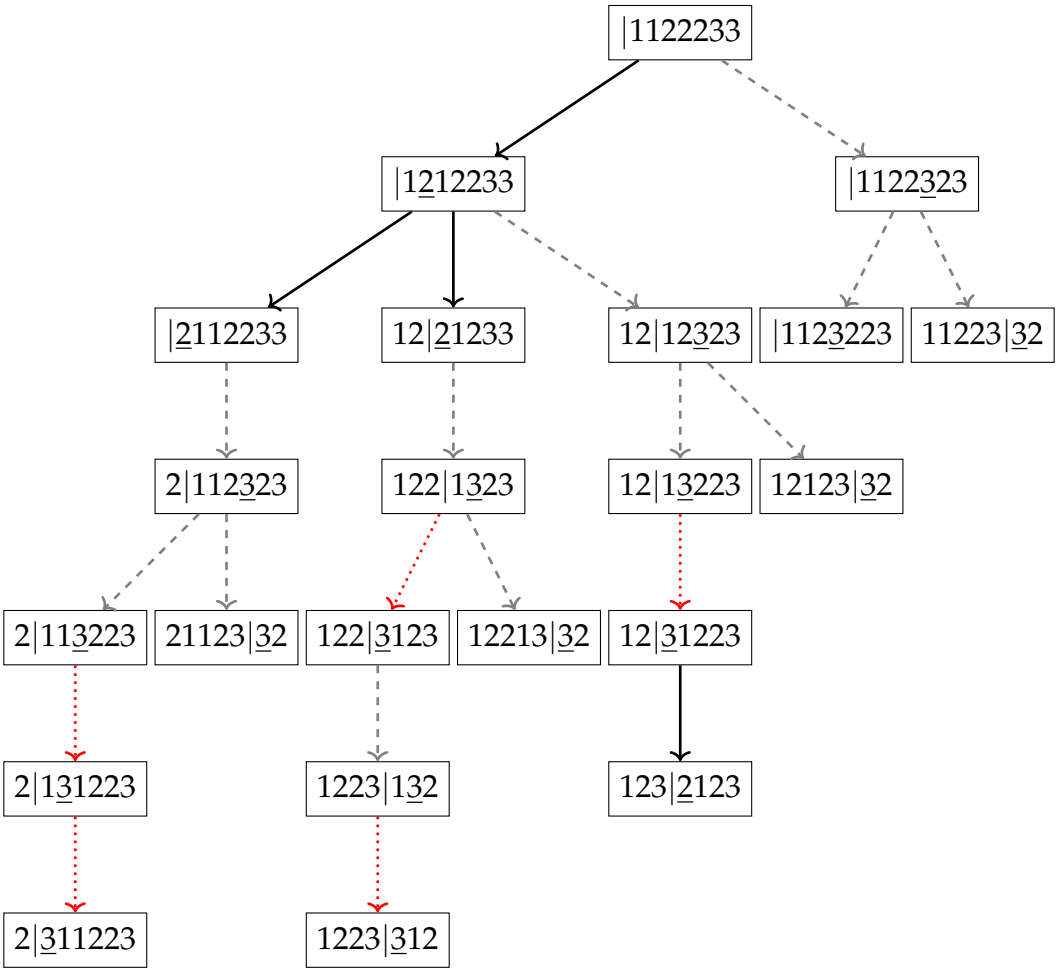


Figure 2.3: Generating the $\sim_2$-class of 1223312.

Note that a polynomial time algorithm checking whether or not two words are k-binomially equivalent has been obtained in [44] and is of independent interest.

2.3.3 Isomorphism with a nil-2 submonoid

Since we are dealing with the extended alphabet $\mathcal{A} \cup \mathcal{A}^{-1}$, let us first introduce a convenient variation of binomial coefficients of words taking into account inverse letters.

Definition 2.3.14. Let $t \geq 0$ be an integer. For all words u over the alphabet $\mathcal{A} \cup \mathcal{A}^{-1}$ and $v \in \mathcal{A}^t$, let us define

$$\begin{bmatrix} u \\ v \end{bmatrix} = \sum_{(e_1,\dots,e_t)\in\{-1,1\}^t} \left(\prod_{i=1}^{t} e_i\right) \binom{u}{v_1^{e_1}\cdots v_t^{e_t}},$$

where $\binom{u}{v_1^{e_1}\cdots v_t^{e_t}}$ is the usual binomial coefficient over the alphabet $\mathcal{A} \cup \mathcal{A}^{-1}$. Let $u \in \left(\mathcal{A} \cup \mathcal{A}^{-1}\right)^*$ and denote[4]

$$\underline{\tilde{\mathbf{\Phi}}}(u) = \left(\begin{bmatrix} u \\ 1 \end{bmatrix}, \dots, \begin{bmatrix} u \\ m \end{bmatrix}, \begin{bmatrix} u \\ 12 \end{bmatrix}, \dots, \begin{bmatrix} u \\ m(m-1) \end{bmatrix}\right)^{\mathsf{T}} \in \mathbb{Z}^{m^2}$$

where the last $m^2 - m$ components are obtained from all the words consisting of two different letters in $\mathcal{A}$, ordered by lexicographical order.

Notice that if u and v are words over $\mathcal{A}$, then

$$\begin{bmatrix} u \\ v \end{bmatrix} = \binom{u}{v}$$

and so, u and v are 2-binomially equivalent if and only if $\underline{\tilde{\mathbf{\Phi}}}(u) = \underline{\tilde{\mathbf{\Phi}}}(v)$.

Example 2.3.15. Let $\mathcal{A} = \{1, 2, 3\}$ and $u = 123^{-1}231^{-1}$. Applying the previous definition, for all $a \in \mathcal{A}$, we have

$$\begin{bmatrix} u \\ a \end{bmatrix} = \binom{u}{a} - \binom{u}{a^{-1}}.$$

Similarly, for all $a, b \in \mathcal{A}$, we have

$$\begin{bmatrix} u \\ ab \end{bmatrix} = \binom{u}{ab} - \binom{u}{a^{-1}b} - \binom{u}{ab^{-1}} + \binom{u}{a^{-1}b^{-1}}. \tag{2.5}$$

Therefore, computing classical binomial coefficients, we obtain

$$\underline{\tilde{\mathbf{\Phi}}}(u) = (0, 2, 0, 2, 0, -2, 1, 0, -1)^{\mathsf{T}}.$$

We are now ready to prove the main result of this section.

Theorem 2.3.16. *Let $\mathcal{A} = \{1, \dots, m\}$. The monoid $\mathcal{A}^*/\sim_2$ is isomorphic to the submonoid, generated by $\mathcal{A}$, of the nil-2 group $N_2(\mathcal{A})$.*

[4]This notation can look cumbersome but the reader will understand the use of $\tilde{}$ in Chapter 4 where the extended Parikh vector $\underline{\mathbf{\Phi}}$ is defined. Indeed, $\underline{\tilde{\mathbf{\Phi}}}$ is an extension to the group $(\mathcal{A} \cup \mathcal{A}^{-1})^*$ of $\underline{\mathbf{\Phi}}$ defined on the monoid $\mathcal{A}^*$.

Proof. Let us recall that π is the natural projection defined in (2.3). We will first show that for any two words u and u' over $\mathcal{A} \cup \mathcal{A}^{-1}$ such that $\pi(u) = \pi(u')$, the relation $\underline{\tilde{\mathbf{\Phi}}}(u) = \underline{\tilde{\mathbf{\Phi}}}(u')$ holds. Indeed, using (2.5) one can easily check that, for all $a, b \in \mathcal{A}$ and $s, t \in (\mathcal{A} \cup \mathcal{A}^{-1})^*$, we have

$$\begin{bmatrix} st \\ ab \end{bmatrix} = \begin{bmatrix} s \\ ab \end{bmatrix} + \begin{bmatrix} t \\ ab \end{bmatrix} + \begin{bmatrix} s \\ a \end{bmatrix} \begin{bmatrix} t \\ b \end{bmatrix}.$$

Now, one can show that, for all $v, w \in (\mathcal{A} \cup \mathcal{A}^{-1})^*$ and $x, y, z \in \mathcal{A} \cup \mathcal{A}^{-1}$,

$$\underline{\tilde{\mathbf{\Phi}}}(vw) = \underline{\tilde{\mathbf{\Phi}}}(vxx^{-1}w) \quad \text{and} \quad \underline{\tilde{\mathbf{\Phi}}}(v[x,y]zw) = \underline{\tilde{\mathbf{\Phi}}}(vz[x,y]w).$$

For instance, let $a, b \in \mathcal{A}$ with $a \neq b$,

$$\begin{aligned} \begin{bmatrix} vxx^{-1}w \\ ab \end{bmatrix} &= \begin{bmatrix} v \\ ab \end{bmatrix} + \begin{bmatrix} xx^{-1}w \\ ab \end{bmatrix} + \begin{bmatrix} v \\ a \end{bmatrix} \begin{bmatrix} xx^{-1}w \\ b \end{bmatrix} \\ &= \begin{bmatrix} v \\ ab \end{bmatrix} + \underbrace{\begin{bmatrix} xx^{-1} \\ ab \end{bmatrix}}_{=0} + \begin{bmatrix} w \\ ab \end{bmatrix} + \underbrace{\begin{bmatrix} xx^{-1} \\ a \end{bmatrix}}_{=0} \begin{bmatrix} w \\ b \end{bmatrix} + \begin{bmatrix} v \\ a \end{bmatrix} \cdot \left(\underbrace{\begin{bmatrix} xx^{-1} \\ b \end{bmatrix}}_{=0} + \begin{bmatrix} w \\ b \end{bmatrix} \right) \\ &= \begin{bmatrix} vw \\ ab \end{bmatrix}. \end{aligned}$$

This implies that a map $\underline{\tilde{\mathbf{\Phi}}}_N$ can be defined on the free nil-2 group (otherwise stated, the diagram depicted in Figure 2.4 is commutative) by

$$\forall r \in N_2(\mathcal{A}), \quad \underline{\tilde{\mathbf{\Phi}}}_N(r) = \underline{\tilde{\mathbf{\Phi}}}(u) \text{ for any } u \text{ such that } \pi(u) = r.$$

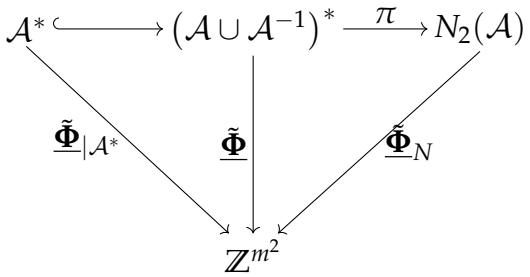


Figure 2.4: A commutative diagram (proof of Theorem 2.3.16).

In particular, if u and u' are words over $\mathcal{A}$ such that $\pi(u) = \pi(u')$, then we may conclude that $\underline{\tilde{\mathbf{\Phi}}}(u) = \underline{\tilde{\mathbf{\Phi}}}(u')$ meaning that they are 2-binomially equivalent.

Otherwise stated, for every $r \in N_2(\mathcal{A})$, $\pi^{-1}(r) \cap \mathcal{A}^*$ is a subset of an equivalence class for $\sim_2$.

To conclude the proof, we have to show that all the elements of an equivalence class for $\sim_2$ are mapped by π on the same element of $N_2(\mathcal{A})$. Let $v, w \in \mathcal{A}^*$ be such that $v \sim_2 w$. Using the algorithm described in Subsection 2.3.2, there exists a path in the associated tree from the root $1^{|v|_1}2^{|v|_2}\cdots m^{|v|_m}$ to v and another one to w. By the definition of the commutator, if v is written $pbas$ with $a < b$, then $v = pab[b,a]s$. Moreover, $\pi(v) = \pi(pabs[b,a])$ since the commutators are central in $N_2(\mathcal{A})$.

Therefore, following backwards the path from v to the root of the tree and recalling that each edge corresponds to an exchange of 2 letters, we obtain

$$\pi(v) = \pi\left(1^{|v|_1}2^{|v|_2}\cdots m^{|v|_m}[2,1]^{\binom{v}{21}}\cdots[m,1]^{\binom{v}{m1}}\cdots[m,m-1]^{\binom{v}{m(m-1)}}\right)$$

and, similarly, following backwards the path from w to the root,

$$\pi(w) = \pi\left(1^{|w|_1}2^{|w|_2}\cdots m^{|w|_m}[2,1]^{\binom{w}{21}}\cdots[m,1]^{\binom{w}{m1}}\cdots[m,m-1]^{\binom{w}{m(m-1)}}\right).$$

But since $v \sim_2 w$, we get $\pi(v) = \pi(w)$. □

2.4 Growth order

Let f, g be real applications defined on $\mathbb{N}_0$. We say that $f \in \mathcal{O}(g)$ if there exist $N \in \mathbb{N}$ and $C > 0$ such that $|f(n)| \leq C|g(n)|$ for any $n \geq N$. Let $\mathcal{A}$ be an arbitrary alphabet; this section is devoted to the study of the function

$$n \mapsto \#(\mathcal{A}^n / \sim_k),$$

also called *growth of* $(\mathcal{A}^* / \sim_k)$. This function counts the number of $\sim_k$-equivalence classes amongst all words of length n. We first show that the growth is bounded by a polynomial in n. This generalizes a result from [109] for a binary alphabet where it is shown that $\#(\{1,2\}^n / \sim_k) \in \mathcal{O}(n^{2((m-1)2^m+1)})$ for $k \geq 2$.

Note that a similar result to Proposition 2.4.1 was obtained in [65]. That result states that for $\Sigma = \{1, \ldots, m\}$, $m \geq 2$ and $k \geq 2$, we have

$$\#(\mathcal{A}^n / \sim_k) \in \mathcal{O}\left(n^{\frac{m}{(m-1)^2}(1+m^k(km-k-1))}\right).$$

Finally, we obtain better estimates for $\sim_2$.

Proposition 2.4.1. *Let $\mathcal{A} = \{1, \ldots, m\}$ and $k \geq 1$. We have*

$$\#(\mathcal{A}^n / \sim_k) \in \mathcal{O}\left(n^{k^2m^k}\right).$$

Proof. For every $u, v \in \mathcal{A}^*$ such that $1 \leq |v| \leq k$ and $|u| = n$, we have

$$0 \leq \binom{u}{v} \leq \binom{|u|}{|v|} \leq n^{|v|} \leq n^k.$$

Therefore, for every v such that $1 \leq |v| \leq k$, we have

$$\#\left\{\binom{u}{v} \,:\, |u| = n\right\} \leq n^k + 1.$$

By definition, the $\sim_k$-equivalence class of u is uniquely determined by the values of $\binom{u}{v}$ for all $v \in \mathcal{A}^*$ such that $|v| \in [k]$. There are

$$\sum_{i=1}^{k} m^i \leq km^k$$

such coefficients and thus,

$$\#(\mathcal{A}^n / \sim_k) \leq \left(n^k + 1\right)^{km^k}.$$

□

We have obtained an upper bound which is far from being optimal but it ensures that the growth is polynomial. However, for $k = 2$, it is possible to obtain the polynomial degree of the growth.

Proposition 2.4.2. *Let $\mathcal{A} = \{1, \ldots, m\}$ be an alphabet of size $m \geq 2$. We have*

$$\#(\mathcal{A}^n / \sim_2) \in \Theta\left(n^{m^2-1}\right).$$

Proof. We are first going to show that $\#(\mathcal{A}^n / \sim_2) \in \mathcal{O}\left(n^{m^2-1}\right)$. Let f be the function such that for any $\underline{\mathbf{x}} \in \mathbb{N}^m$,

$$f(\underline{\mathbf{x}}) = \#(\{u \in \mathcal{A}^* : \underline{\mathbf{\Psi}}(u) = \underline{\mathbf{x}}\} / \sim_2).$$

Namely, $f(x)$ counts the number of 2-binomial equivalence classes whose Parikh vector is $\underline{\mathbf{x}}$. Let $||\cdot||_1 : \mathbb{R}^d \to \mathbb{R}$ be the ℓ_1-norm (i.e., for all vectors $\underline{\mathbf{v}}$, $||\underline{\mathbf{v}}||_1 = \sum_{i=1}^{d} |v_i|$). Clearly for all n,

$$\#(\mathcal{A}^n / \sim_2) = \sum_{\underline{\mathbf{x}} \in \mathbb{N}^m, ||\underline{\mathbf{x}}||_1 = n} f(\underline{\mathbf{x}}). \tag{2.6}$$

For any $a, b \in \mathcal{A}$, $a < b$, and $u \in \mathcal{A}^*$, $\binom{u}{ba} = |u|_a |u|_b - \binom{u}{ab}$ and $\binom{u}{aa} = \binom{|u|_a}{2}$. Therefore any word u has its $\sim_2$-equivalence class uniquely determined by the values

of $|u|_a$ for all $a \in \mathcal{A}$ and $\binom{u}{ab}$ for all $a < b \in \mathcal{A}$. Moreover, for all $u \in \mathcal{A}^*$ and $a < b \in \mathcal{A}$, $\binom{u}{ab} \leq |u|_a |u|_b$. We deduce that for all $\underline{\mathbf{x}} = (x_1, \ldots, x_m) \in \mathbb{N}^m$,

$$f(\underline{\mathbf{x}}) \leq \prod_{1 \leq a < b \leq m} x_a x_b \leq \prod_{1 \leq a < b \leq m} ||\underline{\mathbf{x}}||_1^2 \leq ||\underline{\mathbf{x}}||_1^{m(m-1)}.$$

From Equation (2.6), we get that

$$\#(\mathcal{A}^n / \sim_2) \leq \sum_{\underline{\mathbf{x}} \in \mathbb{N}^m, ||\underline{\mathbf{x}}||_1 = n} ||\underline{\mathbf{x}}||_1^{m(m-1)} = n^{m(m-1)} \ \#\left\{\underline{\mathbf{x}} \in \mathbb{N}^m \ : \ ||\underline{\mathbf{x}}||_1 = n\right\}.$$

Every vector of the set $\{\underline{\mathbf{x}} \in \mathbb{N}^m \ : \ ||\underline{\mathbf{x}}||_1 = n\}$ has its components in $[n]_0$. Moreover, if the first $(m-1)$ components are fixed, the last one is uniquely determined since their sum must equal n. Therefore, there are at most $(n+1)^{m-1}$ such vectors and we get

$$\#(\mathcal{A}^n / \sim_2) \leq n^{m(m-1)} (n+1)^{m-1} \leq (n+1)^{m^2-1}.$$

We conclude that $\#(\mathcal{A}^n / \sim_2) \in \mathcal{O}\left(n^{m^2-1}\right)$. It remains to get a convenient lower bound. We are going to give, for each $\underline{\mathbf{x}} \in \mathbb{N}^m$ such that $||\underline{\mathbf{x}}||_1 = n$, a language $L(\underline{\mathbf{x}})$ of words taking, for every $a, b \in \mathcal{A}^2$ such that $a < b$, a quadratic number of values for the binomial coefficient $\binom{u}{ab}$. Then, using Equation (2.6), we will obtain a lower bound.

For any $a, b \in \mathcal{A}$, $a \neq b$, and $i, j \in \mathbb{N}$, let

$$L_{a,b,i,j} = \{u \in \{a,b\}^* : |u|_a = i, |u|_b = j\}.$$

Considering all possible letter exchanges as in (2.4) from $a^i b^j$ to $b^j a^i$, the binomial coefficient $\binom{u}{ab}$ decreases by 1 at every step from ij to 0, we thus have

$$\left\{ \binom{u}{ab} : u \in L_{a,b,i,j} \right\} = [ij]_0 \tag{2.7}$$

which is a set of cardinality $ij + 1$. For any $\underline{\mathbf{x}} \in \mathbb{N}^m$, let us consider the following language

$$L(\underline{\mathbf{x}}) = \left(\prod_{a=1}^{m} \prod_{b=a+1}^{m} L_{a,b,\left\lfloor \frac{x_a}{m-1} \right\rfloor, \left\lfloor \frac{x_b}{m-1} \right\rfloor} \right) \prod_{a=m}^{1} a^{x_a \% (m-1)},$$

where the products must be understood as language concatenations, the indices of the last product are taken in decreasing order, and $x\%y$ is the remainder of the Euclidean division of x by y.

For instance for $m = 3$,

$$L(\underline{\mathbf{x}}) = \left\{ u_{1,2} u_{1,3} u_{2,3} r_3 r_2 r_1 : \forall a, b \in \mathcal{A}, u_{a,b} \in L_{a,b,\left\lfloor \frac{x_a}{2} \right\rfloor, \left\lfloor \frac{x_b}{2} \right\rfloor}, r_a = a^{x_a \% 2} \right\}.$$

Roughly speaking, for every $a < b$ and $u \in L(\underline{\mathbf{x}})$, we will show with Equation (2.8) that $\binom{u}{ab}$ mostly depends on $u_{a,b}$ and with Equation (2.9) that this binomial

coefficient takes a quadratic number of values (when choosing $u_{a,b}$ accordingly). Furthermore, the role of r_a words is limited to padding. Indeed, observe that for all $u \in L(\underline{\mathbf{x}})$, $\underline{\mathbf{\Psi}}(u) = \underline{\mathbf{x}}$.

Let $\underline{\mathbf{x}} \in \mathbb{N}^m$ and $u \in L(\underline{\mathbf{x}})$. Then, by definition, there exist words $u_{1,2}, u_{1,3}, \ldots, u_{m-1,m}$ with, for all $a < b$, $u_{a,b} \in L_{a,b,\left\lfloor \frac{x_a}{m-1} \right\rfloor, \left\lfloor \frac{x_b}{m-1} \right\rfloor}$, such that

$$u = \left(\prod_{a=1}^{m} \prod_{b=a+1}^{m} u_{a,b} \right) \prod_{a=m}^{1} a^{x_a \% (m-1)}.$$

Let i and j be two integers such that $1 \leq i < j \leq m$ and let us compute the binomial coefficient associated with ij. A subword ij either occurs in a single factor of the above product (the first two terms below), or i and j appear in two different factors:

$$\begin{aligned}
\binom{u}{ij} = &\sum_{a=1}^{m} \sum_{b=a+1}^{m} \binom{u_{a,b}}{ij} + \sum_{a=m}^{1} \binom{a^{x_a \% (m-1)}}{ij} \\
&+ \sum_{a<b \in \mathcal{A}} |u_{a,b}|_i \left(\sum_{\substack{a'<b' \in \mathcal{A} \\ (a',b')>(a,b)}} |u_{a',b'}|_j + \sum_{b \in \mathcal{A}} \left| b^{x_b \% (m-1)} \right|_j \right) \\
&+ \sum_{a=1}^{m} \sum_{b=1}^{a-1} \left| a^{x_a \% (m-1)} \right|_i \left| b^{x_b \% (m-1)} \right|_j .
\end{aligned}$$

Observe that by definition of $L(\underline{\mathbf{x}})$, the second and last terms vanish. Hence,

$$\binom{u}{ij} = \binom{u_{i,j}}{ij} + \underbrace{\sum_{\substack{a<b \in \mathcal{A} \\ a=i \text{ or } b=i}} \left\lfloor \frac{x_i}{m-1} \right\rfloor \left(\left(\sum_{\substack{a'<b' \in \mathcal{A} \\ (a',b')>(a,b) \\ a'=j \text{ or } b'=j}} \left\lfloor \frac{x_j}{m-1} \right\rfloor \right) + x_j \% (m-1) \right)}_{:= h_{i,j}(\underline{\mathbf{x}})}. \quad (2.8)$$

The second term of the latter expression is uniquely a function of $\underline{\mathbf{x}}$ (there is no dependency on u) while, from (2.7),

$$\left\{ \binom{u_{i,j}}{ij} : u_{i,j} \in L_{i,j,\left\lfloor \frac{x_i}{m-1} \right\rfloor, \left\lfloor \frac{x_j}{m-1} \right\rfloor} \right\} = \left\{ 0, 1, \ldots, \left\lfloor \frac{x_i}{m-1} \right\rfloor \left\lfloor \frac{x_j}{m-1} \right\rfloor + 1 \right\}.$$

Thus for a fixed $\underline{\mathbf{x}}$, considering all $u \in L(\underline{\mathbf{x}})$, $\binom{u}{ij}$ can take

$$\left\lfloor \frac{x_i}{m-1} \right\rfloor \left\lfloor \frac{x_j}{m_1} \right\rfloor + 1 \quad (2.9)$$

different values. Moreover, for all $\left(a_{1,2}, a_{1,3}, \ldots, a_{(m-1),m}\right)$ such that

$$a_{i,j} \in \left\{ h_{i,j}(\underline{\mathbf{x}}), \ldots, h_{i,j}(\underline{\mathbf{x}}) + \left\lfloor \frac{x_i}{m-1} \right\rfloor \left\lfloor \frac{x_j}{m-1} \right\rfloor \right\} \qquad \forall i < j,$$

there exists $u \in L(\underline{\mathbf{x}})$ such that $\binom{u}{ij} = a_{i,j}$ for all $i < j$. We deduce that, for all $\underline{\mathbf{x}}$,

$$f(\underline{\mathbf{x}}) \geq \prod_{a<b\in\mathcal{A}} \left(\left\lfloor \frac{x_a}{m-1} \right\rfloor \left\lfloor \frac{x_b}{m-1} \right\rfloor + 1 \right).$$

By Equation (2.6), we finally get the lower bound:

$$\begin{aligned}
\#(\mathcal{A}^n/\sim_2) &\geq \sum_{\substack{\underline{\mathbf{x}}\in\mathbb{N}^m \\ ||\underline{\mathbf{x}}||_1=n}} \prod_{a<b\in\mathcal{A}} \left(\left\lfloor \frac{x_a}{m-1} \right\rfloor \left\lfloor \frac{x_b}{m-1} \right\rfloor + 1 \right) \\
&\geq \sum_{\substack{\underline{\mathbf{x}}\in\mathbb{N}^m \\ ||\underline{\mathbf{x}}||_1=n \\ \forall i, x_i \geq \frac{n}{2m}+m}} \prod_{a<b\in\mathcal{A}} \left\lfloor \frac{x_a}{m-1} \right\rfloor \left\lfloor \frac{x_b}{m-1} \right\rfloor \\
&\geq \sum_{\substack{\underline{\mathbf{x}}\in\mathbb{N}^m \\ ||\underline{\mathbf{x}}||_1=n \\ \forall i, x_i \geq \frac{n}{2m}+m}} \prod_{a<b\in\mathcal{A}} \left(\frac{n}{2m(m-1)} \right)^2 \\
&\geq \left(\frac{n}{2m(m-1)} \right)^{m(m-1)} \# \left\{ \underline{\mathbf{x}} \in \mathbb{N}^m : ||\underline{\mathbf{x}}||_1 = n \ \wedge \ \forall i, x_i \geq \frac{n}{2m} + m \right\}.
\end{aligned}$$

The latter set contains the set

$$\left\{ \underline{\mathbf{x}} \in \mathbb{N}^m : ||\underline{\mathbf{x}}||_1 = n \quad \wedge \quad \forall i, x_i \geq \frac{n}{2m} + m \quad \wedge \quad \forall i < m, x_i \leq \frac{n}{m} \right\}.$$

For n large enough (i.e., $\frac{n}{2m} + m \leq \frac{n}{m}$), the cardinal of this set is

$$\left(\frac{n}{2m} - m + 1 \right)^{m-1} \in \Theta(n^{m-1}).$$

Moreover,

$$\left(\frac{n}{2m(m-1)} \right)^{m(m-1)} \in \Theta\left(n^{m(m-1)} \right)$$

and we conclude that $\#(\mathcal{A}^n/\sim_2) \in \Theta\left(n^{m^2-1}\right)$. □

Remark 2.4.3. Note that even though the growth of $\#(\{1,2,3\}^n/\sim_2)$ is polynomial, this quantity is not a polynomial. It is easy to verify by interpolating the 9 first values. A similar result can be obtained for $\#(\{1,2\}^n/\sim_3)$ whose first values can be found as entry `A258585` in Sloane's encyclopedia.

2.5 k-binomial equivalence over an alphabet of more than 2 letters

In this section, we show that for any alphabet $\mathcal{A}$ of size at least 3 and for any $k \geq 2$, the languages $\mathsf{LL}(\sim_k, \mathcal{A})$ and $\mathsf{Sing}(\sim_k, \mathcal{A})$ are not context-free.

We easily deduce[5] from the previous section that both languages $\mathsf{LL}(\sim_k, \mathcal{A})$ and $\mathsf{Sing}(\sim_k, \mathcal{A})$ have polynomial growth; it is thus enough by Proposition 2.1.15 to show that they are not bounded to infer that they are not context-free. Observe that, in our forthcoming reasonings, we will define particular words $\rho_{p,n}$ over a ternary alphabet (they can trivially be seen as words over a larger alphabet).

Fix a sequence $(s_n)_{n\geq 1}$ of positive integers such that, for all $n \in \mathbb{N}$,

$$\sqrt{\frac{s_n}{2}} \in \mathbb{N}, \tag{D1}$$

$$s_n > \left(\sqrt{\frac{s_n}{2}} + \sum_{i=1}^{n-1} s_i\right)^2, \tag{D2}$$

$$\sqrt{\frac{s_n}{2}} > \left(\sum_{i=1}^{n-1} s_i\right)\left(\sum_{i=1}^{n-3} s_i\right). \tag{D3}$$

These conditions can look strange but we will see in the following results that they are necessary to ensure that $(s_n)_{n\geq 1}$ grows quickly enough. The following result proves that such a sequence exists, by giving it explicitly.

Proposition 2.5.1. *The sequence $(s_n)_{n\geq 1}$ given by*

$$s_n = 2 \cdot 8^{(8^n)},\ \forall n \in \mathbb{N}$$

satisfies conditions (D1), (D2) *and* (D3).

Proof. Condition (D1) is trivial. Let us prove (D2) by recurrence on n. We obviously have $s_1 > \frac{s_1}{2}$. Then assume that (D2) holds for $n \in \mathbb{N}$ and show it for $n+1$. Note that

$$s_{n+1} = 2 \cdot 8^{(8^n)\cdot 8} = \frac{2^8}{2^7}\left(8^{(8^n)}\right)^8 = \frac{s_n^8}{2^7} > s_n^7.$$

Condition (D2) is equivalent to

$$\sqrt{s_n}\left(1 - \frac{\sqrt{2}}{2}\right) > \sum_{i=1}^{n-1} s_i.$$

[5] $\mathsf{Sing}(\sim_k, \mathcal{A}) \subseteq \mathsf{LL}(\sim_k, \mathcal{A})$ and $\mathsf{LL}(\sim_k, \mathcal{A})$ is in one-to-one correspondence with $\mathcal{A}^*/\sim_k$.

We get

$$\begin{aligned}\sqrt{s_{n+1}}\left(1-\frac{\sqrt{2}}{2}\right) &> s_n^3\sqrt{s_n}\left(1-\frac{\sqrt{2}}{2}\right)\\ &= \sqrt{s_n}\left(1-\frac{\sqrt{2}}{2}\right)+(s_n^3-1)\sqrt{s_n}\left(1-\frac{\sqrt{2}}{2}\right)\\ &> \sum_{i=1}^{n-1} s_i + s_n,\end{aligned}$$

where the latter inequality comes from the induction hypothesis, and from the facts that $s_n^3 - 1 > s_n$ and $\sqrt{s_n}\left(1-\frac{\sqrt{2}}{2}\right) > 1$.

It remains to check that Condition (D3) holds. We will even show something stronger; we claim that

$$\sqrt{\frac{s_n}{2}} > \left(\sum_{i=1}^{n-1} s_i\right)^2,$$

for all $n \in \mathbb{N}$. The result is obviously true for $n = 1$; assuming that it holds for $n \in \mathbb{N}$ and proceeding by induction we get

$$\begin{aligned}\sqrt{\frac{s_{n+1}}{2}} &> \frac{1}{\sqrt{2}}\, s_n^3\sqrt{s_n}\\ &> s_n^3\left(\sum_{i=1}^{n-1} s_i\right)^2.\end{aligned}$$

Since

$$\begin{aligned}\frac{s_n^3}{3}\left(\sum_{i=1}^{n-1} s_i\right)^2 &> \left(\sum_{i=1}^{n-1} s_i\right)^2,\\ \frac{s_n^3}{3}\left(\sum_{i=1}^{n-1} s_i\right)^2 &> s_n^2,\\ \text{and}\quad \frac{s_n^3}{3}\left(\sum_{i=1}^{n-1} s_i\right)^2 &> 2s_n\sum_{i=1}^{n-1} s_i,\end{aligned}$$

we finally get

$$\sqrt{\frac{s_{n+1}}{2}} > \left(\sum_{i=1}^{n} s_i\right)^2,$$

as desired. □

Definition 2.5.2. For any integers n and p, let us define the word

$$\rho_{p,n} = 1^p 2^{s_{n-1}} 3^{s_{n-2}} 1^{s_{n-3}} \cdots a^{s_1}$$

over $\{1,2,3\}$, where $a \equiv n \pmod 3$.

In Subsection 2.5.1, we prove that the $\sim_2$-class of any $\rho_{p,n}$ is a singleton. Then, it is proven in Subsection 2.5.2 that $\{\rho_{p,n} \mid p,n \in \mathbb{N}\}$ is not a bounded language. Putting together these results, we get the following.

Theorem 2.5.3. *For any alphabet $\mathcal{A} = \{1,\dots,m\}$ of size at least 3 and for any $k \geq 2$, the languages $\mathsf{LL}(\sim_k, \mathcal{A})$ and $\mathsf{Sing}(\sim_k, \mathcal{A})$ are not context-free.*

The proof can be found at the end of Subsection 2.5.2.

2.5.1 A family of singletons

Proposition 2.5.4. *For any two positive integers n and p and any word u, at least one of the following is false:*

- $u \neq \rho_{p,n}$; (2.10)
- $\underline{\mathbf{\Psi}}(u) = \underline{\mathbf{\Psi}}(\rho_{p,n})$; (2.11)
- $\binom{u}{12} \geq \binom{\rho_{p,n}}{12}$; (2.12)
- $\binom{u}{23} \geq \binom{\rho_{p,n}}{23}$; (2.13)
- $\binom{u}{31} \geq \binom{\rho_{p,n}}{31}$. (2.14)

Proof. Let us show the proposition by induction on n. The result clearly holds for $n \leq 3$. Let $n \geq 4$ be an integer such that the result holds for any $i < n$. Now let us proceed by contradiction to show that the result also holds for n.

For the sake of contradiction, let p and u be such that Assertions (2.10) to (2.14) are verified. Let $u = vw$ where v is the prefix of length $p + s_{n-1} - \sqrt{\frac{s_{n-1}}{2}}$ of u. Similarly, let $\rho_{p,n} = v'w'$ where $|v| = |v'|$. In the first part of the proof, we show that $\underline{\mathbf{\Psi}}(v) = \underline{\mathbf{\Psi}}(v')$ and more precisely $|v|_1 = p = |v'|_1$, $|v|_3 = 0 = |v'|_3$. We proceed into three steps.

• Proof of $|v|_1 \geq p$:

For the sake of contradiction, suppose $|v|_1 \leq p-1$. Then

$$\begin{aligned}\binom{u}{12} &= \binom{v}{12} + \binom{w}{12} + |v|_1|w|_2 \\ &\leq |v|_1|v|_2 + |w|_1|w|_2 + |v|_1|w|_2 \\ &\leq |v|_1|u|_2 + |w|_1|w|_2 \\ &\leq (p-1)|u|_2 + |w|^2.\end{aligned}$$

Replacing $|w|$ by its value, we get

$$\binom{u}{12} \leq p|u|_2 + \left(\sqrt{\frac{s_{n-1}}{2}} + \sum_{i=1}^{n-2} s_i\right)^2 - |u|_2. \tag{2.15}$$

By (2.11), $|u|_2 = |\rho_{p,n}|_2$ and condition (D2) implies

$$0 > \left(\sqrt{\frac{s_{n-1}}{2}} + \sum_{i=1}^{n-2} s_i\right)^2 - s_{n-1} \geq \left(\sqrt{\frac{s_{n-1}}{2}} + \sum_{i=1}^{n-2} s_i\right)^2 - |u|_2.$$

Together with (2.15), it gives $\binom{u}{12} < p|\rho_{p,n}|_2 \leq \binom{\rho_{p,n}}{12}$. This is a contradiction with hypothesis (2.12) and we conclude that $|v|_1 \geq p$.

• Proof of $|v|_3 = 0$:

For the sake of contradiction, suppose $|v|_3 \geq 1$.

$$\begin{aligned}\binom{u}{23} &= |u|_2|u|_3 - \binom{u}{32} \\ &= |u|_2|u|_3 - \binom{v}{32} - \binom{w}{32} - |v|_3|w|_2 \\ &\leq |u|_2|u|_3 - |v|_3|w|_2 \\ &\leq |u|_2|u|_3 - |w|_2. \end{aligned} \tag{2.16}$$

Observe that

$$\begin{aligned}|w|_2 &= |u|_2 - |v|_2 \\ &= |u|_2 - |v| + |v|_1 + |v|_3 \\ &> |u|_2 - |v| + |v|_1\end{aligned}$$

and

$$\begin{aligned}|u|_2 - |v| + |v|_1 &= |u|_2 - p - s_{n-1} + \sqrt{\frac{s_{n-1}}{2}} + |v|_1 \\ &= (|u|_2 - s_{n-1}) + (|v|_1 - p) + \sqrt{\frac{s_{n-1}}{2}} \\ &\geq \sqrt{\frac{s_{n-1}}{2}}\,.\end{aligned}$$

Moreover, by (2.11), $|u|_2|u|_3 = |\rho_{p,n}|_2|\rho_{p,n}|_3$. We can use these two remarks in Inequality (2.16).

$$\begin{aligned}\binom{u}{23} &< |\rho_{p,n}|_2|\rho_{p,n}|_3 - \sqrt{\frac{s_{n-1}}{2}} \\ &< |\rho_{p,n}|_2|\rho_{p,n}|_3 - \left(\sum_{i=1}^{n-2} s_i\right)\left(\sum_{i=1}^{n-4} s_i\right) \text{ (from (D3))} \\ &< |\rho_{p,n}|_2|\rho_{p,n}|_3 - \sum_{i=0}^{\lfloor\frac{n-3}{3}\rfloor}\left(s_{n-2-3i}\sum_{j=i}^{\lfloor\frac{n-5}{3}\rfloor} s_{n-4-3j}\right)\end{aligned}$$

The latter quantity is equal to $|\rho_{p,n}|_2|\rho_{p,n}|_3 - \binom{\rho_{p,n}}{32}$ and thus

$$\binom{u}{23} < \binom{\rho_{p,n}}{23}.$$

This contradicts hypothesis (2.13) and we conclude that $|v|_3 = 0$.

• Proof of $|v|_1 \leq p$:

For the sake of contradiction, suppose $|v|_1 \geq p+1$. Then

$$\binom{u}{13} \geq |v|_1|w|_3 \geq p|w|_3 + |w|_3.$$

Since $|v|_3 = 0$, $|w|_3 = |u|_3 = |\rho_{p,n}|_3 \geq s_{n-2} > \sqrt{\frac{s_{n-2}}{2}}$. From condition (D3), taking into account the structure of $\rho_{p,n}$, we deduce

$$\binom{u}{13} > p|\rho_{p,n}|_3 + \left(\sum_{i=1}^{n-3} s_i\right)\left(\sum_{i=1}^{n-5} s_i\right) \geq \binom{\rho_{p,n}}{13}.$$

This yields a contradiction with hypothesis (2.14). We conclude that $|v|_1 = p$. We thus have

$$\underline{\mathbf{\Psi}}(v') = \underline{\mathbf{\Psi}}(v) \text{ and } \underline{\mathbf{\Psi}}(w') = \underline{\mathbf{\Psi}}(w). \tag{2.17}$$

We will now use Equation (2.17) to find the contradiction; we are going to show that v' and w' are shorter words for which the result doesn't hold. From hypothesis (2.12), we get

$$\begin{aligned}\binom{w}{12} &= \binom{u}{12} - \binom{v}{12} - |v|_1|w|_2 \\ &\geq \binom{\rho_{p,n}}{12} - \binom{v}{12} - |v|_1|w|_2.\end{aligned}$$

Since $\rho_{p,n} = v'w'$, this latter quantity is equal to

$$\binom{v'}{12} + \binom{w'}{12} + |v'|_1|w'|_2 - \binom{v}{12} - |v|_1|w|_2 = \binom{v'}{12} + \binom{w'}{12} - \binom{v}{12}$$

where the last equality is due to (2.17). We thus obtain that

$$\binom{w}{12} \geq \binom{v'}{12} + \binom{w'}{12} - \binom{v}{12}. \tag{2.18}$$

Since v' is of the form $1^\alpha 2^\beta$,

$$\binom{v'}{12} = \max\left\{\binom{x}{12} : \underline{\Psi}(x) = \underline{\Psi}(v')\right\} \geq \binom{v}{12}$$

and we get $\binom{w}{12} \geq \binom{w'}{12}$. With similar arguments and the fact that $|v|_3 = 0 = |v'|_3$, we obtain $\binom{w}{23} \geq \binom{w'}{23}$ and $\binom{w}{31} \geq \binom{w'}{31}$.

Observe that $w \neq w'$. Indeed, it is obvious if $v = v'$ (since $vw \neq v'w'$) and otherwise, $\binom{v'}{12} > \binom{v}{12}$ and, from (2.18), we get $\binom{w}{12} > \binom{w'}{12}$.

Let σ be the morphism such that $\sigma(1) = 3; \sigma(2) = 1; \sigma(3) = 2$. Then $\sigma(w') = \rho_{\sqrt{\frac{s_{n-1}}{2}},n-1}$ and since σ is a permutation of the alphabet, we get

- $\sigma(w) \neq \sigma(w')$;
- $\underline{\Psi}(\sigma(w)) = \underline{\Psi}(\sigma(w'))$;
- $\binom{\sigma(w)}{12} \geq \binom{\sigma(w')}{12}$;
- $\binom{\sigma(w)}{23} \geq \binom{\sigma(w')}{12}$;
- $\binom{\sigma(w)}{31} \geq \binom{\sigma(w')}{31}$.

This is a contradiction with our induction hypothesis. We deduce that there is no such pair of integers and this concludes the proof of the proposition. □

As an immediate corollary, we get the following result.

Corollary 2.5.5. *For any two positive integers n and p and word u such that $u \neq \rho_{p,n}$, we have $u \not\sim_2 \rho_{p,n}$.*

2.5.2 Unboundedness

It remains us to prove that the language $\{\rho_{p,n} : p, n \in \mathbb{N}\}$ is not bounded. We will make use of the following notation. If u is a non-empty word, its *letter-factorization* is $(c_1, q_1), \ldots, (c_r, q_r)$, where

- $u = c_1^{q_1} c_2^{q_2} \cdots c_r^{q_r}$;
- $r \geq 1$;
- $c_1, \ldots, c_r$ are letters such that for all i, $c_i \neq c_{i+1}$;

- $q_1, \dots, q_r$ are positive integers.

The *number of blocks* in the word u, denoted by $nb(u)$, is r. It corresponds to the length of the decomposition.

Example 2.5.6. Let $u = 112333122132$. We have $c_1 = 1$, $c_2 = 2$, $c_3 = 3$, $c_4 = 1$, $c_5 = 2$, $c_6 = 1$, $c_7 = 3$, $c_8 = 2$, and $q_1 = 2$, $q_2 = 1$, $q_3 = 3$, $q_4 = 1$, $q_5 = 2$, $q_6 = q_7 = q_8 = 1$. Moreover, $nb(u) = 8$.

The letter-factorization of a word of the form $\rho_{p,n}$ has particular properties that we record in the following remark.

Remark 2.5.7. For all $p, n \in \mathbb{N}$, if $(c_1, q_1), \dots, (c_r, q_r)$ is the letter-factorization of $\rho_{p,n}$, we know that

- for all $i \geq 1$, $c_i \equiv i \pmod 3$, with $c_i \in \{1, 2, 3\}$;
- $q_1 = p$ and for all $i \in \{2, \dots, n\}$, $q_i = s_{n-i+1}$;
- $nb(\rho_{p,n}) = n$.

Lemma 2.5.8. *For all $\ell \in \mathbb{N}$ and words $u_1, \dots, u_\ell \in \mathcal{A}^*$, we have*

$$\{\rho_{p,n} : p, n \in \mathbb{N}\} \not\subset u_1^* \cdots u_\ell^*.$$

Proof. For the sake of contradiction, let us assume that there exist $\ell \in \mathbb{N}$ and words $u_1, \dots, u_\ell \in \mathcal{A}^*$ such that

$$\mathcal{R} := \{1^p 2^{s_{n-1}} 3^{s_{n-2}} \cdots : p \in \mathbb{N}, n \in \mathbb{N}\} \subseteq u_1^* \cdots u_\ell^*.$$

We will first show that, under this assumption, there exist $N \in \mathbb{N}$ and words $z_1, \dots, z_q$ such that, for all i, $nb(z_i) \leq 2$, and the subset

$$\mathcal{R}_N := \{\rho_{p,n} : p \in \mathbb{N}, n \geq N\}$$

of $\mathcal{R}$ is included in $z_1^* \dots z_q^*$.

Let us take the least $i \in [\ell]$ such that $nb(u_i) \geq 3$. If such an i does not exist, we can take $N = 0$, $\ell = q$ and $z_i = w_i$ for all i. Otherwise, the letter-factorization of u_i begins with $(a_1, \alpha_1), (a_2, \alpha_2), (a_3, \alpha_3)$. Assume that there exist p, n such that the factorization of $\rho_{p,n}$ in terms of $u_1, \dots, u_\ell$

$$\rho_{p,n} = u_1^{n_1} \cdots u_i^{n_i} \cdots u_\ell^{n_\ell}$$

contains an occurrence of u_i, i.e., $n_i > 0$, (if this is not the case, $\mathcal{R}$ is thus included in $u_1^* \cdots u_{i-1}^* u_{i+1}^* \cdots u_\ell^*$ and we can proceed to the next index such that $nb(u_i) \geq 3$).

Because of Remark 2.5.7, if $nb(u_j) = 2$ then $u_j u_j$ is never a factor of a word in $\mathcal{R}$ (this would mean that two letters out of three are alternating). In that case, we must have $n_j = 1$ in the above factorization. Also, if $nb(u_j) = 1$, then $nb(u_j^{n_j}) = 1$. By definition of i, $nb(u_j) \leq 2$ for all $j < i$. Therefore there exists $\gamma \leq 2i$ such that, if $(c_1, q_1), \ldots, (c_r, q_r)$ is the letter-factorization of $\rho_{p,n}$,

- $c_1^{q_1} \cdots c_{\gamma-1}^{q_{\gamma-1}} \in u_1^* \cdots u_{i-1}^* a_1^{\alpha_1}$;
- $c_\gamma = a_2$ and $q_\gamma = \alpha_2$;
- $c_{\gamma+1} = a_3$,

see Figure 2.5 for an illustration.

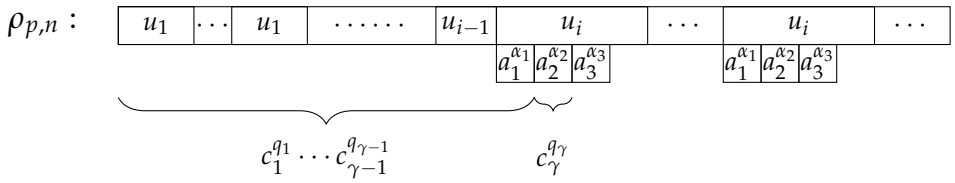


Figure 2.5: Decomposition of $\rho_{p,n}$ into blocks.

With Remark 2.5.7, we know that if $\gamma = 2$ then $q_{\gamma-1} = p$ and in all cases, $q_\gamma = s_{n-\gamma+1}$. Therefore, if we take N such that $s_{N-2i+1} > \alpha_2$, the set $\mathcal{R}_N$, which is included in $u_1^* \ldots u_\ell^*$ is also included in $u_1^* \ldots u_{i-1}^* u_{i+1}^* \ldots u_\ell^*$. We can proceed the same way to eliminate other factors u_j with $nb(u_j) \geq 3$ to finally obtain an integer N such that

$$\mathcal{R}_N = \{\rho_{p,n} : p \in \mathbb{N}, n \geq N\}$$

is included in a set of the form $z_1^* \ldots z_q^*$ where, for all i, $nb(z_i) \leq 2$.

It remains to show that this observation leads to a contradiction. Let $\rho_{p,n} \in \mathcal{R}_N$. It can be factorized as $z_1^{n_1} \cdots z_q^{n_q}$. We have already observed that if $nb(z_i) = 2$, then $n_i = 1$. Otherwise, $nb(z_i) = 1$ and thus $nb(z_i^{n_i}) = 1$. For this reason, we obtain that for all $n \geq N$,

$$nb(\rho_{p,n}) \leq 2q,$$

which is a contradiction because $nb(\rho_{p,n}) = n$ and this concludes the proof. □

Let us now come back to the theorem we were interested in, stated at the beginning of the section and that we recall here before giving the proof.

Theorem 2.5.3. *For any alphabet $\mathcal{A} = \{1, \ldots, m\}$ of size at least 3 and for any $k \geq 2$, the languages $\mathsf{LL}(\sim_k, \mathcal{A})$ and $\mathsf{Sing}(\sim_k, \mathcal{A})$ are not context-free.*

Proof. First note that

$$\{\rho_{p,n} \mid p, n \in \mathbb{N}\} \subseteq \{1,2,3\}^* \subseteq \mathcal{A}^*.$$

Taking into account Corollary 2.5.5, observe that

$$\{\rho_{p,n} \mid p, n \in \mathbb{N}\} \subseteq \mathsf{Sing}(\sim_k, \mathcal{A}) \subseteq \mathsf{LL}(\sim_k, \mathcal{A}).$$

From Proposition 2.4.1, the languages $\mathsf{LL}(\sim_k, \mathcal{A})$, and thus $\mathsf{Sing}(\sim_k, \mathcal{A})$, have a polynomial growth. From Lemma 2.5.8, the language $\{\rho_{p,n} \mid p, n \in \mathbb{N}\}$ is not bounded. Therefore, $\mathsf{Sing}(\sim_k, \mathcal{A})$ and $\mathsf{LL}(\sim_k, \mathcal{A})$ are not bounded and we conclude from Proposition 2.1.15. □

Remark 2.5.9. This result is in fact true not only for $\mathsf{LL}(\sim_k, \mathcal{A})$, but for all languages having exactly one representative of each $\sim_k$-class.

2.6 Further questions

As we have seen, there is a simple switch operation given by $\equiv$ that permits us to easily describe the 2-binomial equivalence class of a word over a binary alphabet. One could try to generalize this operation over larger alphabets or for $k \geq 3$, but the question has no clear answer yet.

However, over a larger alphabet, we gave algorithmic and algebraic descriptions of the 2-binomial classes. A natural question is to extend these results for $k \geq 3$.

We proved that $\mathsf{LL}(\sim_k, \mathcal{A})$ is not context-free if $k \geq 2$ and $\#\mathcal{A} \geq 3$. We know that $\mathsf{LL}(\sim_2, \{1,2\})$ is context-free. However, the question is still open about $\mathsf{LL}(\sim_k, \{1,2\})$ with $k \geq 3$. It seems that a method similar to the one carried in Section 2.5 could work, but it remains to find an unbounded set of singletons.

When $\mathsf{LL}(\sim_k, \mathcal{A})$ is not context-free, a measure of descriptional complexity is the so-called automaticity [114]: the automaticity of the language L is the function

$$n \in \mathbb{N} \mapsto A_L(n) \in \mathbb{N},$$

which counts the minimum number of states in any DFA M such that $\mathcal{L}(M) \cap \mathcal{A}^{\leq n}$ is equal to $L \cap \mathcal{A}^{\leq n}$.

Let $M = (Q, \mathcal{A}, \delta, q_0, F)$ be a DFA. For any $q \in Q$, we denote by L_q the set of words accepted by the automaton $(Q, \mathcal{A}, \delta, q, F)$, which is the automaton M where the initial state has been replaced by q:

$$L_q = \{u_1 \cdots u_n \in \mathcal{A}^* \ : \ \delta(\cdots(\delta(\delta(q, u_1), u_2)\cdots), u_n) \in F\}.$$

Nerode congruence [110] is the relation $\sim_N$ defined on Q and such that

$$q \sim_N q' \Leftrightarrow L_q = L_{q'}.$$

Let L be a language and C, t be integers. The idea for computing the automaticity is that we only know the words of L of length at most C. Consider the following approximation of Nerode congruence: for any two words u, v such that $|u|, |v| \leq t$,

$$u \approx_{L,C,t} v \;\Leftrightarrow\; \left(u^{-1}\left(L \cap \mathcal{A}^{\leq C}\right)\right) \cap \mathcal{A}^{\leq C-t} = \left(v^{-1}\left(L \cap \mathcal{A}^{\leq C}\right)\right) \cap \mathcal{A}^{\leq C-t}.$$

The quantity $\#\left(\mathcal{A}^{\leq t} / \approx_{L,C,t}\right)$ gives a lower approximation of the automaticity of L. For $L = \mathsf{LL}(\sim_3, \{1,2\})$, $C = 15$ and $t = 1, 2, \ldots, 9$, the first few values are

$$1, 3, 5, 9, 16, 27, 49, 88, 154.$$

For $L = \mathsf{LL}(\sim_2, \{1,2,3\})$, $C = 9$ and $t = 1, 2, \ldots, 6$, they are

$$1, 4, 8, 19, 42, 62.$$

Can the automaticity of such languages be characterized or estimated?

3 | The Thue–Morse word

We come back to the notion of k-binomial complexity. As mentioned in the first chapter, the question of determining the exact value of $b_{\mathbf{w}}^{(k)}$ is still open for a huge range of purely morphic words $\mathbf{w}$. We start by considering the k-binomial complexity of the Thue–Morse word, since $\mathbf{t}$ is one of the most classical binary infinite words. The goal of this chapter is to give the exact value of the mentioned function. From Proposition 1.4.5 we already know that $b_{\mathbf{t}}^{(k)}$ is bounded by a constant depending on k.

The Thue–Morse word $\mathbf{t} = 0110100110010110\cdots$ is ubiquitous in combinatorics on words [5, 101, 120]. It is an archetypal example of a 2-automatic sequence [6]: it is the fixed point of the morphism

$$\varphi : \begin{cases} 0 & \mapsto & 01; \\ 1 & \mapsto & 10. \end{cases}$$

All along this chapter we set $\mathcal{A} = \{0,1\}$ and $\overline{0} = 1$, $\overline{1} = 0$ (then, by induction, $\overline{u_1 \cdots u_n} = \overline{u_1} \cdots \overline{u_n}$). We also stress the fact that we consider in the following the first letter of $\mathbf{t}$ to be $\mathbf{t}_0$ and not $\mathbf{t}_1$ as it is stated for an arbitrary infinite word in Notation 1.1.3. This convention will simplify some of our calculations since we will work modulo 2^k.

The most prominent property of the Thue–Morse word is that it avoids overlaps, i.e., it does not contain any factor of the form $auaua$ where u is a word and a a symbol. Consequently it also avoids cubes of the form uuu and is aperiodic. The Thue–Morse word appears in many problems with a number-theoretic flavor, to cite a few: the Prouhet–Tarry–Escott problem for partitioning integers, transcendence of real numbers, duplication of the sine,... [3, 4, 34, 63].

To cite a more funny problem, it has been shown [27] that two duelers with similar shooting skills (called *Galois duelers*) will choose to take turns firing in accordance with the Thue–Morse sequence if they greedily demand their chances to fire as soon as the other's probability of winning exceeds their own. This property can be applied to football and following the first letters of $\mathbf{t}$ gives the best order to follow for penalty shoots-out at the end of a competition.

Let us finally mention a sentence from the review on MathSciNet of [13]: "The

nice combinatorial properties of its subword structure have inspired a number of papers" and Ochsenschläger [91] was the first to consider the subwords of its prefixes.

The factor complexity of the Thue–Morse word is in $\Theta(n)$ (see [6, Corollary 10.3.2]) and is recalled in the following proposition; for more details see [18, 33] or [15, Corollary 4.10.7].

Proposition 3.0.1. *The factor complexity $p_{\mathbf{t}}$ of the Thue–Morse word is given by $p_{\mathbf{t}}(0) = 1$, $p_{\mathbf{t}}(1) = 2$, $p_{\mathbf{t}}(2) = 4$ and for $n \geq 3$,*

$$p_{\mathbf{t}}(n) = \begin{cases} 4n - 2 \cdot 2^m - 4, & \text{if } 2 \cdot 2^m < n \leq 3 \cdot 2^m; \\ 2n + 4 \cdot 2^m - 2, & \text{if } 3 \cdot 2^m < n \leq 4 \cdot 2^m. \end{cases}$$

For many complexity measures, Sturmian words have the lowest complexity among aperiodic words, and variations of the Morse–Hedlund theorem notably exist for k-Abelian complexity [59]. Nevertheless, it is not always the case. The arithmetical complexity [10] of an infinite word $\mathbf{w}$ is the function $a_{\mathbf{w}}$ mapping to every $n \geq 0$ the number of words of length n that occur in arithmetical subsequences of $\mathbf{w}$:

$$a_{\mathbf{w}}(n) = \#\{\mathbf{w}_k \mathbf{w}_{k+d} \cdots \mathbf{w}_{k+(n-1)d} \mid k \geq 0, d \geq 1\}.$$

The arithmetical complexity of Sturmian words is in $\mathcal{O}(n^3)$ while Toeplitz words[1] have a linear arithmetical complexity, see [22, 45]. A similar phenomenon appears for k-binomial complexity: the Thue–Morse word has a bounded function while for Sturmian words, k-binomial complexity, $k \geq 2$, is equal to the factor complexity $p(n) = n + 1$ [109].

In Subsection 1.2.2, we gave formulas for computing the binomial coefficient $\binom{\sigma(u)}{v}$, where σ is an arbitrary morphism. Knowing precisely σ leads to, in particular cases, huge simplifications of the previous results. These simplifications applied in the case of the Thue–Morse morphism will be the main subject of Section 3.1. We are able to express the difference $\binom{\varphi(u)}{v} - \binom{\varphi(u')}{v}$ as a linear combination of the form

$$\sum_z m(z) \left[\binom{u}{z} - \binom{u'}{z} \right],$$

where the sum ranges over words z of length shorter than v, and we are able to precisely describe the integer coefficients $m(z)$.

Section 3.2 deals with $b_{\mathbf{t}}^{(2)}$ that has to be treated separately from the case $k \geq 3$. Then the general technique is presented in Section 3.3, where we develop a theory of

[1] Toeplitz words are iterative infinite words that can be defined as follows. Let ? be a letter not in $\mathcal{A}$ and let $u \in \mathcal{A}(\mathcal{A} \cup \{?\})^*$ be a pattern. Define the sequence of infinite words $(I_i(u))_{i \in \mathbb{N}_0}$: set $I_0(u) = u^\omega$ and for every $i \geq 1$, $I_i(u)$ is the word obtained from $I_{i-1}(u)$ by replacing the first occurrence of ? in u by the i^{th} letter of $I_{i-1}(u)$ (and by induction hypothesis, this letter is in $\mathcal{A}$). Then the Toeplitz word associated to the pattern u is $\lim_{i \to +\infty} I_i(u)$.

cutting bars, cutting sets and types of factors of $\mathbf{t}$. We finally establish in Section 3.4 the value of $b_{\mathbf{t}}^{(k)}(n)$ which is equal to

$$b_{\mathbf{t}}^{(k)}(n) = \begin{cases} p_{\mathbf{t}}(n), & \text{if } n \leq 2^k - 1; \\ 3 \cdot 2^k - 3, & \text{if } n \geq 2^k \text{ and } n \equiv 0 \pmod{2^k}; \\ 3 \cdot 2^k - 4, & \text{otherwise,} \end{cases}$$

as conjectured by J. Shallit during a research stay in Liège. Further directions are considered in the last section. Most of the results come from [67] by Julien Leroy, Marie Lejeune and Michel Rigo, published in Journal of Combinatorial Theory, Series A. The results were also presented in *Developments in Language Theory* 2019 and [66] is the proceeding version.

Contents

3.1 Computing the binomial coefficient of the image by φ of a word

As we have seen in Subsection 1.2.2 and more especially in Theorem 1.2.14, for any morphism σ the binomial coefficient $\binom{\sigma(u)}{v}$ can be expressed using binomial coefficients on u, as well as binomial coefficients applied on $\sigma(a)$, $a \in \mathcal{A}$. For this reason it is evident that the more we know about properties of σ, the more the expression of $\binom{\sigma(u)}{v}$ is easy to compute. In this section we improve the formula for the Thue–Morse morphism φ.

3.1.1 The formula

Let us start with an introductory example.

Example 3.1.1. We want to compute

$$\binom{\varphi(0110001)}{v} \quad \text{with } v = 01011.$$

Let u be equal to 0110001. The word $\varphi(u)$ belongs to $\{01, 10\}^*$. It can be factored into consecutive blocks $b_1 b_2 \cdots b_7$ of length 2. To count the number of occurrences of the subword v in the image by φ of a word, several cases need to be taken into account:

- the five symbols of v appear in pairwise distinct 2-blocks of $\varphi(u)$ (each 2-block contains both 0 and 1 exactly once), and there are
$$\binom{|u|}{|v|} = \binom{7}{5} = 21$$
such choices;

- the prefix 01 of v is one of the 2-blocks b_i of $\varphi(u)$ and the last three symbols of v appear in subsequent pairwise distinct 2-blocks b_j, $j > i$. Since $\varphi(0) = 01$, we have to count the number of occurrences of the subword $0z$, for all words z of length 3, in u. There are
$$\sum_{z \in \mathcal{A}^3} \binom{0110001}{0z} = \binom{6}{3} + 1 = 21$$
such choices;

- the first symbol of v appears in a 2-block, the factor 10 in v appears as a whole 2-block and the last two symbols 11 occur in subsequent pairwise distinct 2-blocks. There are
$$\sum_{x \in \mathcal{A}} \sum_{z \in \mathcal{A}^2} \binom{0110001}{x1z} = \binom{5}{2} + \binom{2}{1}\binom{4}{2} = 22$$
such choices;

- the first two symbols of v appear in distinct 2-blocks, the second occurrence of 01 in v appears as a whole 2-block and the last symbol 1 occurs in a subsequent 2-block. There are
$$\sum_{x \in \mathcal{A}^2} \sum_{z \in \mathcal{A}} \binom{0110001}{x0z} = \binom{3}{2}\binom{3}{1} + \binom{4}{2}\binom{2}{1} + \binom{5}{2} = 31$$
such choices;

- finally, v can appear as two 2-blocks $\varphi(0)$ followed by a single letter occurring in a subsequent 2-block. There are
$$\sum_{z \in \mathcal{A}} \binom{0110001}{00z} = \binom{3}{1} + \binom{2}{1}\binom{2}{1} + \binom{3}{1} = 10$$
such choices.

Hence $\binom{\varphi(u)}{v} = 105$.

The five cases discussed here above correspond to the following factorizations of v:

$$01011,\ \varphi(0)011,\ 0\varphi(1)11,\ 01\varphi(0)1,\ \varphi(0)\varphi(0)1.$$

The general scheme behind this computation is expressed by Theorem 3.1.10 given below. The reader can already feel that, in the Thue–Morse case, we need to take into account particular factorizations of v with respect to occurrences of a factor $\varphi(0)$ or $\varphi(1)$. We thus introduce the notion of a φ-factorization. Even if this factorization can be defined for an arbitrary morphism σ, we restrict ourselves to φ. Indeed, we will be able to precisely describe the different coefficients of Theorem 3.1.10 in this case, while no interesting generalization arises for an arbitrary morphism σ.

Definition 3.1.2 (φ-factorization)**.** If a word $v \in \mathcal{A}^*$ contains a factor 01 or 10, then it can be factorized as

$$v = w_0\, \varphi(a_1)\, w_1 \cdots w_{k-1}\, \varphi(a_k)\, w_k \tag{3.1}$$

for some $k \geq 1$, $a_1, \ldots, a_k \in \mathcal{A}$ and $w_0, \ldots, w_k \in \mathcal{A}^*$ (some of these words are possibly empty). We call this factorization, a *φ-factorization* of v. It is coded by the k-tuple of positions where the $\varphi(a_i)$'s occurs:

$$\kappa = (|w_0|, |w_0\varphi(a_1)w_1|, |w_0\varphi(a_1)w_1\varphi(a_2)w_2|, \ldots, |w_0\varphi(a_1)w_1\varphi(a_2)w_2 \cdots w_{k-1}|).$$

The set of all φ-factorizations of v is denoted by φ-Fac(v).

Example 3.1.3. Consider the word 010110. It has 9 φ-factorizations. These factorizations and the coding tuples are depicted in Figure 3.1.

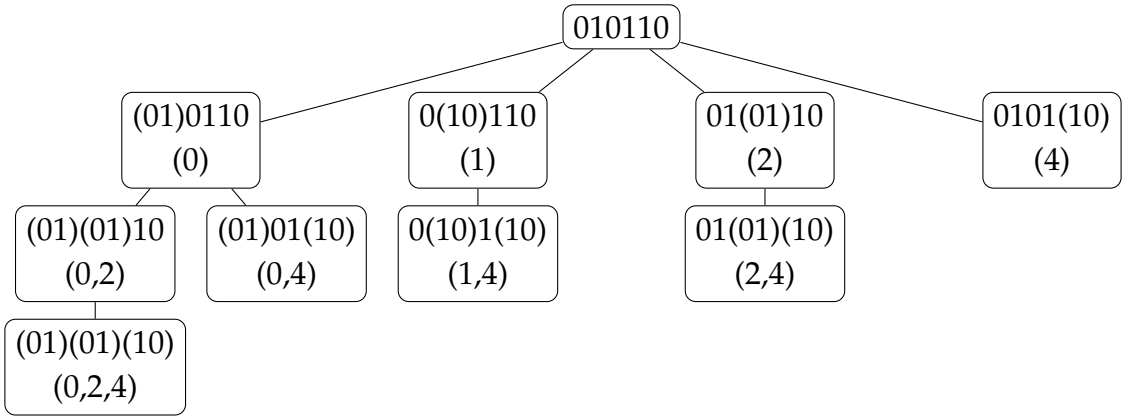


Figure 3.1: The tree of φ-factorizations of 010110.

Notation 3.1.4. Recall that a multiset M is just a set where elements can be repeated with a finite integer multiplicity. If x belongs to M, its multiplicity is denoted by

$m_M(x)$ or simply $m(x)$. If $x \notin M$, then $m_M(x) = 0$. If we enumerate the elements of a multiset, we adopt the convention to write multiplicities as subscripts. The *multiset sum* $M \uplus N$ of two multisets M and N is the union of the two multisets and the multiplicity of an element is equal to the sum of the respective multiplicities, i.e., for $x \in M \cup N$, $m_{M \uplus N}(x) = m_M(x) + m_N(x)$.

We define a map f from $\mathcal{A}^*$ to the set of finite multisets of words over $\mathcal{A}^*$. This map is defined as follows.

Definition 3.1.5. Let f be the function such that if $v \in 0^* \cup 1^*$, then $f(v) = \emptyset$ (the meaning for this choice will be clear with Theorem 3.1.10). Otherwise, v contains a factor from $\{01, 10\}$. With every φ-factorization $\kappa \in \varphi\text{-Fac}(v)$ of v of the form (3.1)

$$v = w_0\, \varphi(a_1)\, w_1 \cdots w_{k-1}\, \varphi(a_k)\, w_k$$

for some $k \geq 1$, $a_1, \ldots, a_k \in \mathcal{A}$ and $w_0, \ldots, w_k \in \mathcal{A}^*$, we define the language

$$\mathcal{L}(v, \kappa) := \mathcal{A}^{|w_0|}\, a_1\, \mathcal{A}^{|w_1|} \cdots \mathcal{A}^{|w_{k-1}|} a_k \mathcal{A}^{|w_k|}$$

of words of length $|v| - k$ (there are $2^{|v|-2k}$ of these words[2]). Such a language is considered as a multiset whose elements have multiplicities equal to 1. Now, $f(v)$ is defined as the multiset sum of the above languages for all φ-factorizations of v, i.e.,

$$f(v) := \biguplus_{\kappa \in \varphi\text{-Fac}(v)} \mathcal{L}(v, \kappa).$$

Example 3.1.6. Consider the word $v = 01011$ and the Thue–Morse morphism φ. It has four φ-factorizations of the form (3.1)

$$(01)011,\ 0(10)11,\ 01(01)1, (01)(01)1.$$

The first three are coded respectively by the 1-tuples (0), (1) and (2). The last one is coded by $(0, 2)$. The corresponding four languages are

$$\begin{aligned} 0\,\mathcal{A}^3 &= \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111\}, \\ \mathcal{A}\,1\,\mathcal{A}^2 &= \{0100, 0101, 0110, 0111, 1100, 1101, 1110, 1111\}, \\ \mathcal{A}^2\,0\,\mathcal{A} &= \{0000, 0001, 0100, 0101, 1000, 1001, 1100, 1101\}, \\ 00\,\mathcal{A} &= \{000, 001\}. \end{aligned}$$

Consequently, $f(v)$ is the multiset

$$\{000_1, 001_1, 0000_2, 0001_2, 0010_1, 0011_1, 0100_3, 0101_3, 0110_2, 0111_2, \\ 1000_1, 1001_1, 1100_2, 1101_2, 1110_1, 1111_1\}.$$

[2]This quantity counts all words of length $|v| - k$ where k letters at fixed positions are given.

Definition 3.1.7. Now that f is defined over $\mathcal{A}^*$, we can extend it to any finite multiset M of words over $\mathcal{A}$. It is the multiset sum of the $f(v)$'s, for all $v \in M$, repeated with their multiplicities.

Example 3.1.8. Continuing Example 3.1.6 with $v = 01011$, we get

$$f^2(v) = \{00_4, 01_2, 10_2, 000_{20}, 001_{16}, 010_{28}, 011_{24}, 100_{12}, 101_8, 110_{20}, 111_{16}\}$$

and

$$f^3(v) = \{0_2, 1_2, 00_{76}, 01_{100}, 10_{44}, 11_{68}\}.$$

Observe that if we apply f an extra time, $f^4(v) = \{0_{100}, 1_{44}\}$ and for all $n \geq 5$, $f^n(v) = \emptyset$.

Remark 3.1.9. The observation made in the previous example is general. If v does not belong to $0^* \cup 1^*$, then $f^{|v|-2}(v)$ contains only elements in $\{0, 1, 00, 01, 10, 11\}$ and $f^{|v|-1}(v)$ contains only elements in $\{0, 1\}$. For $n \geq |v|$, $f^n(v)$ is empty.

Recall that $f(v)$ is a multiset. Hence $m_{f(v)}(x)$ denotes the multiplicity of x as an element of $f(v)$. The following result expresses the value of $\binom{\varphi(u)}{v}$ in terms of binomial coefficients applied on u. It directly follows from the definitions of φ-factorization and $\mathcal{L}(v, \kappa)$ (the reader should probably reconsider the introductory Example 3.1.1). Either we pick all the symbols of v in pairwise distinct 2-blocks of $\varphi(u)$, which corresponds to the first term of the sum, or some factors of v are realized by $\varphi(0)$ or $\varphi(1)$. Therefore we need to consider all the possible φ-factorizations. For the sake of completeness, we provide a detailed proof.

Theorem 3.1.10. *With the above notation, for all words u, v, we have*

$$\binom{\varphi(u)}{v} = \binom{|u|}{|v|} + \sum_{\substack{\kappa \in \varphi\text{-Fac}(v) \\ x \in \mathcal{L}(v,\kappa)}} \binom{u}{x} = \binom{|u|}{|v|} + \sum_{x \in f(v)} m_{f(v)}(x) \binom{u}{x}.$$

Proof. Assume that $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_k$. Let us also write $\varphi(u) = x_1 \cdots x_{2n}$. Our aim is to count the number of k-tuples $(i_1, \ldots, i_k)$ such that

$$1 \leq i_1 < i_2 < \cdots < i_k \leq 2n$$

and $x_{i_1} \cdots x_{i_k} = v$.

We first count the k-tuples $(i_1, \ldots, i_k)$ where letters of v are picked in distinct 2-blocks $\varphi(x_m)$, i.e., such that

$$\forall r < k, \forall m \leq n, \ (i_r, i_{r+1}) \neq (2m-1, 2m). \tag{3.2}$$

Using the fact that $|\varphi(0)|_0 = |\varphi(0)|_1 = |\varphi(1)|_0 = |\varphi(1)|_1 = 1$, the set of such k-tuples satisfying (3.2) is in bijection with the subwords of length k of u. Thus, the number of such tuples is equal to $\binom{|u|}{|v|}$.

Now let us count the other possible tuples, if any. Observe that if $r < k$ and $m \leq n$ are such that $(i_r, i_{r+1}) = (2m-1, 2m)$, then $v_{i_r} v_{i_{r+1}} = \varphi(x_m)$. In particular, this implies that v is not in $0^* \cup 1^*$. For every $R \leq k/2$, we count all k-tuples $(i_1, \ldots, i_k)$ for which exactly R pairs of indices are of the form $(2m-1, 2m)$. Let thus $1 \leq \ell_1 < \cdots < \ell_R < k$ and $1 \leq m_1 < \cdots < m_R \leq n$ be such that

$$\begin{cases} (i_{\ell_r}, i_{\ell_r+1}) = (2m_r - 1, 2m_r), & \forall 1 \leq r \leq R, \\ (i_\ell, i_{\ell+1}) \neq (2m-1, 2m), & \forall \ell \notin \{\ell_1, \ldots, \ell_R\}, \forall m \leq n. \end{cases} \tag{3.3}$$

Again using the fact that $|\varphi(0)|_0 = |\varphi(0)|_1 = |\varphi(1)|_0 = |\varphi(1)|_1 = 1$, such a k-tuple satisfies $v = x_{i_1} \cdots x_{i_k}$ if and only if v admits a φ-factorization κ of the form:

$$v = V_0 \varphi(u_{m_1}) V_1 \varphi(u_{m_2}) \cdots V_{R-1} \varphi(u_{m_R}) V_R,$$

where $V_0 = v_1 \cdots v_{\ell_1 - 1}$, $V_r = v_{\ell_r + 2} \cdots v_{\ell_{r+1} - 1}$ for $1 \leq r < R$ and $V_R = u_{\ell_R + 2} \cdots v_k$. Furthermore, every word $x \in \mathcal{L}(v, \kappa)$ yields exactly $\binom{u}{x}$ k-tuples satisfying (3.3). By construction, a given k-tuple cannot be associated with two different words x, x' from $\mathcal{L}(v, \kappa)$. Also, summing on all $R \in [\lfloor k/2 \rfloor]$ and on all $2R$-tuples $(\ell_1, m_1, \ldots, \ell_R, m_R)$ satisfying (3.3) corresponds to summing on all φ-factorizations of v. This ends the proof of the first equality. The second one directly follows from the definition of $m_{f(v)}$. □

Corollary 3.1.11. *Let $k \geq 1$. For all words u, u', we have*

$$u \sim_k u' \Rightarrow \varphi(u) \sim_{k+1} \varphi(u').$$

Proof. Let v be a word of length at most $k+1$. From Theorem 3.1.10, we have

$$\binom{\varphi(u)}{v} = \binom{|u|}{|v|} + \sum_{x \in f(v)} m_{f(v)}(x) \binom{u}{x}.$$

For all $x \in f(v)$, we have $|x| \leq k$ and thus $\binom{u}{x} = \binom{u'}{x}$. The conclusion follows since $|u| = |u'|$. □

Theorem 3.1.10 can be extended to iterates of φ. If we apply it twice, we get

$$\begin{aligned}\binom{\varphi^2(u)}{v} &= \binom{|\varphi(u)|}{|v|} + \sum_{x\in f(v)} m_{f(v)}(x)\binom{\varphi(u)}{x} \\ &= \binom{|\varphi(u)|}{|v|} + \sum_{x\in f(v)} m_{f(v)}(x)\left[\binom{|u|}{|x|} + \sum_{y\in f(x)} m_{f(x)}(y)\binom{u}{y}\right] \\ &= \binom{|\varphi(u)|}{|v|} + \sum_{x\in f(v)} m_{f(v)}(x)\binom{|u|}{|x|} + \sum_{z\in f^2(v)} m_{f^2(v)}(z)\binom{u}{z}.\end{aligned}$$

The last equality comes from the fact that

$$\sum_{x\in f(v)}\sum_{y\in f(x)} m_{f(v)}(x)\, m_{f(x)}(y) = \sum_{z\in f^2(v)} m_{f^2(v)}(z).$$

Indeed, y appears $m_{f(x)}(y)$ times in the multiset $f(x)$ and x itself appears $m_{f(v)}(x)$ times in $f(v)$. Thus y appears $m_{f(v)}(x)\, m_{f(x)}(y)$ in $f^2(v)$.
We set $f^0(v) = \{v\}$ (where v has multiplicity one).

Corollary 3.1.12. *With the above notation, for $\ell \geq 1$ and all words u, v, we have*

$$\binom{\varphi^\ell(u)}{v} = \sum_{i=0}^{\ell-1}\sum_{z\in f^i(v)} m_{f^i(v)}(z)\binom{|\varphi^{\ell-i-1}(u)|}{|z|} + \sum_{z\in f^\ell(v)} m_{f^\ell(v)}(z)\binom{u}{z}.$$

When proving that two words u, u' are not k-binomially equivalent, it is convenient to find a word v of length k such that the difference $\binom{u}{v} - \binom{u'}{v}$ is non-zero. It is therefore interesting to make the following observation.

Remark 3.1.13. Since $|\varphi^i(u)| = 2^i|u|$, note that the first of the two terms in the above Corollary only depends on $|u|$ and v. Otherwise stated, if u, u' are two words of the same length, then

$$\binom{\varphi^\ell(u)}{v} - \binom{\varphi^\ell(u')}{v} = \sum_{z\in f^\ell(v)} m_{f^\ell(v)}(z)\left[\binom{u}{z} - \binom{u'}{z}\right].$$

3.1.2 About multiplicities

In this subsection we give more insight about multiplicities of the form $m_{f^\ell(v)}(z)$ appearing in Corollary 3.1.12. This will permit us to prove results about k-binomially (non-)equivalent factors of the Thue–Morse word of the form $\varphi^k(a)$, where a is a letter.

Lemma 3.1.14. *Let w be a word. Let M be a (finite) multiset of words such that u belongs to M if and only if its complement $\overline{u}$ belongs to M with the same multiplicity. Then, for all $i \geq 0$, the multiplicity of w in $f^i(M)$ is equal to the one of $\overline{w}$.*

Proof. Let u be a word in M. Because of the special form of the morphism φ, we deduce that the set of tuples coding the φ-factorizations of u is equal to the set of tuples coding the φ-factorizations of $\overline{u}$. Moreover, a word v belongs to $\mathcal{L}(u,\kappa)$ if and only if $\overline{v}$ belongs to $\mathcal{L}(\overline{u},\kappa)$. Indeed, these two languages are respectively of the form

$$\mathcal{A}^{|w_0|}\, a_1\, \mathcal{A}^{|w_1|} \cdots \mathcal{A}^{|w_{k-1}|} a_k \mathcal{A}^{|w_k|} \quad \text{and} \quad \mathcal{A}^{|w_0|}\, \overline{a_1}\, \mathcal{A}^{|w_1|} \cdots \mathcal{A}^{|w_{k-1}|} \overline{a_k} \mathcal{A}^{|w_k|}.$$

Think about Example 3.1.6 and consider the word $\overline{u} = 10100$,

$$u = (01)011,\ 0(10)11,\ 01(01)1, (01)(01)1$$

and

$$\overline{u} = (10)100,\ 1(01)00,\ 10(10)0, (10)(10)0.$$

For instance, the third φ-factorization gives, respectively, the languages

$$\mathcal{A}^2\, 0\, \mathcal{A} \quad \text{and} \quad \mathcal{A}^2\, 1\, \mathcal{A}.$$

Let w be a word over $\mathcal{A}$. Since u and $\overline{u}$ have the same multiplicity, the total number of times w occurs in the $m(u)$ copies of $\mathcal{L}(u,\kappa)$ is equal to the number of times $\overline{w}$ occurs in the copies of $\mathcal{L}(\overline{u},\kappa)$. This observation holds for every φ-factorization. Consequently $f(M)$ has the same property as M: words and their complement appear with the same multiplicity in $f(M)$. We can thus iterate the construction and the argument. □

Example 3.1.15. Consider the multiset $M = \{011_2, 100_2, 0110_1, 1001_1\}$. In $f(M)$ the words 00 and 11 have multiplicity 2, 01 and 10 have multiplicity 3 and all words of length 3 appear twice. Then

$$f^2(M) = \{0_3, 1_3, 00_8, 11_8, 01_8, 10_8\}.$$

Proposition 3.1.16. *For all $n \geq 1$, the multiplicity of 01 (resp., of 00) in the multiset $f^n(01^{n+1}) = f^{n-1}(0\mathcal{A}^n)$ is larger than the one of 10 (resp., of 11). More precisely, these multiplicities in the multiset $f^n(01^{n+1})$ satisfy*

$$m(01) - m(10) = m(00) - m(11) = 1 \cdot 2 \cdot 4 \cdot 8 \cdots 2^{n-1} = 2^{n(n-1)/2}.$$

Proof. We proceed by induction on n. For $n = 1$, $f^0(0\mathcal{A}) = \{00, 01\}$ and the result is obvious. Let $n \geq 2$. Assume that the result holds for all $j < n$. We consider $f^{n-1}(0\mathcal{A}^n)$.

Note that $0\mathcal{A}^n$ is the disjoint union of $\{00u \mid u \in \mathcal{A}^{n-1}\}$ and $\{01\overline{u} \mid u \in \mathcal{A}^{n-1}\}$. These two sets are in one-to-one correspondence with the map $0w \mapsto 0\overline{w}$. Since we proceed by induction, let us start by applying f once. We will apply f^{n-2} later on.

Let $u \in \mathcal{A}^{n-1}$. First observe that there is a one-to-one correspondence between the set of φ-factorizations of $00u$ and the set of φ-factorizations of $01\overline{u}$ coded by tuples

whose first element is at least equal to 1. In this case, we have exactly the same tuples. For instance, consider the word $u = 1011$.

$00u = 001011$	κ	$01\overline{u} = 010100$
0(01)011	(1)	0(10)100
00(10)11	(2)	01(01)00
001(01)1	(3)	010(10)0
0(01)(01)1	(1,3)	0(10)(10)0

For each such φ-factorization κ of $00u$, we have a language of the form

$$\mathcal{L}(00u, \kappa) = \mathcal{A}^{i_0} a_1 \mathcal{A}^{i_1} \cdots \mathcal{A}^{i_{k-1}} a_k \mathcal{A}^{i_k}$$

with $i_0 \geq 1$, $k \geq 1$, $a_1, \ldots, a_k \in \mathcal{A}$, $i_1, \ldots, i_k \geq 0$ and the corresponding φ-factorization of $01\overline{u}$ gives the language

$$\mathcal{L}(01\overline{u}, \kappa) = \mathcal{A}^{i_0} \overline{a_1} \mathcal{A}^{i_1} \cdots \mathcal{A}^{i_{k-1}} \overline{a_k} \mathcal{A}^{i_k}.$$

Observe that the union of these two languages satisfies the assumption of the previous lemma. Thus, applying iteratively f to the words belonging to these languages will eventually provide words 01 and 10 (resp., 00 and 11) with the same multiplicity. We stress the fact that in the above φ-factorizations, i_0 is non-zero.

We still have to consider the φ-factorizations of $01\overline{u}$ coded with tuples starting with 0 (these are the only remaining ones). With the running example $u = 1011$, we have the extra three φ-factorizations:

$$(01)0100,\ (01)(01)00,\ (01)0(10)0.$$

Let us consider φ-factorizations of $01\overline{u}$ coded by a k-tuple starting with 0 and for $k \geq 2$. Thus the resulting languages are made of words of length at most

$$|u| + 2 - k = n + 1 - k \leq n - 1.$$

By Remark 3.1.9, applying f^{n-2} to words of such lengths will only provide words in $\{0, 1\}$. Hence, they do not provide any copy of $00, 01, 10$ or 11.

We finally have the φ-factorization of $01\overline{u}$ coded by the 1-tuple (0). The corresponding language $\mathcal{L}(01\overline{u}, (0))$ is $0\mathcal{A}^{|u|}$. Recall that u (and thus $\overline{u}$) ranges over $\mathcal{A}^{n-1}$. Thus there are 2^{n-1} copies of this language. By the induction hypothesis, the difference of multiplicities of 01 and 10 (resp., 00 and 11) for $f^{n-2}(0\mathcal{A}^{n-1})$ is $1 \cdot 2 \cdots 2^{n-2}$. Multiplying the latter number by the number of copies provides us with the result. □

Table 3.1 provides the computed multiplicities of 01 and 10 in $f^{n-1}(0\mathcal{A}^n)$ for the first few values of n. We also indicate the corresponding differences given in the previous proposition.

	$m(01)$	$m(10)$	$m(01) - m(10)$
1	1	0	$1 = 2^0$
2	3	1	$2 = 2^1$
3	28	20	$8 = 2^3$
4	800	736	$64 = 2^6$
5	61952	60928	$1024 = 2^{10}$
6	11812864	11780096	$32768 = 2^{15}$
7	5285871616	5283774464	$2097152 = 2^{21}$

Table 3.1: Multiplicities in $f^{n-1}(0\mathcal{A}^n)$.

Proposition 3.1.17. *For all $n \geq 1$, the multiplicity of 0 in the multiset*

$$f^n(01^{n+1}) = f^{n-1}(0\mathcal{A}^n)$$

is larger than or equal to the multiplicity of 1.

Proof. For $n = 1, 2$, there is no 0 and no 1 in $f^n(01^{n+1})$. Assume $n \geq 3$. The multiplicity of 0 (resp., 1) in the multiset $f^n(01^{n+1})$ is equal to the multiplicity of 01 (resp., 10) in $f^{n-1}(01^{n+1}) = f^{n-2}(0\mathcal{A}^n)$. We use the same reasoning as the one in the proof of Proposition 3.1.16 except for φ-factorizations starting with 0 and for $k \geq 2$ where a more careful discussion is needed.

Consider the φ-factorizations of the words $01\overline{u}$ of length $n+1$ coded by a k-tuple starting with 0 and for $k \geq 2$. If $k > 2$, when applying f once, the resulting languages are made of words of length less than $n-1$ and applying f^{n-3} to these words will provide no 01 nor 10. But for $k = 2$, applying f once to all such words, we get the multiset $0\mathcal{A}^{n-2}$ where each word has multiplicity 2^{n-3}. Indeed, the word $\overline{u}$ is ranging over $\mathcal{A}^{n-1}$ and we consider φ-factorizations where one replacement of 01 or 10 is made inside $\overline{u}$. For all $i, j \leq n-3$, we have

$$\begin{aligned}\#\{x01y \in \mathcal{A}^{n-1} \mid |x| = i\} = \#\{x01y \in \mathcal{A}^{n-1} \mid |x| = j\} \\ =\#\{x10y \in \mathcal{A}^{n-1} \mid |x| = i\} = \#\{x10y \in \mathcal{A}^{n-1} \mid |x| = j\} = 2^{n-3}.\end{aligned}$$

To conclude the proof, observe that the difference of multiplicity of 01 and 10 in $f^{n-3}(0\mathcal{A}^{n-2})$ is obtained from the previous proposition. □

Some consequences for the factors of Thue–Morse

We collect some important properties of iterates of φ with respect to the k-binomial equivalence $\sim_k$. A trace of the first result below can be found in [80] or in [91].

Lemma 3.1.18 (Ochsenschläger). *Let $k \geq 1$. We have*

$$\varphi^k(0) \sim_k \varphi^k(1) \quad \textit{and} \quad \varphi^k(0) \nsim_{k+1} \varphi^k(1).$$

In particular, if $|u| = |v|$, then $\varphi^k(u) \sim_k \varphi^k(v)$.

Proof. We have $\varphi(0) \sim_1 \varphi(1)$. Thus the first part follows from Corollary 3.1.11.

Let us show that $\varphi^k(0) \nsim_{k+1} \varphi^k(1)$. The case $k = 1$ is obvious: $01 \nsim_2 10$. Observe that $f(01^k) = 0\mathcal{A}^{k-1}$ thus $f^{k-1}(01^k) = f^{k-2}(0\mathcal{A}^{k-1})$. Using Remark 3.1.13, we compute

$$\binom{\varphi^k(0)}{01^k} - \binom{\varphi^k(1)}{01^k}$$

and get

$$\binom{\varphi^{k-1}(\varphi(0))}{01^k} - \binom{\varphi^{k-1}(\varphi(1))}{01^k} = \sum_{z \in f^{k-1}(01^k)} m_{f^{k-1}(01^k)}(z) \left[\binom{\varphi(0)}{z} - \binom{\varphi(1)}{z} \right].$$

The elements of the multiset $f^{k-1}(01^k)$ belong to $\{0, 1, 00, 01, 10, 11\}$. The last factor in brackets in the previous sum is non-zero only if $z = 01$ or $z = 10$. Hence, we get

$$\binom{\varphi^k(0)}{01^k} - \binom{\varphi^k(1)}{01^k} = m_{f^{k-1}(01^k)}(01) - m_{f^{k-1}(01^k)}(10) = 2^{(k-1)(k-2)/2} \geq 1.$$

The last equality comes from Proposition 3.1.16. □

Lemma 3.1.19 (Transfer lemma). *Let $k \geq 1$. Let u, v, v' be three non-empty words such that $|v| = |v'|$. We have*

$$\varphi^{k-1}(u)\, \varphi^k(v) \sim_k \varphi^k(v')\, \varphi^{k-1}(u).$$

Proof. Observe that $u\varphi(v) \sim_1 \varphi(v')u$ because v and v' have the same length. The conclusion follows from Corollary 3.1.11: $\varphi^{k-1}(u\, \varphi(v)) \sim_k \varphi^{k-1}(\varphi(v')\, u)$. □

Corollary 3.1.20. *Let $k \geq 1$ and $n \geq 2$. Let $u_1, \ldots, u_n$ be non-empty words. Let $i_1, \ldots, i_n$ be integers greater than or equal to k, except for one of these being equal to $k-1$ and denoted by i_r. For all permutations ν of $\{1, \ldots, n\}$, we have*

$$\varphi^{i_1}(u_1)\varphi^{i_2}(u_2) \cdots \varphi^{i_n}(u_n) \sim_k \varphi^{i_{\nu(1)}}(u'_{\nu(1)})\varphi^{i_{\nu(2)}}(u'_{\nu(2)}) \cdots \varphi^{i_{\nu(n)}}(u'_{\nu(n)})$$

for all words $u'_1, \ldots, u'_n$ where $|u_i| = |u'_i|$, for all i, and $u_{i_r} = u'_{i_r}$.

Proof. It is enough to see that one can permute any two consecutive factors: any permutation is a product of such type of transpositions. This is a direct consequence of the two previous lemmas. □

3.2 2-binomial complexity

In this section we compute the value of $b_{\mathbf{t}}^{(2)}(n)$. First of all, the next proposition ensures us that all the words we will consider in the proof of Theorem 3.2.2 really appear as factors of $\mathbf{t}$.

The reader familiar with Büchi's theorem [20] and the characterization of k-automatic sequences in terms of first-order logic [19] can obtain an alternative proof of this result. Basically, one has to check that the four closed formulas (for $a, b \in \{0,1\}$)

$$(\forall m)(\exists i)(\mathbf{t}_i = a \wedge \mathbf{t}_{i+m+1} = b)$$

hold. This can be done automatically using the Walnut package [90].

Proposition 3.2.1 (Folklore)**.** *Let $k \in \mathbb{N}$, $m \in \mathbb{N}_0$ and $a, b \in \{0,1\}$. Let p be a suffix of $\varphi^k(a)$ and s be a prefix of $\varphi^k(b)$. Then there exists $z \in \{0,1\}^m$ such that $p\varphi^k(z)s$ is a factor of $\mathbf{t}$.*

Proof. Let $a, b \in \{0,1\}, m \in \mathbb{N}_0$. We will prove by induction that there exists $z \in \{0,1\}^m$ such that $azb \in \mathrm{Fac}(\mathbf{t})$. Therefore, $\varphi^k(a)\varphi^k(z)\varphi^k(b)$ (and thus, $p\varphi^k(z)s$) is a factor of $\mathbf{t}$.

For the base case ($m = 0, 1$), note that $00, 11, 010, 011, 100, 101$ are factors of Thue–Morse.

For the induction step, observe that if the factor u contains letters a and b at distance n, then $\varphi(u)$ contains letters a and b at distance $2n$, as well as letters a and $\overline{b}$ at distance $2n + 1$. □

Using this result, we can compute the values of $b_{\mathbf{t}}^{(2)}$.

Theorem 3.2.2. *[65, Theorem 3.3.6] We have $b_{\mathbf{t}}^{(2)}(0) = 1$, $b_{\mathbf{t}}^{(2)}(1) = 2$, $b_{\mathbf{t}}^{(2)}(2) = 4$, $b_{\mathbf{t}}^{(2)}(3) = 6$ and for all $n \geq 4$,*

$$b_{\mathbf{t}}^{(2)}(n) = \begin{cases} 9, & \text{if } n \equiv 0 \pmod 4; \\ 8, & \text{otherwise.} \end{cases}$$

Proof. Assume $n \geq 4$. First observe that, for all words u, v of the same length,

$$u \sim_2 v \Leftrightarrow \binom{u}{0} = \binom{v}{0} \text{ and } \binom{u}{01} = \binom{v}{01}.$$

Indeed, this is due to the fact that $\binom{u}{1} = |u| - \binom{u}{0}$, $\binom{u}{aa} = \binom{|u|_a}{2}$ for every $a \in \{0,1\}$ and $\binom{u}{10} = \binom{|u|}{2} - \binom{u}{00} - \binom{u}{01} - \binom{u}{11}$.

We will consider four cases depending on the value of $\lambda \in \{0,1,2,3\}$ such that $\lambda = n \bmod 4$. For each of them, we will compute

$$b_{\mathbf{t}}^{(2)}(n) = \#\left\{\left(\binom{u}{0}, \binom{u}{01}\right) \in \mathbb{N}\times\mathbb{N} : u \in \mathrm{Fac}_n(\mathbf{t})\right\}.$$

Since $\mathbf{t}$ is a fixed point of the morphism φ, we know that every factor u of length n of $\mathbf{t}$ can be written $p_u\varphi^2(z)s_u$ for some $z \in \mathcal{A}^*$ and p_u (resp., s_u) suffix (resp., prefix) of a word in $\{\varphi^2(0), \varphi^2(1)\}$. From the previous proposition, we also know that every word of this form occurs at least once in $\mathbf{t}$. Moreover, we have $|p_u| + |s_u| \in \{\lambda, \lambda+4\}$ and, as a consequence, $|z| = \lfloor \frac{n}{4} \rfloor = \frac{n-\lambda}{4}$ or $|z| = \lfloor \frac{n}{4} \rfloor - 1$. Set $\ell = \frac{n-\lambda}{4}$.

Let us first consider the case $\lambda = 0$. We have

$$\begin{aligned}\mathrm{Fac}_n(\mathbf{t}) = \{&\varphi^2(az), 0\varphi^2(z)011, 0\varphi^2(z)100, 1\varphi^2(z)011, 1\varphi^2(z)100,\\ &01\varphi^2(z)01, 01\varphi^2(z)10, 10\varphi^2(z)01, 10\varphi^2(z)10,\\ &110\varphi^2(z)0, 110\varphi^2(z)1, 001\varphi^2(z)0, 001\varphi^2(z)1 : z \in \mathcal{A}^{\ell-1}, a \in \mathcal{A}, az \in \mathrm{Fac}(\mathbf{t})\}.\end{aligned}$$

Let us illustrate the computation of $\left(\binom{u}{0}, \binom{u}{01}\right)$ on $u = 0\varphi^2(z)011 \in \mathrm{Fac}_n(\mathbf{t})$. Firstly,

$$\binom{u}{0} = \binom{0}{0} + \binom{\varphi^2(z)}{0} + \binom{011}{0} = 2 + 2|z| = 2\ell$$

since $|z| = \ell - 1$. Similarly, we have

$$\begin{aligned}\binom{u}{01} &= \binom{0}{01} + \binom{\varphi^2(z)}{01} + \binom{011}{01} + \binom{0}{0}\binom{\varphi^2(z)}{1} + \binom{0}{0}\binom{011}{1} + \binom{\varphi^2(z)}{0}\binom{011}{1}\\ &= \binom{|\varphi(z)|}{2} + \binom{\varphi(z)}{0} + 2 + |\varphi(z)| + 2 + 2|\varphi(z)|\\ &= |z|(2|z|-1) + |z| + 6|z| + 4 = 2\ell^2 + 2\ell.\end{aligned}$$

All the computations are summarized in the table below. We give the form of the factors and respective values for the pairs $\left(\binom{u}{0}, \binom{u}{01}\right)$.

Case	$\varphi^2(az)$ $01\varphi^2(z)10$ $10\varphi^2(z)01$	$0\varphi^2(z)011$ $001\varphi^2(z)1$	$1\varphi^2(z)100$ $110\varphi^2(z)0$	$0\varphi^2(z)100$	$001\varphi^2(z)0$
$\binom{u}{0}$ $\binom{u}{01}$	2ℓ $2\ell^2$	2ℓ $2\ell^2+2\ell$	2ℓ $2\ell^2-2\ell$	$2\ell+1$ $2\ell^2-1$	$2\ell+1$ $2\ell^2$
Case	$1\varphi^2(z)011$	$110\varphi^2(z)1$	$01\varphi^2(z)01$	$10\varphi^2(z)10$	
$\binom{u}{0}$ $\binom{u}{01}$	$2\ell-1$ $2\ell^2$	$2\ell-1$ $2\ell^2+1$	2ℓ $2\ell^2+1$	2ℓ $2\ell^2-1$	

It is thus clear that if $n \equiv 0 \pmod 4$, we have $b_{\mathbf{t}}^{(2)}(n) = 9$.

Let us now present the results in the case $\lambda = 1$. Let a be a letter and z be any word of $\mathcal{A}^{\ell-1}$. We obtain the results below, showing that $b_{\mathbf{t}}^{(2)}(n) = 8$.

Case	$\varphi^2(az)0$ $01\varphi^2(z)100$	$0\varphi^2(az)$ $110\varphi^2(z)01$	$\varphi^2(az)1$ $10\varphi^2(z)011$	$1\varphi^2(az)$ $001\varphi^2(z)10$
$\binom{u}{0}$	$2\ell+1$	$2\ell+1$	2ℓ	2ℓ
$\binom{u}{01}$	$2\ell^2$	$2\ell^2+2\ell$	$2\ell^2+2\ell$	$2\ell^2$
Case	$10\varphi^2(z)100$	$001\varphi^2(z)01$	$01\varphi^2(z)011$	$110\varphi^2(z)10$
$\binom{u}{0}$	$2\ell+1$	$2\ell+1$	2ℓ	2ℓ
$\binom{u}{01}$	$2\ell^2-1$	$2\ell^2+2\ell+1$	$2\ell^2+2\ell+1$	$2\ell^2-1$

In the case of $\lambda = 2$, if z is a word of $\mathcal{A}^{\ell-1}$ and if a is a letter, we obtain $b_{\mathbf{t}}^{(2)}(n) = 8$ due to the following results:

Case	$\varphi^2(az)01$ $01\varphi^2(az)$	$\varphi^2(az)10$ $10\varphi^2(az)$	$1\varphi^2(az)1$ $110\varphi^2(z)011$	$0\varphi^2(az)0$ $001\varphi^2(z)100$
$\binom{u}{0}$	$2\ell+1$	$2\ell+1$	2ℓ	$2\ell+2$
$\binom{u}{01}$	$2\ell^2+2\ell+1$	$2\ell^2+2\ell$	$2\ell^2+2\ell$	$2\ell^2+2\ell$
Case	$1\varphi^2(az)0$	$0\varphi^2(az)1$	$001\varphi^2(z)011$	$110\varphi^2(z)100$
$\binom{u}{0}$	$2\ell+1$	$2\ell+1$	$2\ell+1$	$2\ell+1$
$\binom{u}{01}$	$2\ell^2$	$2\ell^2+4\ell+1$	$2\ell^2+4\ell+2$	$2\ell^2-1$

Finally, if $\lambda = 3$ and using the same notation, we obtain that $b_{\mathbf{t}}^{(2)}(n) = 8$ due to the following computations,

Case	$\varphi^2(az)011$ $01\varphi^2(az)1$	$110\varphi^2(az)$ $1\varphi^2(az)10$	$\varphi^2(az)100$ $10\varphi^2(az)0$	$001\varphi^2(az)$ $0\varphi^2(az)01$
$\binom{u}{0}$	$2\ell+1$	$2\ell+1$	$2\ell+2$	$2\ell+2$
$\binom{u}{01}$	$2\ell^2+4\ell+2$	$2\ell^2+2\ell$	$2\ell^2+2\ell$	$2\ell^2+4\ell+2$
Case	$1\varphi^2(az)01$	$10\varphi^2(az)1$	$0\varphi^2(az)10$	$01\varphi^2(az)0$
$\binom{u}{0}$	$2\ell+1$	$2\ell+1$	$2\ell+2$	$2\ell+2$
$\binom{u}{01}$	$2\ell^2+2\ell+1$	$2\ell^2+4\ell+1$	$2\ell^2+4\ell+1$	$2\ell^2+2\ell+1$

which concludes the proof. □

3.3 How to cut factors of the Thue–Morse word

Computing $b_{\mathbf{t}}^{(k)}(n)$, for all $k \geq 3$, will require much more knowledge about the factors of $\mathbf{t}$. This section is concerned about particular factorizations of factors occurring in $\mathbf{t}$.

Similar concepts (n-partitions and points of degree d) were previously introduced in [115, 116] for studying other combinatorial problems.

3.3.1 Cutting sets and associated factorizations

Since $\mathbf{t}$ is a fixed point of φ, it is very often convenient to view $\mathbf{t}$ as a concatenation of blocks belonging to $\{\varphi^k(0), \varphi^k(1)\}$. Hence, we first define a function bar_k that roughly plays the role of a ruler marking the positions where a new block of length 2^k occurs (these positions are called *cutting bars of order k*). For all $k \geq 1$, let us consider the function $\mathrm{bar}_k : \mathbb{N}_0 \to \mathbb{N}_0$ defined by

$$\mathrm{bar}_k(n) = |\varphi^k(\mathbf{t}_{[0,n)})| = n \cdot 2^k,$$

where $\mathbf{t}_{[0,n)}$ is the prefix of length n of $\mathbf{t}$.

Given a factor u of $\mathbf{t}$, we are interested in the relative positions of $\mathrm{bar}_k(\mathbb{N}_0)$ in u: we look at all the occurrences of u in $\mathbf{t}$ and see what configurations can be achieved, that is how an interval I such that $\mathbf{t}_I = u$ can intersect $\mathrm{bar}_k(\mathbb{N}_0)$.

For instance, for $k = 1$, the word $u = 010$ occurs in $\mathbf{t}$ with two different factorizations:

$$\begin{array}{rcccccccccc} \mathbf{t} & = & \varphi(0) & \varphi(1) & \varphi(1) & \varphi(0) & \varphi(1) & \varphi(0) & \varphi(0) & \varphi(1) & \cdots \\ & = & 01 & 10 & 10 & 01 & 10 & 01 & 01 & 10 & \cdots \\ & = & 01 \cdot & 1\boxed{0 \cdot} & 10 \boxed{} \cdot & 01 \cdot & 10 \cdot & \boxed{01 \cdot} & 0\boxed{1} \cdot & 10 & \cdots \end{array} \tag{3.4}$$

The first occurrence of 010 is obtained as a suffix of $\varphi(11)$ and the second one as a prefix of $\varphi(00)$. The dots represented in the above figure are representing the *cutting bars* (of order 1) of the substitution. So, we see that for the factor 010, two kinds of configurations of the cutting bars can be achieved.

Definition 3.3.1 (Cutting set)**.** For all $k \geq 1$, we define the set $\mathrm{Cut}_k(u)$ of non-empty sets of relative positions of cutting bars

$$\mathrm{Cut}_k(u) := \left\{ \left([i, i + |u|] \cap \mathrm{bar}_k(\mathbb{N}_0)\right) - i \mid i \in \mathbb{N}_0, u = \mathbf{t}_{[i,i+|u|)} \right\}.$$

A *cutting set of order k* is an element of $\mathrm{Cut}_k(u)$. Observe that we consider the closed interval $[i, i + |u|]$ because we are also interested in knowing if the end of u coincide with a cutting bar.

To continue with our example, we have $\mathrm{Cut}_1(010) = \{\{1, 3\}, \{0, 2\}\}$, meaning that u contains two cutting bars and the first one is situated before or after the first letter. We also represent this by

$$\mathrm{Cut}_1(010) = \{0 \cdot 10 \cdot, \cdot\, 01 \cdot 0\}.$$

Remark 3.3.2. Let u be a factor of $\mathbf{t}$. Observe that, for all $\ell \geq 1$, $\mathrm{Cut}_\ell(u) \neq \emptyset$. This results from the following three observations. Obviously, $\mathrm{bar}_k(\mathbb{N}_0) \subset \mathrm{bar}_{k-1}(\mathbb{N}_0)$ and thus if $\mathrm{Cut}_k(u)$ is non-empty, then the same holds for $\mathrm{Cut}_{k-1}(u)$.

Next notice that if $\mathrm{Cut}_k(u)$ contains a singleton, then $\mathrm{Cut}_{k+1}(u)$ contains a singleton. Indeed, we can write $u = u_1u_2$ with u_1 a suffix of $\varphi^k(a)$, u_2 a prefix of $\varphi^k(b)$. Thus u_1 is a suffix of $\varphi^{k+1}(\overline{a})$ and u_2 is a prefix of $\varphi^{k+1}(b)$.

Finally, there exists a unique k such that $2^{k-1} \leq |u| \leq 2^k - 1$. There also exists i such that $u = \mathbf{t}_{[i,i+|u|)}$. Simply notice that either $[i, i+|u|] \cap \mathrm{bar}_k(\mathbb{N}_0)$ is a singleton, or $[i, i+|u|] \cap \mathrm{bar}_{k-1}(\mathbb{N}_0)$ is a singleton. The conclusion follows.

Observe that for any word u and any set $C \in \mathrm{Cut}_k(u)$, there is a unique integer $r \in [2^k - 1]_0$ such that $C \subset 2^k\mathbb{N}_0 + r$.

Lemma 3.3.3. *Let k be a positive integer, u be a factor of $\mathbf{t}$ and $C = \{i_1 < i_2 < \cdots < i_n\}$ be a set in $\mathrm{Cut}_k(u)$. There is a unique factor v of $\mathbf{t}$ of length $n-1$ such that $u = p\varphi^k(v)s$, with $|p| = i_1$. Furthermore, if $i_1 > 0$ (resp., $i_n < |u|$), there is a unique letter a such that p (resp., s) is a proper suffix (resp., prefix) of $\varphi^k(a)$.*

Proof. Since $u_{[i_1,i_n)}$ belongs to $\varphi^k(\mathcal{A}^*)$, the uniqueness of v follows from the injectivity of φ. For the uniqueness of a, this follows from the fact that $\varphi^k(0)$ and $\varphi^k(1)$ do not have any (non-empty) common prefix or suffix. □

Definition 3.3.4 (Factorization of order k)**.** Let u be a factor of $\mathbf{t}$ of length at least $2^k - 1$ and let C be a set of $\mathrm{Cut}_k(u)$. By Lemma 3.3.3, we can associate with C a unique pair

$$(p, s) \in \mathcal{A}^* \times \mathcal{A}^*$$

and a unique triple

$$(a, v, b) \in (\mathcal{A} \cup \{\varepsilon\}) \times \mathcal{A}^* \times (\mathcal{A} \cup \{\varepsilon\})$$

such that $u = p\varphi^k(v)s$, where either $a = p = \varepsilon$ (resp., $b = s = \varepsilon$), or $a \neq \varepsilon$ and p is a proper suffix of $\varphi^k(a)$ (resp., $b \neq \varepsilon$ and s is a proper prefix of $\varphi^k(b)$). In particular, we have $a = p = \varepsilon$ exactly when $\min(C) = 0$ and $b = s = \varepsilon$ exactly when $\max(C) = |u|$. The triple (a, v, b) is called the *desubstitution of u associated with C* and the pair (p, s) is called the *factorization of u associated with C*. If $C \in \mathrm{Cut}_k(u)$, then (a, v, b) and (p, s) are respectively desubstitutions and factorizations of *order k*.

If u is a factor of $\mathbf{t}$ of length at least 4, then $\mathrm{Cut}_1(u)$ contains a single set. Indeed, the factors of length 4 of $\mathbf{t}$ are

$$\{0010, 0011, 0100, 0101, 0110, 1001, 1010, 1011, 1101, 1100\}.$$

If the word 00 or 11 occurs in u, then u necessarily has a cutting bar between the two occurrences of 0 or of 1 and this cutting bar determines all the others, forcing the set $\mathrm{Cut}_1(u)$ to be a singleton. If 00 and 11 do not occur in v, then $v = 0101$ or $v = 1010$, those cases being symmetric. If $v = 0101$, then the potential cutting bars are

$$\cdot\, 01 \cdot 01 \,\cdot \qquad \text{or} \qquad 0 \cdot 10 \cdot 1.$$

However, the second case implies that the factor v occurs in $\mathbf{t}$ as a factor of $\varphi(111)$. As 111 is not a factor of $\mathbf{t}$, this shows that $\mathrm{Cut}_1(v) = \{\cdot\, 01 \cdot 01 \,\cdot\}$.

In the following statement, taking $k \geq 3$ ensures that we consider long enough words to have a unique set in $\mathrm{Cut}_1(u)$.

Lemma 3.3.5. *Let $k \geq 3$ be an integer, u be a factor of $\mathbf{t}$ of length at least $2^k - 1$. Let (a, v, b) be the desubstitution of u associated with the unique set in $\mathrm{Cut}_1(u)$ and let us write $u = p\varphi(v)s$, with p suffix of $\varphi(a)$ and s prefix of $\varphi(b)$. Let finally C be a set in $\mathrm{Cut}_k(u)$ and denote by C_p the set $(C + |p|)/2$.*

1. *If $\min C < 2^k - 1$ and $|u| - \max C < 2^k - 1$, then the set $C' = C_p$ belongs to $\mathrm{Cut}_{k-1}(avb)$;*
2. *If $\min C = 2^k - 1$ and $|u| - \max C < 2^k - 1$, then the set $C' = \{0\} \cup C_p$ belongs to $\mathrm{Cut}_{k-1}(avb)$;*
3. *If $\min C < 2^k - 1$ and $|u| - \max C = 2^k - 1$, then the set $C' = \{|avb|\} \cup C_p$ belongs to $\mathrm{Cut}_{k-1}(avb)$;*
4. *If $\min C = 2^k - 1$ and $|u| - \max C = 2^k - 1$, then the set $C' = \{0, |avb|\} \cup C_p$ belongs to $\mathrm{Cut}_{k-1}(avb)$.*

Moreover, the application from $\mathrm{Cut}_k(u)$ to $\mathrm{Cut}_{k-1}(avb)$ that maps C to C' is a bijection.

Proof. We consider the desubstitution $(a_0, a_1a_2 \cdots a_t, a_{t+1})$ associated with C. By definition, there is a unique $r \in [2^k - 1]_0$ such that $C = \{r, r + 2^k, r + 2 \cdot 2^k, \ldots, r + t \cdot 2^k\}$ and we have

$$u = \alpha\varphi^k(a_1 \cdots a_t)\beta,$$

with α the suffix of length r of $\varphi^k(a_0)$ and β the prefix of length $|u| - r - t \cdot 2^k$ of $\varphi^k(a_{t+1})$. There exist words $u_1, u_2, \ldots, u_m, u'_1, u'_2, \ldots, u'_n$ in $\varphi(\mathcal{A})$ such that

$$\alpha = pu_1u_2 \cdots u_m \quad \text{and} \quad \beta = u'_1u'_2 \cdots u'_ns.$$

Let $v_1, \ldots, v_m, v'_1, \ldots, v'_n \in \mathcal{A}$ such that $u_i = \varphi(v_i)$ and $u'_i = \varphi(v'_i)$ for all i. We have

$$avb = av_1 \cdots v_m\varphi^{k-1}(a_1 \cdots a_t)v'_1 \cdots v'_nb \tag{3.5}$$

and $av_1 \cdots v_m$ is a suffix of $\varphi^{k-1}(a_0)$ and $v'_1 \cdots v'_n b$ is a prefix of $\varphi^{k-1}(a_{t+1})$.

If $|\alpha|, |\beta| < 2^k - 1$, then

$$|av_1 \cdots v_m| = \lceil r/2 \rceil < 2^{k-1} \quad \text{and} \quad |v'_1 \cdots v_n b| < 2^{k-1}.$$

Therefore, the set $C' \in \mathrm{Cut}_{k-1}(avb)$ associated with the factorization (3.5) is

$$C' = \{\lceil r/2 \rceil, \lceil r/2 \rceil + 2^{k-1}, \dots, \lceil r/2 \rceil + t \cdot 2^{k-1}\}.$$

For the other cases, if for instance $|\alpha| = 2^k - 1$, then $|av_1 \cdots v_m| = 2^{k-1}$, which explains why we add 0 in C'.

Let us show that the correspondence between C and C' is bijective. It is trivially surjective. If there is some other cutting set $D = \{r', r' + 2^k, \dots\}$ in $\mathrm{Cut}_k(u)$, then $|r - r'| \geq 2$ since both C and D must be included in the unique set of $\mathrm{Cut}_1(u)$. This shows that the associated sets $C', D' \in \mathrm{Cut}_{k-1}(avb)$ are different. □

The substitution φ being primitive and $\mathbf{t}$ being aperiodic, Mossé's recognizability theorem ensures that the substitution φ^k is *bilaterally recognizable* [88, 89] for all $k \geq 1$, i.e., any sufficiently long factor u of $\mathbf{t}$ can be uniquely desubstituted by φ^k (up to a prefix and a suffix of bounded length). In the case of the Thue–Morse substitution, we can make this result more precise. Similar results are considered in [51] where the term (maximal extensible) *reading frames* is used. The following lemma appears in an equivalent form in [117, Proposition 2.1(4)].

Lemma 3.3.6. *Let k be a positive integer. If u is a factor of $\mathbf{t}$ of length $|u| > 3 \cdot 2^{k-1}$, then $\mathrm{Cut}_k(u)$ is a singleton.*

Proof. First observe that given a word u and a prefix v of u, a set of cutting bars for v can be extended in a unique way into a set of cutting bars for u. More precisely, if v is a prefix of u and if C belongs to $\mathrm{Cut}_k(v)$, there is a unique set C' such that $C' \in \mathrm{Cut}_k(u)$ and $C \subset C'$. It is thus enough to prove the result for words of length exactly $3 \cdot 2^{k-1} + 1$.

We proceed by induction on k. The case $k = 1$ has already been considered before Lemma 3.3.5. Let us now assume that the result is true for all $\ell \leq k$ and let us prove it for $k + 1$. If $|u| = 3 \cdot 2^k + 1$, then by the induction hypothesis, $\mathrm{Cut}_1(u)$ is a singleton and, using Lemma 3.3.3, there is a unique factor v of $\mathbf{t}$ such that

1. u is a factor of $\varphi(v)$;
2. u is not a factor of $\varphi(v')$ for any proper factor v' of v.

Since u is a factor of $\varphi(v)$, we have $|u| \leq 2|v|$ and thus, since $|v|$ is an integer, $|v| > 3 \cdot 2^{k-1}$. Using again the induction hypothesis and Lemma 3.3.3, there is a unique factor w of $\mathbf{t}$ such that

1. v is a factor of $\varphi^k(w)$;
2. v is not a factor of $\varphi^k(w')$ for any proper factor w' of w.

This word w is thus the unique factor of $\mathbf{t}$ such that

1. u is a factor of $\varphi^{k+1}(w)$;
2. u is not a factor of $\varphi^{k+1}(w')$ for any proper factor w' of w.

This shows that $\mathrm{Cut}_{k+1}(u)$ is a singleton. □

Thanks to the previous lemma, if u is of length greater than $3 \cdot 2^{k-1}$, $\mathrm{Cut}_k(u)$ is a singleton. The following lemma claims that it is not always true if $|u| \leq 3 \cdot 2^{k-1}$ but in this case, $\mathrm{Cut}_k(u)$ contains only two different cutting sets. Moreover, these two are related.

Lemma 3.3.7. *Let $k \geq 3$ be an integer and u be a factor of $\mathbf{t}$ of length $2^k - 1 \leq |u| \leq 3 \cdot 2^{k-1}$. Then $\mathrm{Cut}_k(u)$ is a not a singleton if and only if u is a factor of $\varphi^{k-1}(010)$ or of $\varphi^{k-1}(101)$, in which case $\mathrm{Cut}_k(u) = \{C_1, C_2\}$ and $|\min C_1 - \min C_2| = 2^{k-1}$. In this case, let (p_1, s_1), (p_2, s_2) be the two factorizations of order k respectively associated with $C_1, C_2 \in \mathrm{Cut}_k(u)$. Without loss of generality, assume that $|p_1| < |p_2|$. Then, there exists $a \in \mathcal{A}$ such that either*

$$|p_1| + |s_1| = |p_2| + |s_2| \text{ and } (p_2, \varphi^{k-1}(a)s_2) = (p_1\varphi^{k-1}(a), s_1),$$

or

$$||p_1| + |s_1| - (|p_2| + |s_2|)| = 2^k \text{ and } (p_2, s_2) = (p_1\varphi^{k-1}(\bar{a}), \varphi^{k-1}(a)s_1).$$

Proof. The case $k = 3$ can be checked by hand. Assume that the result holds for $k \geq 3$ and let us prove for $k+1$. Let (a, v, b) be the desubstitution of u associated with the unique set in $\mathrm{Cut}_1(u)$. By Lemma 3.3.5, we have $\#\mathrm{Cut}_{k+1}(u) = \#\mathrm{Cut}_k(avb)$ and if $\mathrm{Cut}_k(avb) = \{C_1', C_2'\}$ and $\mathrm{Cut}_{k+1}(u) = \{C_1, C_2\}$, then

$$|\min C_1 - \min C_2| = 2|\min C_1' - \min C_2'|.$$

Furthermore, u is a factor of $\varphi^k(010)$ (resp., of $\varphi^k(101)$) if and only if avb is a factor of $\varphi^{k-1}(010)$ (resp., of $\varphi^{k-1}(101)$).

For the last part of the proof, first assume that u is a factor of $\varphi^{k-1}(a\bar{a}a)$, but not a prefix nor a suffix. Since $|u| \geq 2^k - 1$, we have $u = u'\varphi^{k-1}(\bar{a})u''$, with u' and u'' respectively suffix and prefix of $\varphi^{k-1}(a)$, $|u'|, |u''| < 2^{k-1}$. Therefore, u admits the two cutting sets

$$u' \cdot \varphi^{k-1}(\bar{a})u'' \quad \text{and} \quad u'\varphi^{k-1}(\bar{a}) \cdot u''.$$

The associated factorizations are

$$(u', \varphi^{k-1}(\bar{a})u'') \quad \text{and} \quad (u'\varphi^{k-1}(\bar{a}), u'')$$

so we are in the first situation.

Assume now that u is a prefix of $\varphi^{k-1}(a\bar{a}a)$; the case where u is a suffix is similar. Two cases can occur: either $\varphi^{k-1}(a\bar{a})$ is a prefix of u, or u is a proper prefix of $\varphi^{k-1}(a\bar{a})$. If $\varphi^{k-1}(a\bar{a})$ is a prefix of u, then $u = \varphi^{k-1}(a\bar{a})u'$ for some prefix u' of $\varphi^{k-1}(a)$. If $|u'| < 2^{k-1}$, the two cutting sets of order k of u are

$$\cdot\, \varphi^k(a) \cdot u' \quad \text{and} \quad \varphi^{k-1}(a) \cdot \varphi^{k-1}(\bar{a})u'$$

and the associated factorizations are respectively

$$(\varepsilon, u') \quad \text{and} \quad (\varphi^{k-1}(a), \varphi^{k-1}(\bar{a})u').$$

We are thus in the second situation. Otherwise, $u' = \varphi^{k-1}(a)$, the two cutting sets of order k of u are

$$\cdot\, \varphi^k(a) \cdot u' \quad \text{and} \quad \varphi^{k-1}(a) \cdot \varphi^{k-1}(\bar{a})u' \cdot$$

and the associated factorizations are respectively

$$(\varepsilon, u') \quad \text{and} \quad (\varphi^{k-1}(a), \varepsilon).$$

We are in the first situation.

If u is a proper prefix of $\varphi^{k-1}(a\bar{a})$, then $u = \varphi^{k-1}(a)u'$ where u' is the prefix of length $2^{k-1} - 1$ of $\varphi^{k-1}(\bar{a})$ (because $|u| \geq 2^k - 1$). The two cutting sets of order k of u are

$$\cdot\, \varphi^{k-1}(a)u' \quad \text{and} \quad \varphi^{k-1}(a) \cdot u'$$

and the associated factorizations are respectively

$$(\varepsilon, \varphi^{k-1}(a)u') \quad \text{and} \quad (\varphi^{k-1}(a), u').$$

We are thus in the first situation again. □

3.3.2 Types associated with a factor

Remark 3.3.8. All the following constructions rely on Lemma 3.3.7. Thus, in the remaining of this chapter, we will always assume that $k \geq 3$.

Lemma 3.3.7 ensures us that whenever a word has two cutting sets, then their associated factorizations are strongly related. We will now show that whenever two factors u, v of the same length of $\mathbf{t}$ admit factorizations of order k that are similarly related, then these two words are k-binomially equivalent.

To this aim, we introduce an equivalence relation $\equiv_k$ on the set of pairs $(x, y) \in \mathcal{A}^{<2^k} \times \mathcal{A}^{<2^k}$. The core result of this section is given by Theorem 3.3.14 stating that two words are k-binomially equivalent if and only if their factorizations

of order k are equivalent for this new relation $\equiv_k$. So, the computation of $b_{\mathbf{t}}^{(k)}(n)$ amounts to determining the number of equivalence classes for $\equiv_k$ among the factorizations of order k for words in $\mathrm{Fac}_n(\mathbf{t})$.

Definition 3.3.9. Two pairs (p_1, s_1) and (p_2, s_2) of $\mathcal{A}^{<2^k} \times \mathcal{A}^{<2^k}$ are equivalent for $\equiv_k$ whenever there exists $a \in \mathcal{A}$ such that one of the following situations occurs:

1. $|p_1| + |s_1| = |p_2| + |s_2|$ and
   (a) $(p_1, s_1) = (p_2, s_2)$;
   (b) $(p_1, \varphi^{k-1}(a)s_1) = (p_2\varphi^{k-1}(a), s_2)$;
   (c) $(p_2, \varphi^{k-1}(a)s_2) = (p_1\varphi^{k-1}(a), s_1)$;
   (d) $(p_1, s_1) = (s_2, p_2) = (\varphi^{k-1}(a), \varphi^{k-1}(\bar{a}))$;
2. $\big||p_1| + |s_1| - (|p_2| + |s_2|)\big| = 2^k$ and
   (a) $(p_1, s_1) = (p_2\varphi^{k-1}(a), \varphi^{k-1}(\bar{a})s_2)$;
   (b) $(p_2, s_2) = (p_1\varphi^{k-1}(a), \varphi^{k-1}(\bar{a})s_1)$.

Remark 3.3.10. Note that if $(p_1, s_1) \equiv_k (p_2, s_2)$, then either $|p_1| = |p_2|$, or $||p_1| - |p_2|| = 2^{k-1}$. So $(p_1, s_1) \equiv_k (p_2, s_2)$ implies that $|p_1| \equiv |p_2| \pmod{2^{k-1}}$.

Example 3.3.11. Let us consider $k = 3$ and

$$u = 0101100110100110010110100 = 01\varphi^2(0)\varphi^3(01)100,$$
$$v = 0110010110100101100110100 = 01\varphi^3(11)\varphi^2(0)100.$$

From Lemma 3.3.6, they admit a unique factorization of order 3 that are respectively

$$(p_u, s_u) = (01\varphi^2(0), 100) \quad \text{and} \quad (p_v, s_v) = (01, \varphi^2(0)100).$$

By definition of $\equiv_3$, we thus have $(p_u, s_u) \equiv_3 (p_v, s_v)$.

Similarly, consider now

$$u' = 0010110100110010110100101100 = 001\varphi^3(011)0,$$
$$v' = 0010110100101100110100110010 = 001\varphi^2(0)\varphi^3(10)\varphi^2(1)0.$$

They admit a unique factorization of order 3 that are respectively

$$(p_{u'}, s_{u'}) = (001, 0) \quad \text{and} \quad (p_{v'}, s_{v'}) = (001\varphi^2(0), \varphi^2(1)0),$$

so that we again have $(p_{u'}, s_{u'}) \equiv_3 (p_{v'}, s_{v'})$.

The next result is a direct consequence of Lemma 3.3.7.

Corollary 3.3.12. *If a factor of* $\mathbf{t}$ *of length at least* $2^k - 1$ *has two distinct factorizations of order* k*, then these two are equivalent for* $\equiv_k$*.*

Definition 3.3.13 (Type of order k)**.** Given a factor u of $\mathbf{t}$ of length at least $2^k - 1$, the *type of order* k of u is the equivalence class of a factorization of order k of u. We also let (p_u, s_u) denote the factorization of order k of u for which $|p_u|$ is minimal (we assume that k is understood from the context). Therefore, two words u and v have the same type of order k if and only if

$$(p_u, s_u) \equiv_k (p_v, s_v).$$

Theorem 3.3.14. *Let* u, v *be factors of* $\mathbf{t}$ *of length* $n \geq 2^k - 1$*. We have*

$$u \sim_k v \Leftrightarrow (p_u, s_u) \equiv_k (p_v, s_v).$$

The condition is sufficient and the proof is straightforward using Proposition 1.3.6 (cancellation property) and Lemma 3.1.19. For instance, applying several times these two results, we obtain $\varphi^3(01) \sim_3 \varphi^3(11)$, thus

$$\varphi^2(0)\varphi^3(01) \sim_3 \varphi^2(0)\varphi^3(11) \sim_3 \varphi^3(11)\varphi^2(0)$$

and finally $u \sim_3 v$ for the words of Example 3.3.11.

The proof that the condition is necessary is done in Section 3.4. Preliminary to this, we consider the case of words u, v that do not have any non-empty common prefix or suffix and split the result into two lemmas: either $|p_u| \not\equiv |p_v| \pmod{2^{k-1}}$ (Lemma 3.3.15) or $|p_u| \equiv |p_v| \pmod{2^{k-1}}$ (Lemma 3.3.16). We end the section with Lemma 3.3.17 that permits us to deal with factors having some common prefix or suffix.

Lemma 3.3.15. *Let* u, v *be factors of* $\mathbf{t}$ *of length* $n \geq 2^k - 1$ *with no non-empty common prefix or suffix. If* $(p_u, s_u), (p_v, s_v)$ *satisfy* $|p_u| + |s_u| < |u|$*,* $|p_v| + |s_v| < |v|$ *and* $|p_u| \not\equiv |p_v| \pmod{2^{k-1}}$*, then* $u \nsim_k v$*.*

Proof. The assumptions $|p_u| + |s_u| < |u|$ and $|p_v| + |s_v| < |v|$ imply that there exist non-empty words z, z' such that

$$u = p_u\, \varphi^k(z)\, s_u \quad \text{and} \quad v = p_v\, \varphi^k(z')\, s_v.$$

Let $x \in \{u, v\}$. If $p_x = s_x = \varepsilon$, set $j_x := k$. Otherwise, define j_x as the largest integer such that $|x| \equiv 0 \pmod{2^{j_x - 1}}$ and $|p_x|$ or $|s_x|$ is congruent to 2^{j_x-1} modulo 2^{j_x}. In this case, such a $j_x \geq 1$ exists because p_x (resp., s_x) is a suffix (resp., prefix) of $\varphi^k(a)$ for some letter a: so it is of the form $p_x = \varphi^{i_r}(a_r) \cdots \varphi^{i_2}(a_2)\varphi^{i_1}(a_1)$ with $i_r < \cdots < i_2 < i_1 < k$ (resp., $s_x = \varphi^{i'_s}(a'_s) \cdots \varphi^{i'_2}(a'_2)\varphi^{i'_1}(a'_1)$ with $k > i'_s > \cdots > i'_2 > i'_1$). More precisely, we have

$$x = \varphi^{i_r}(a_r) \cdots \varphi^{i_2}(a_2)\varphi^{i_1}(a_1)\varphi^k(z)\varphi^{i'_s}(a'_s) \cdots \varphi^{i'_2}(a'_2)\varphi^{i'_1}(a'_1)$$

for some word z and $j_x = 1 + \min\{i_r, i'_1\}$.

Let $j = \min\{j_u, j_v\}$. Observe that $j \leq k-1$. First, since $|p_u| \not\equiv |p_v| \pmod{2^{k-1}}$, we cannot have $p_u = p_v = s_u = s_v = \varepsilon$. Moreover, proceed by contradiction and assume that $j = k$, i.e., $j_u = j_v = k$. In this case, since $|u| \equiv 0 \pmod{2^{k-1}}$, the fact that $|p_u|$ or $|s_u|$ is congruent to 2^{k-1} modulo 2^k implies that $|u|, |p_u|, |s_u|$ are all congruent to 0 modulo 2^{k-1}. The same conclusion holds for v contradicting the assumption $|p_u| \not\equiv |p_v| \pmod{2^{k-1}}$.

We will prove that $u \nsim_{j+1} v$. We have two main cases to discuss. Since u and v have the same length and $|u|, |v| \equiv 0 \pmod{2^{j-1}}$, we have either $|u| = |v| \equiv 2^{j-1} \pmod{2^j}$, or $|u| = |v| \equiv 0 \pmod{2^j}$.

The first case is split into three subcases.

1.1) Since u and v have no common prefix, we can first assume that $u = \varphi^{j-1}(0)\varphi^j(u')$ and $v = \varphi^{j-1}(1)\varphi^j(v')$ for some words u', v' (we can exchange the roles of 0 and 1). The conclusion $u \nsim_j v$ follows directly from Lemma 1.3.7 because $\varphi^{j-1}(0) \nsim_j \varphi^{j-1}(1)$ by Lemma 3.1.18.

1.2) Consider the case where $u = \varphi^{j-1}(0)\varphi^j(u')$ and $v = \varphi^j(v')\varphi^{j-1}(1)$ for some words u', v'. We can make use of Lemma 3.1.19, $v \sim_j \varphi^{j-1}(1)\varphi^j(v')$ and conclude as in the previous case.

1.3) The last subcase is when

$$u = \varphi^{j-1}(0)\varphi^j(u') = \varphi^{j-1}(0\varphi(u')) \quad \text{and} \quad v = \varphi^j(v')\varphi^{j-1}(0) = \varphi^{j-1}(\varphi(v')0)$$

(the situation with 1 instead of 0 can be treated similarly). If $j = 1$, we have $\binom{u}{01} - \binom{v}{01} = |u'| > 0$. We will assume $j > 1$. Consequently, $|u| = 2^{j-1} + 2^j|u'|$ is even, thus $|u| \geq 2^k$ and $|u'| \geq 2^{k-j} \geq 2$. From Remark 3.1.13 where multiplicities $m(x)$ are here related to $f^{j-1}(01^j)$, we get

$$\binom{u}{01^j} - \binom{v}{01^j} = \sum_{z \in f^{j-1}(01^j)} m(z) \left[\binom{0\varphi(u')}{z} - \binom{\varphi(v')0}{z} \right].$$

Recall that $f^{j-1}(01^j)$ only contains elements in $\mathcal{A}^{\leq 2}$. In the above formula, only $z = 01$ and $z = 10$ will give non-zero terms. Compute

$$\begin{aligned} \binom{0\varphi(u')}{01} - \binom{\varphi(v')0}{01} &= |u'| + \binom{\varphi(u')}{01} - \binom{\varphi(v')}{01} \\ &= |u'| + \binom{|u'|}{2} + \binom{u'}{0} - \binom{|v'|}{2} - \binom{v'}{0}. \end{aligned}$$

Hence,

$$\begin{aligned}\binom{u}{01^j} - \binom{v}{01^j} &= (m(01) - m(10))\,|u'| \\ &\quad + m(01)\left(\binom{u'}{0} - \binom{v'}{0}\right) + m(10)\left(\binom{u'}{1} - \binom{v'}{1}\right) \\ &= (m(01) - m(10))\left(|u'| + \binom{u'}{0} - \binom{v'}{0}\right).\end{aligned}$$

The last equality comes from the fact that $\binom{u'}{0} - \binom{v'}{0} = \binom{v'}{1} - \binom{u'}{1}$ because $|u'| = |v'|$.

Since u' and v' are factors of $\mathbf{t}$ of the same length, it is clear that $\binom{u'}{0} - \binom{v'}{0} \in \{-2, -1, 0, 1, 2\}$. However, in this subcase the value -2 is not realized, since v' starts with 1 (because u and v have no common prefix). Thus, by Proposition 3.1.16,

$$\binom{u}{01^j} - \binom{v}{01^j} \geq (m(01) - m(10))(|u'| - 1) > 0$$

and $u \nsim_{j+1} v$.

For the second case, we assume that $|u| = |v| \equiv 0 \pmod{2^j}$. We have four subcases for which we know that $|u'| \geq 1$.

2.1) If $u = \varphi^{j-1}(0)\varphi^j(u')\varphi^{j-1}(0)$ and $v = \varphi^j(v')$, then we know that v' is of the form $1v''$ because u and v have no common prefix. We have $u \sim_j \varphi^{j-1}(0)\varphi^{j-1}(0)\varphi^j(u')$ and $v = \varphi^{j-1}(1)\varphi^{j-1}(0)\varphi^j(v'')$ so we can directly conclude that $u \nsim_j v$ applying Lemma 1.3.7 and Lemma 3.1.18.

2.2) If $u = \varphi^{j-1}(0)\varphi^j(u')\varphi^{j-1}(1) = \varphi^{j-1}(0\varphi(u')1)$ and $v = \varphi^j(v')$, we know that v' starts with 1 and ends with 1 because u and v have no common prefix or suffix. We have

$$\begin{aligned}\binom{u}{01^j} - \binom{v}{01^j} &= m(01)\left(\binom{0\varphi(u')1}{01} - \binom{\varphi(v')}{01}\right) \\ &\quad + m(10)\left(\binom{0\varphi(u')1}{10} - \binom{\varphi(v')}{10}\right) \\ &= m(01)\left[1 + 2|u'| + \binom{u'}{0} - \binom{v'}{0} + \binom{|u'|}{2} - \binom{|v'|}{2}\right] \\ &\quad + m(10)\left[\binom{|u'|}{2} - \binom{|v'|}{2} + \binom{u'}{1} - \binom{v'}{1}\right].\end{aligned}$$

Here, $|u'| = |v'| - 1$, so $\binom{u'}{1} - \binom{v'}{1} = \binom{v'}{0} - \binom{u'}{0} - 1$, $\binom{|u'|}{2} - \binom{|v'|}{2} = -|u'|$ and we obtain

$$\binom{u}{01^j} - \binom{v}{01^j} = (m(01) - m(10))\left(1 + |u'| + \binom{u'}{0} - \binom{v'}{0}\right).$$

We need to characterize the values that can be taken by $\binom{u'}{0} - \binom{v'}{0}$. Two cases may happen: if $|u'|$ is even, there exists $\ell > 0$ such that $|u'| = 2\ell$. In this case, $|v'| = 2\ell + 1$. Since v' begins and ends with a 1, $\binom{v'}{0} = \ell$. Therefore,

$$\binom{u'}{0} - \binom{v'}{0} \in \{-1, 0, 1\}.$$

If $|u'|$ is odd, there exists ℓ such that $|u'| = 2\ell + 1$ and $|v'| = 2\ell + 2$. For the same reason as above, we cannot have $\binom{v'}{0} = \ell + 2$ and $\binom{u'}{0} - \binom{v'}{0}$ takes the same values. We thus have, in both cases, $\binom{u}{01^j} - \binom{v}{01^j} > 0$.

2.3) Now assume that $u = \varphi^{j-1}(0)\varphi^j(u')\varphi^{j-1}(0)$ and $v = \varphi^{j-1}(1)\varphi^j(v')\varphi^{j-1}(1)$. Since $|0\varphi(u')0|_0 \neq |1\varphi(v')1|_0$, when applying Remark 3.1.13 all words in $\mathcal{A}^{\leq 2}$ are contributing and we obtain

$$\begin{aligned}
\binom{u}{01^j} - \binom{v}{01^j} &= 2(m(0) - m(1)) + (m(00) - m(11))(2|u'| + 1) \\
&\quad + m(01)\left(\binom{u'}{0} - \binom{v'}{0}\right) + m(10)\left(\binom{u'}{1} - \binom{v'}{1}\right) \\
&= 2(m(0) - m(1)) \\
&\quad + (m(00) - m(11))\left(2|u'| + 1 + \binom{u'}{0} - \binom{v'}{0}\right)
\end{aligned}$$

where the last equality comes from Proposition 3.1.16. One can again conclude in the same way, making use of Proposition 3.1.17.

2.4) The last case is when $u = \varphi^{j-1}(0)\varphi^j(u')\varphi^{j-1}(1)$ and $v = \varphi^{j-1}(1)\varphi^j(v')\varphi^{j-1}(0)$. We have

$$\begin{aligned}
\binom{u}{01^j} - \binom{v}{01^j} &= m(01)\left(\binom{0\varphi(u')1}{01} - \binom{1\varphi(v')0}{01}\right) \\
&\quad + m(10)\left(\binom{0\varphi(u')1}{10} - \binom{1\varphi(v')0}{10}\right) \\
&= (m(01) - m(10))(2|u'| + 1) \\
&\quad + m(01)\left(\binom{u'}{0} - \binom{v'}{0}\right) + m(10)\left(\binom{u'}{1} - \binom{v'}{1}\right) \\
&= (m(01) - m(10))\left(2|u'| + 1 + \binom{u'}{0} - \binom{v'}{0}\right) \\
&\geq (m(01) - m(10))(2|u'| - 1) > 0
\end{aligned}$$

with the same reasoning as above.

□

Lemma 3.3.16. *Let u, v be factors of $\mathbf{t}$ of length $n \geq 2^k - 1$ with no non-empty common prefix or suffix. If $(p_u, s_u) \not\equiv_k (p_v, s_v)$ with $|p_u| \equiv |p_v| \pmod{2^{k-1}}$, then $u \nsim_k v$.*

Proof. Let ℓ (resp., ℓ') be the greatest integer less than k such that $|p_u| \equiv 0 \pmod{2^\ell}$ (resp., $|s_u| \equiv 0 \pmod{2^{\ell'}}$). The assumption $|p_u| \equiv |p_v| \pmod{2^{k-1}}$ implies that $|s_u| \equiv |s_v| \pmod{2^{k-1}}$ and thus, $|p_u| \equiv |p_v| \pmod{2^\ell}$ and $|s_u| \equiv |s_v| \pmod{2^{\ell'}}$. We have three cases to take into account.

1) If $\ell < \ell'$ (the case $\ell' < \ell$ is symmetric taking the reversals of the words), then $|s_u|$ and $|s_v|$ are even multiples of 2^ℓ, i.e., there exist $x, x' \in \mathcal{A}^*$ such that $s_u = \varphi^{\ell+1}(x)$ and $s_v = \varphi^{\ell+1}(x')$. Moreover, by maximality of ℓ, $|p_u|$ and $|p_v|$ are odd multiples of 2^ℓ, i.e., there exist $a \in \mathcal{A}$ and $y, y' \in \mathcal{A}^*$ such that

$$p_u = \varphi^\ell(a)\varphi^{\ell+1}(y), \quad p_v = \varphi^\ell(\overline{a})\varphi^{\ell+1}(y')$$

hence

$$u = p_u\varphi^{\ell+1}(z)s_u = \varphi^\ell(a)\varphi^{\ell+1}(y)\varphi^{\ell+1}(z)\varphi^{\ell+1}(x)$$

and

$$v = p_v\varphi^{\ell+1}(z')s_v = \varphi^\ell(\overline{a})\varphi^{\ell+1}(y')\varphi^{\ell+1}(z')\varphi^{\ell+1}(x')$$

for some z, z'. As usual, by Proposition 1.3.7, we can conclude because $\varphi^\ell(a) \nsim_{\ell+1} \varphi^\ell(\overline{a})$, $|yzx| = |y'z'x'|$ and $\ell + 1 \leq \ell' \leq k - 1$.

2) If $\ell = \ell' = k - 1$, we have to distinguish the cases where p_u or s_u are empty.

   – If $p_u = \varepsilon = s_u$, we have neither $p_v = \varepsilon = s_v$, nor $p_v = \varphi^{k-1}(a), s_v = \varphi^{k-1}(\overline{a})$ because u and v do not have the same type of order k. This implies that v is of the form $\varphi^{k-1}(a)\varphi^k(z')\varphi^{k-1}(a)$ and we can conclude that $u \nsim_k v$. Indeed, since $z = \overline{a}z''$ (recall that u and v have no common prefix), then

$$\begin{aligned} u = \varphi^k(z) \sim_k \varphi^{k-1}(\overline{a})\varphi^{k-1}(a)\varphi^k(z'') &\sim_k \varphi^{k-1}(\overline{a})\varphi^k(z'')\varphi^{k-1}(a) \\ &\nsim_k \varphi^{k-1}(a)\varphi^k(z')\varphi^{k-1}(a) = v. \end{aligned}$$

   – If $p_u = \varepsilon$ and $s_u = \varphi^{k-1}(a)$ (or the opposite), the fact that $(p_u, s_u) \not\equiv_k (p_v, s_v)$ gives us the possibilities $v = \varphi^{k-1}(\overline{a})\varphi^k(z')$, or $v = \varphi^k(z')\varphi^{k-1}(\overline{a})$ for some z'. But, using Proposition 1.3.7 and Lemma 3.1.19, $u \nsim_k v$.

   – If $p_u = \varphi^{k-1}(a)$ and $s_u = \varphi^{k-1}(b)$, then $u = \varphi^{k-1}(a)\varphi^k(z)\varphi^{k-1}(b)$ and $v = \varphi^{k-1}(\overline{a})\varphi^k(z')\varphi^{k-1}(\overline{b})$ for some z, z'. Moreover $a = b$ because u and v do not have the same type of order k.
   Let us assume that $a = b = 0$. Since $\varphi^{k-1}(0) \nsim_k \varphi^{k-1}(1)$, there exists a word w of length k such that

$$\binom{\varphi^{k-1}(0)}{w} \neq \binom{\varphi^{k-1}(1)}{w}.$$

Therefore, we get

$$\binom{u}{w} - \binom{v}{w} = \sum_{\substack{r,s,t\in\mathcal{A}^* \\ rst=w}} \left[\binom{\varphi^{k-1}(0)}{r}\binom{\varphi^k(z)}{s}\binom{\varphi^{k-1}(0)}{t} - \binom{\varphi^{k-1}(1)}{r}\binom{\varphi^k(z')}{s}\binom{\varphi^{k-1}(1)}{t}\right].$$

In the above sum, every term such that $|r| < k$ and $|t| < k$ vanishes because $\varphi^{k-1}(0) \sim_{k-1} \varphi^{k-1}(1)$. Hence, we get

$$\binom{u}{w} - \binom{v}{w} = 2\left[\binom{\varphi^{k-1}(0)}{w} - \binom{\varphi^{k-1}(1)}{w}\right] \neq 0.$$

3) Now assume $\ell = \ell' < k-1$. Hence, there exist letters a, b and words z, z' such that

$$u = \varphi^\ell(a)\varphi^{\ell+1}(z)\varphi^\ell(b) \quad \text{and} \quad v = \varphi^\ell(\overline{a})\varphi^{\ell+1}(z')\varphi^\ell(\overline{b}).$$

If $a = b$, then we can conclude that $u \nsim_k v$ as in the last part of case 2). Assume that $a \neq b$ (and $a = 0$, $b = 1$). Then compute (the reader should be used to this kind of computations)

$$\begin{aligned}
&\binom{u}{01^{\ell+1}} - \binom{v}{01^{\ell+1}} \\
&= \left(m_{f^\ell(01^{\ell+1})}(01) - m_{f^\ell(01^{\ell+1})}(10)\right)\left[1 + 2|z| + \binom{\varphi(z)}{01} - \binom{\varphi(z')}{01}\right] \\
&= \left(m_{f^\ell(01^{\ell+1})}(01) - m_{f^\ell(01^{\ell+1})}(10)\right)\left[1 + 2|z| + \binom{|z|}{2} + |z|_0 - \binom{|z'|}{2} - |z'|_0\right]
\end{aligned}$$

which is positive since $|z| = |z'|$.

□

When deleting common prefixes and suffixes of two factors with different types of order k, if the resulting factors are long enough, their types of order k are different.

Lemma 3.3.17. *Let u and v be factors of $\mathbf{t}$ of the same length which do not have the same type of order k. Let x (resp., y) be the longest common prefix (resp., suffix) of u and v, i.e., $u = xu'y$ and $v = xv'y$. If $|u'y| \geq 2^k - 1$ then $u'y$ and $v'y$ do not have the same type of order k. Similarly, if $|xu'| \geq 2^k - 1$ then xu' and xv' do not have the same type of order k.*

Proof. We only show the result for $u'y$ and $v'y$. Let us assume that $x \neq \varepsilon$.

Let $D \in \mathrm{Cut}_k(u)$ such that $|p_u| = \min D$. There exists $C \in \mathrm{Cut}_k(u'y)$ such that $C + |x| \subset D$. In particular, $|x| + \min C \equiv \min D \pmod{2^k}$. There exists

$C' \in \mathrm{Cut}_k(u'y)$ such that $|p_{u'y}| = \min C'$. From Lemma 3.3.7, we know that $\min C \equiv \min C' \pmod{2^{k-1}}$. Hence

$$|xp_{u'y}| = |x| + \min C' \equiv |x| + \min C \pmod{2^{k-1}}$$

and we conclude that $|xp_{u'y}| \equiv |p_u| \pmod{2^{k-1}}$. Otherwise stated,

$$|p_u| \equiv |p_v| \pmod{2^{k-1}} \quad \text{if and only if } |p_{u'y}| \equiv |p_{v'y}| \pmod{2^{k-1}}.$$

Using that fact, if $|p_u| \not\equiv |p_v| \pmod{2^{k-1}}$ then $u'y$ and $v'y$ do not have the same type of order k (see Remark 3.3.10). In what follows, we may assume that $|p_u| \equiv |p_v| \pmod{2^{k-1}}$ and thus, $|s_u| \equiv |s_v| \pmod{2^{k-1}}$.

We have two main cases to discuss: either $|p_u| = |p_v|$, or $||p_u| - |p_v|| = 2^{k-1}$.

1) Assume $|p_u| = |p_v|$. This implies that $p_u = p_v$. Indeed these two words are suffixes of the same length of a word of the form $\varphi^k(a)$ (where a is a letter). Since they share a common prefix ($x \neq \varepsilon$), they must be equal. Consequently, we have $s_u \neq s_v$, otherwise u and v would have the same type. Therefore, $y = \varepsilon$ and we will write u' instead of $u'y$. Let us show that $(p_{u'}, s_{u'}) = (\varepsilon, s_u)$ and $(p_{v'}, s_{v'}) = (\varepsilon, s_v)$ meaning that $u'y$ and $v'y$ do not have the same type. The words u and v are respectively of the form $p_u\varphi^k(u'')s_u$ and $p_v\varphi^k(v'')s_v$ for some words u'', v''.

   Since $p_u = p_v$, we have $|x| \geq |p_u|$. If $|x| = |p_u|$, $u' = \varphi^k(u'')s_u$ and $v' = \varphi^k(v'')s_v$ so $(p_{u'}, s_{u'}) = (\varepsilon, s_u)$ and $(p_{v'}, s_{v'}) = (\varepsilon, s_v)$. Otherwise, $|x| > |p_u|$ and there exists $\ell > 0$ such that[3] $p_u\varphi^k(u''_{[1,\ell-1]})$ is a proper prefix of x and such that x is a prefix of $p_u\varphi^k(u''_{[1,\ell]})$. Then $x = p_u\varphi^k(u''_{[1,\ell]})$. This is due to the fact that if $\varphi^k(a)$ and $\varphi^k(b)$ share a non-empty common prefix, then $a = b$. Thus, $u' = \varphi^k(u''_{[\ell+1,|u''|]})s_u$, $v' = \varphi^k(v''_{[\ell+1,|v''|]})s_v$ and we are done.

2) Let us consider the second case and assume that $|p_v| = |p_u| + 2^{k-1}$. As usual, u and v are of the form $p_u\varphi^k(u'')s_u$, $p_v\varphi^k(v'')s_v$. Set $u''' = \varphi(u'')$ and $v''' = \varphi(v'')$. Let a be the letter such that p_v is a suffix of $\varphi^k(a)$. Two subcases have to be considered: either $|s_v| = |s_u| + 2^{k-1}$, or $|s_u| = |s_v| + 2^{k-1}$. In both cases, we choose the letter b so that the longest word in $\{s_u, s_v\}$ is a prefix of $\varphi^k(b)$.

2.a) Consider the first subcase, $|s_v| = |s_u| + 2^{k-1}$. By definition of b, s_v is a prefix of $\varphi^k(b)$. We have $p_v = w_1\varphi^{k-1}(\overline{a})$ and $s_v = \varphi^{k-1}(b)w_2$ where w_1 (resp., w_2) is a suffix (resp., prefix) of $\varphi^{k-1}(a)$ (resp., $\varphi^{k-1}(\overline{b})$). Recall that u and v have a non-empty common prefix x. Since $|p_v| = |p_u| + 2^{k-1} < 2^k$, we get $|p_u| < 2^{k-1}$ and p_u is a suffix of $\varphi^{k-1}(a)$. Hence, $p_u = w_1$. Figure 3.2 illustrates the situation.

[3] As a finite word u'' has its first symbol indexed by 1, $u''_{[1,j]}$ is a shortcut for denoting the prefix $u''_1 \cdots u''_j$ of u''.

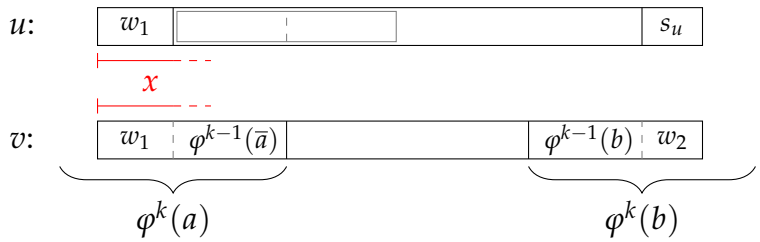


Figure 3.2: Decomposition of u and v in the first subcase.

The word s_u is a prefix of some $\varphi^k(c)$. Hence $w_2 = \overline{s_u}$ or $w_2 = s_u$ depending on whether y is empty or not. Since u and v do not have the same type, $w_2 = \overline{s_u}$ and these words are non-empty, or $\overline{a} = b$.

Using the same argument as before, $|x| \geq |p_u|$. If $|x| = |p_u|$, we have $u'y = \varphi^k(u'')s_u$ and $v'y = \varphi^{k-1}(\overline{a})\varphi^k(v'')\varphi^{k-1}(b)w_2$ and comparing the pairs (ε, s_u) and $(\varphi^{k-1}(\overline{a}), \varphi^{k-1}(b)w_2)$, we conclude that $u'y$ and $v'y$ do not have the same type (whenever $w_2 = \overline{s_u} \neq \varepsilon$, or $\overline{a} = b$).

Otherwise, $|x| > |p_u|$ and there exists some $\ell > 0$ such that

$$x = p_u\varphi^{k-1}(u'''_{[1,\ell]}) = p_u\varphi^{k-1}(\overline{a})\varphi^{k-1}(v'''_{[1,\ell-1]}). \tag{3.6}$$

We thus have

$$u'y = \varphi^{k-1}(u'''_{[\ell+1,|u'''|]})s_u \quad \text{and} \quad v'y = \varphi^{k-1}(v'''_{[\ell,|v'''|]})s_v.$$

From equalities (3.6), we observe that

$$u'''_i = \begin{cases} \overline{a}, & \text{if } i = 1; \\ v'''_{i-1}, & \text{if } 1 < i \leq \ell. \end{cases} \tag{3.7}$$

Moreover, since $u'''_{2i+1}u'''_{2i+2} = \varphi(u''_{i+1})$, for all $i \in [|u''|]$, we have $u'''_{2i+1} = \overline{u'''_{2i+2}}$. Similarly, we have $v'''_{2i+1} = \overline{v'''_{2i+2}}$. We may thus conclude that

$$u'''_i = \begin{cases} \overline{a}, & \text{if } i \text{ is odd;} \\ a, & \text{if } i \text{ is even.} \end{cases} \tag{3.8}$$

If ℓ is even, we have

$$u'y = \varphi^k(u''_{[\frac{\ell+2}{2},|u''|]})s_u \quad \text{and} \quad v'y = \varphi^{k-1}(v'''_\ell)\varphi^k(v''_{[\frac{\ell+2}{2},|v''|]})\varphi^{k-1}(b)w_2.$$

Thus, $(p_{u'y}, s_{u'y}) = (\varepsilon, s_u)$ and $(p_{v'y}, s_{v'y}) = (\varphi^{k-1}(\overline{a}), \varphi^{k-1}(b)w_2)$ and $u'y$, $v'y$ do not have the same type of order k.

If ℓ is an odd number,

$$u'y = \varphi^{k-1}(u'''_{\ell+1})\varphi^k(u''_{[\frac{\ell+3}{2},|u''|]})s_u \quad \text{and} \quad v'y = \varphi^k(v''_{[\frac{\ell+1}{2},|v''|]})\varphi^{k-1}(b)w_2.$$

In this case, $(p_{u'y}, s_{u'y}) = (\varphi^{k-1}(a), s_u)$ and $(p_{v'y}, s_{v'y}) = (\varepsilon, \varphi^{k-1}(b)w_2)$ and again, $u'y$, $v'y$ do not have the same type of order k.

2.b) Let us care about the second subcase: $|s_u| = |s_v| + 2^{k-1}$. By definition of b, s_u is a prefix of $\varphi^k(b)$. Thus $p_v = w_1\varphi^{k-1}(\overline{a})$ and $s_u = \varphi^{k-1}(b)w_2$ where w_1 (resp., w_2) is a suffix (resp., prefix) of $\varphi^{k-1}(a)$ (resp., $\varphi^{k-1}(\overline{b})$). Otherwise stated, as illustrated in Figure 3.3, we have

$$u = p_u\varphi^k(u'')\varphi^{k-1}(b)w_2 \quad \text{and} \quad v = p_u\varphi^{k-1}(\overline{a})\varphi^k(v'')s_v.$$

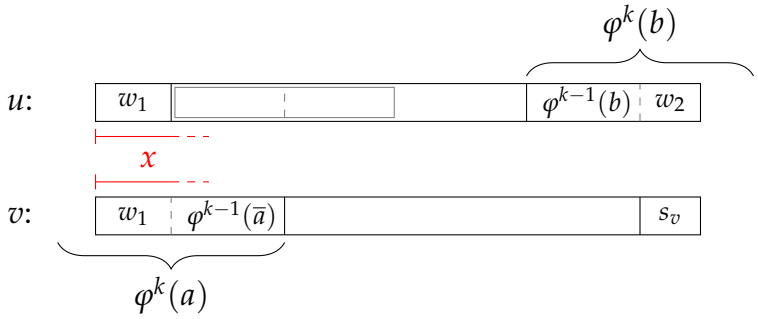


Figure 3.3: Decomposition of u and v in the second subcase.

Observe again that $w_2 = s_v$ or $w_2 = \overline{s_v}$.

Because u and v do not have the same type of order k, $w_2 = \overline{s_v}$ and these words are non-empty, or $a = b$.

If $x = p_u$, $(p_{u'y}, s_{u'y}) = (\varepsilon, \varphi^{k-1}(b)w_2)$ and $(p_{v'y}, s_{v'y}) = (\varphi^{k-1}(\overline{a}), s_v)$. Comparing these two pairs, we get $(p_{u'y}, s_{u'y}) \not\equiv_k (p_{v'y}, s_{v'y})$, i.e., $u'y$ and $v'y$ do not have the same type.

Otherwise, as in the previous case, there exists $\ell > 0$ such that $x = p_u\varphi^{k-1}(u'''_{[1,\ell]})$. Observe that Equalities (3.7) and (3.8) are still valid. One can proceed as in the previous case discussing the parity of ℓ to conclude.

□

3.4 k-binomial complexity of the Thue–Morse word

Using the lemmas from the previous section, we first show that two distinct factors of $\mathbf{t}$ of length at most $2^k - 1$ are never k-binomially equivalent. Then, we take into account factors of length at least 2^k. On the one hand, we prove that $(p_u, s_u) \not\equiv_k (p_v, s_v)$ implies $u \not\sim_k v$. On the other hand, we compute the number of equivalence classes of $\#\left(\{(p_u, s_u) : u \in \mathrm{Fac}_n(\mathbf{t})\}/\equiv_k\right)$.

Proposition 3.4.1. *Let u, v be two different factors of $\mathbf{t}$ of length $n \leq 2^k - 1$, which do not have any common prefix or suffix. We have $u \not\sim_k v$.*

Proof. If $n \leq 3$, this is trivial: take u, v two different factors of length n, $\binom{u}{u} = 1$ and $\binom{v}{u} = 0$. Since $k \geq 3$, $u \not\sim_k v$. Let us assume $n \geq 4$ and set $j = \max\{i \leq k : |u| \geq 2^{i+1}\}$. Notice that $1 \leq j \leq k-2$. The type of order j of u and v is well defined. Either they have the same type of order j, or they do not. If they do not have the same type, since we always have $|p_u| + |s_u| \leq 2^{j+1} - 2$, we have $|p_u| + |s_u| < |u|$. The same holds for v. By applying Lemmas 3.3.15 or 3.3.16, we obtain that $u \not\sim_j v$, thus $u \not\sim_k v$.

We can thus assume that $(p_u, s_u) \equiv_j (p_v, s_v)$. Let us consider the different cases of Definition 3.3.9. Since u and v do not share any common prefix or suffix, this gives restrictions to the different possibilities. For instance, in the situation (1.a) of the definition, we get $(p_u, s_u) = (p_v, s_v) = (\varepsilon, \varepsilon)$. In what follows, let a, b be two distinct letters.

(1.a) If $(p_u, s_u) = (p_v, s_v)$, then $p_u = p_v = s_u = s_v = \varepsilon$, so $u = \varphi^j(u')$ and $v = \varphi^j(v')$ for some words u' and v' of length 2 or 3 (by definition of j). Since u and v do not have any non-empty common prefix and suffix, u' and v' have distinct first and last letter. Recalling that $\mathbf{t}$ is cube-free, we thus have to consider the cases

$$(u', v') \in \{(aa, bb), (aab, bba), (aba, bab), (ab, ba), (aab, baa) \mid a, b \in \mathcal{A}, a \neq b\}.$$

Using the same kind of computations as before, e.g., in the proof of Lemma 3.3.15 making use of Remark 3.1.13, we get

$$\begin{aligned}\binom{u}{ab^j} - \binom{v}{ab^j} &= \binom{\varphi^j(u')}{ab^j} - \binom{\varphi^j(v')}{ab^j} \\ &= \left(m_{f^{j-1}(ab^j)}(ab) - m_{f^{j-1}(ab^j)}(ba)\right)\left(|u'|_a - |v'|_a\right)\end{aligned}$$

which is positive for the first three pairs (u', v'). For the last two ones, simply compute

$$\begin{aligned}\binom{u}{ab^{j+1}} - \binom{v}{ab^{j+1}} = m_{f^j(ab^{j+1})}(ab)&\left(\binom{u'}{ab} - \binom{v'}{ab}\right) \\ - m_{f^j(ab^{j+1})}(ba)&\left(\binom{u'}{ba} - \binom{v'}{ba}\right) > 0.\end{aligned}$$

Since $j \leq k-2$, this implies in all cases that $u \nsim_k v$.

(1.b) and (1.c) Let us consider the case where there exists some letter a such that $u = \varphi^{j-1}(a)\varphi^j(u')$ and $v = \varphi^j(v')\varphi^{j-1}(a)$ (or the converse). We know that the first letter of v' is b and the last letter of u' is a (because u and v have distinct first and last letter). Since $\mathbf{t}$ is overlap-free, we have to consider the cases

$$\begin{aligned}(u', v') \in \{&(aa, ba), (aa, bb), (ba, ba), (ba, bb),\\ &(aba, bab), (aba, bba), (baa, bab), (baa, bba) \mid a, b \in \mathcal{A}, a \neq b\}.\end{aligned}$$

Indeed, u' cannot be equal to bba because otherwise u would be equal to $\varphi^{j-1}(ababaab)$. Similarly, v' cannot be equal to baa. For every pair in the above set, compute

$$\begin{aligned}\binom{\varphi^{j-1}(a\varphi(u'))}{ab^j} - \binom{\varphi^{j-1}(\varphi(v')a)}{ab^j}&\\ = (m_{f^{j-1}(ab^j)}(ab) - m_{f^{j-1}(ab^j)}(ba))(|u'| + |u'|_a - |v'|_a) > 0.&\end{aligned}$$

(1.d) Assume now that there exists a letter a such that $u = \varphi^{j-1}(a)\varphi^j(u')\varphi^{j-1}(b)$ and $v = \varphi^{j-1}(b)\varphi^j(v')\varphi^{j-1}(a)$. We have $|u| = 2^j + 2^j|u'|$ and from the definition of j, we get $|u'| \leq 2$. Then compute

$$\begin{aligned}\binom{\varphi^{j-1}(a\varphi(u')b)}{ab^j} - \binom{\varphi^{j-1}(b\varphi(v')a)}{ab^j}&\\ = (m_{f^{j-1}(ab^j)}(ab) - m_{f^{j-1}(ab^j)}(ba))(1 + 2|u'| + |u'|_a - |v'|_a).&\end{aligned}$$

This quantity is positive for every $u', v' \in \mathcal{A} \cup \mathcal{A}^2$.

(2.a) and (2.b) Otherwise, there exists some letter a such that $u = \varphi^{j-1}(a)\varphi^j(u')\varphi^{j-1}(b)$ and $v = \varphi^j(v')$. Again $|u'| \leq 2$ and $|v'| = |u'| + 1$. Then, v' has to begin with a and end with b. The cases to consider are the following ones:

$$\begin{aligned}(u', v') \in \{&(a, ab), (b, ab), (aa, aab), (aa, abb),\\ &(ab, aab), (ab, abb), (ba, aab), (ba, abb) \mid a, b \in \mathcal{A}, a \neq b\}.\end{aligned}$$

Reasoning as in subcase 2.2) of Lemma 3.3.15,

$$\begin{aligned}\binom{\varphi^{j-1}(a\varphi(u')b)}{ab^j} - \binom{\varphi^{j-1}(\varphi(v'))}{ab^j}&\\ = (m_{f^{j-1}(ab^j)}(ab) - m_{f^{j-1}(ab^j)}(ba))(1 + |u'| + |u'|_a - |v'|_a) > 0.&\end{aligned}$$

□

Corollary 3.4.2. *Let $k \geq 3$. For all $n \leq 2^k - 1$, we have $b_{\mathbf{t}}^{(k)}(n) = p_{\mathbf{t}}(n)$.*

Proof. Let us take two different factors u and v of the same length $n \leq 2^k - 1$. If u and v do not share any common prefix or suffix, $u \nsim_k v$ by the previous proposition. Otherwise, there exist words x, y, u', v' such that $u = xu'y$, $v = xv'y$ where u' and v' do not share any common prefix or suffix. We apply the previous proposition to u', v' and conclude using the cancellation property (Proposition 1.3.6). □

Let u, v be distinct factors of $\mathbf{t}$ of length $n \geq 2^k - 1$. We are now ready to prove that $(p_u, s_u) \not\equiv_k (p_v, s_v)$ implies $u \nsim_k v$. We thus recall Theorem 3.3.14 that we are going to prove.

Theorem 3.3.14. *Let u, v be factors of $\mathbf{t}$ of length $n \geq 2^k - 1$. We have*

$$u \sim_k v \Leftrightarrow (p_u, s_u) \equiv_k (p_v, s_v).$$

Proof. Recall that we already discussed the sufficiency of the result. Let us prove that it is necessary. Let x and y respectively denote the longest common prefix and suffix of u and v: $u = xu'y$ and $v = xv'y$. We obviously have $u' \neq v'$ and, by Proposition 1.3.6, we have $u \sim_k v$ if and only if $u' \sim_k v'$.

If $|u'| \leq 2^k - 1$, using Proposition 3.4.1, we conclude that $u' \nsim_k v'$. Otherwise, from Lemma 3.3.17, u' and v' do not have the same type of order k. Thus, without loss of generality we may now assume that u and v do not have any non-empty common prefix or suffix.

Let j be the greatest integer less than or equal to k such that $|u|, |v| \geq 2^{j+1}$. If u and v do not have the same type of order j, then we fall into one of the complementary situations of Lemmas 3.3.15 or 3.3.16 (indeed, the extra assumption of Lemmas 3.3.15 holds because $|u| \geq 2^{j+1}$, $|p_u|, |s_u| \leq 2^j - 1$ and thus $(p_u, s_u), (p_v, s_v)$ satisfy $|p_u| + |s_u| < |u|$, $|p_v| + |s_v| < |v|$). We thus have $u \nsim_j v$ and then $u \nsim_k v$.

Otherwise u and v have the same type of order j. By assumption, they do not have the same type of order k, hence $j < k$. One has to do the same proof as the one of Proposition 3.4.1, except that one more argument is needed. In case (1.a) and if $(u', v') \in \{(ab, ba), (aab, baa)\}$, we compute $\binom{u}{ab^{j+1}} - \binom{v}{ab^{j+1}}$. We need to stress the fact that in this particular case, $j < k - 1$. Indeed, $j = k - 1$ would give $u = \varphi^{k-1}(ab)$, $v = \varphi^{k-1}(ba)$ or $u = \varphi^{k-1}(a)\varphi^k(a)$, $v = \varphi^k(b)\varphi^{k-1}(a)$. In both cases, this is impossible since u and v do not have the same type of order k. □

Due to Theorem 3.3.14, the k-binomial complexity of $\mathbf{t}$ can be computed from

$$b_{\mathbf{t}}^{(k)}(n) = \#\left(\mathrm{Fac}_n(\mathbf{t}) / \sim_k\right) = \#\left(\{(p_u, s_u) : u \in \mathrm{Fac}_n(\mathbf{t})\} / \equiv_k\right).$$

The last theorem provides this quantity.

Theorem 3.4.3. *For all $k \geq 3$, $n \geq 2^k$, we have*

$$\#\left(\{(p_u, s_u) : u \in \mathrm{Fac}_n(\mathbf{t})\}/\equiv_k\right) = \begin{cases} 3\cdot 2^k - 3, & \text{if } n \equiv 0 \pmod{2^k}; \\ 3\cdot 2^k - 4, & \text{otherwise.} \end{cases}$$

Proof. Let $n \geq 2^k$ and set $\lambda \in [2^k - 1]_0$ the integer such that $n \equiv \lambda \pmod{2^k}$.

For every $\ell \in [2^{k-1} - 1]_0$,

$$P_\ell = \{(p_u, s_u) : u \in \mathrm{Fac}_n(\mathbf{t}), |p_u| = \ell \text{ or } |p_u| = 2^{k-1} + \ell\}$$

and

$$S_\ell = \{(p_u, s_u) : u \in \mathrm{Fac}_n(\mathbf{t}), |s_u| = \ell \text{ or } |s_u| = 2^{k-1} + \ell\}.$$

If ℓ and ℓ' are two distinct elements from $\{0, \ldots, 2^{k-1} - 1\}$ then, due to Definition 3.3.9, for all $(p_u, s_u) \in P_\ell, (p_v, s_v) \in P_{\ell'}$, we have $(p_u, s_u) \not\equiv_k (p_v, s_v)$. The idea of the proof is to count the number of equivalence classes in the pairwise disjoint sets P_ℓ.

We can notice that $|p_u| + |s_u| \in \{\lambda, \lambda + 2^k\}$ for all factorizations (p_u, s_u). From this, note that $P_0 = S_0$ if and only if $\lambda = 0$ or $\lambda = 2^{k-1}$. In this case, we set $\ell_0 = 0$ and thus $P_{\ell_0} = S_0$. Otherwise, there exists $\ell_0 \neq 0$ such that $P_{\ell_0} = S_0$.

We will show that

$$\#\left((P_0 \cup P_{\ell_0})/\equiv_k\right) = \#\left((P_0 \cup S_0)/\equiv_k\right) = \begin{cases} 3, & \text{if } \lambda = 0; \\ 2, & \text{if } \lambda = 2^{k-1}; \\ 8, & \text{otherwise,} \end{cases}$$

and that, for every $\ell \in [2^{k-1} - 1]_0 \setminus \{0, \ell_0\}$,

$$\#\left(P_\ell/\equiv_k\right) = 6.$$

Obviously, we have

$$\#[2^{k-1} - 1]_0 \setminus \{0, \ell_0\} = \begin{cases} 2^{k-1} - 1, & \text{if } \lambda = 0 \text{ or } \lambda = 2^{k-1}; \\ 2^{k-1} - 2, & \text{otherwise.} \end{cases}$$

Putting together the previous observations, we therefore get

$$\#\left(\{(p_u, s_u) : u \in \mathrm{Fac}_n(\mathbf{t})\}/\equiv_k\right) = \#\bigcup_{\ell=0}^{2^{k-1}-1} P_\ell = \begin{cases} 6\left(2^{k-1} - 1\right) + 3, & \text{if } \lambda = 0; \\ 6\left(2^{k-1} - 1\right) + 2, & \text{if } \lambda = 2^{k-1}; \\ 6\left(2^{k-1} - 2\right) + 8, & \text{otherwise,} \end{cases}$$

which gives us the expected result.

First, let us deal with P_0 and S_0. If $\lambda = 0$, due to Proposition 3.2.1 (ensuring that every pair of P_0 appears as a couple (p_u, s_u) for a word u which is a factor of $\mathbf{t}$, and this argument is repeated all along the proof), we have $P_0 = S_0$ which is equal to

$$\{(\varepsilon, \varepsilon), (\varphi^{k-1}(0), \varphi^{k-1}(0)), (\varphi^{k-1}(0), \varphi^{k-1}(1)), (\varphi^{k-1}(1), \varphi^{k-1}(0)), (\varphi^{k-1}(1), \varphi^{k-1}(1))\}.$$

By Definition 3.3.9, $\#(P_0/\equiv_k) = 3$. If $\lambda = 2^{k-1}$,

$$P_0 = S_0 = \{(\varepsilon, \varphi^{k-1}(0)), (\varphi^{k-1}(0), \varepsilon), (\varepsilon, \varphi^{k-1}(1)), (\varphi^{k-1}(1), \varepsilon)\}$$

and $\#(P_0/\equiv_k) = 2$. Finally, two subcases have to be distinguished if $\lambda \notin \{0, 2^{k-1}\}$: either $0 < \lambda < 2^{k-1}$ or $2^{k-1} < \lambda < 2^k$. Let y be the prefix of $\varphi^k(0)$ of length λ.

In the first subcase, y is also a prefix of $\varphi^{k-1}(0)$. We thus have

$$\begin{aligned} P_0 = \{&(\varepsilon, y), (\varepsilon, \overline{y}), (\varphi^{k-1}(0), \varphi^{k-1}(1)y), (\varphi^{k-1}(1), \varphi^{k-1}(1)y), \\ &(\varphi^{k-1}(0), \varphi^{k-1}(0)\overline{y}), (\varphi^{k-1}(1), \varphi^{k-1}(0)\overline{y})\} \end{aligned}$$

and $\#(P_0/\equiv_k) = 4$. We can proceed in the same way for S_0 and get a total of 8 classes.

In the second subcase, we can write $y = \varphi^{k-1}(0)z$ where z is the prefix of $\varphi^{k-1}(1)$ of length $\lambda - 2^{k-1}$. We have

$$P_0 = \{(\varepsilon, \varphi^{k-1}(0)z), (\varepsilon, \varphi^{k-1}(1)\overline{z}), (\varphi^{k-1}(0), z), (\varphi^{k-1}(1), z), (\varphi^{k-1}(0), \overline{z}), (\varphi^{k-1}(1), \overline{z})\}$$

and once again, $\#(P_0/\equiv_k) = 4$. The same result holds for S_0.

Let us now consider $\ell \in \{0, \ldots, 2^{k-1} - 1\} \setminus \{0, \ell_0\}$ and show that $\#(P_\ell/\equiv_k) = 6$. Two cases have to be considered: either $\lambda < \ell$, or $\lambda > \ell$. Indeed, we cannot have $\lambda = \ell$. Observe that if $\lambda = \ell$ or $\lambda = \ell + 2^{k-1}$, then $S_0 = P_\ell$ which means that $\ell_0 = \ell$ but we are assuming that $\ell \notin \{0, \ell_0\}$. Recall that $|p_u| + |s_u| \in \{\lambda, \lambda + 2^k\}$ for all factorizations (p_u, s_u). We will make constant use of this fact.

a) If $\lambda < \ell$, we cannot have $|p_u| + |s_u| = \lambda$, so obviously $|p_u| + |s_u| = 2^k + \lambda$ for all $(p_u, s_u) \in P_\ell$. Therefore, if $|p_u| = \ell < 2^{k-1}$, then $|s_u| > 2^{k-1}$. On the other hand, if $|p_u| = \ell + 2^{k-1}$, then $|s_u| < 2^{k-1}$. Set x (resp., y) the suffix (resp., prefix) of $\varphi^{k-1}(0)$ of length ℓ (resp., $\lambda + 2^k - \ell$). We thus have

$$\begin{aligned} P_\ell = \{&(x, \varphi^{k-1}(0)\overline{y}), (x, \varphi^{k-1}(1)y), (\overline{x}, \varphi^{k-1}(0)\overline{y}), (\overline{x}, \varphi^{k-1}(1)y), \\ &(x\varphi^{k-1}(1), y), (x\varphi^{k-1}(1), \overline{y}), (\overline{x}\varphi^{k-1}(0), y), (\overline{x}\varphi^{k-1}(0), \overline{y})\} \end{aligned}$$

and, from Definition 3.3.9, $\#(P_\ell/\equiv_k) = 6$.

b) If $\ell < \lambda$, observe that since $|p_u| = \ell$, $|s_u| = \lambda - \ell$. Indeed, since $|s_u| < 2^k$, $|p_u| + |s_u| < \ell + 2^k < \lambda + 2^k$, hence we have $|p_u| + |s_u| = \lambda$. Two subcases have to be considered: $\lambda - \ell < 2^{k-1}$, or $\lambda - \ell > 2^{k-1}$.

b.1) In the first subcase, $|p_u| = \ell$, thus $|s_u| = \lambda - \ell < 2^{k-1}$. Otherwise stated, if p_u is a suffix of some $\varphi^{k-1}(a)$, then s_u is a prefix of some $\varphi^{k-1}(b)$. Moreover, $\ell > \lambda - 2^{k-1}$ ensures that $|p_u| = \ell + 2^{k-1}$ and so $|p_u| + |s_u| = \lambda + 2^k$. Therefore, if $|p_u| = \ell + 2^{k-1}$, then $|s_u| > 2^{k-1}$. Otherwise stated, if p_u has a suffix of the form $\varphi^{k-1}(a)$, then s_u has a prefix of the form $\varphi^{k-1}(b)$.

b.2) In the second subcase, $\ell + 2^{k-1} < \lambda$ implies that $|p_u| = \ell + 2^{k-1}$ and thus $|p_u| + |s_u| = \lambda$. This is why, if $|p_u| = \ell + 2^{k-1}$, then $|s_u| < 2^{k-1}$. Otherwise stated, if p_u has a suffix of the form $\varphi^{k-1}(a)$, then s_u is a prefix of some $\varphi^{k-1}(b)$. Finally, we already know that if $|p_u| = \ell$, then $|s_u| = \lambda - \ell$ which is here greater than 2^{k-1}. Otherwise stated, if p_u is a suffix of some $\varphi^{k-1}(a)$, then s_u has a prefix of the form $\varphi^{k-1}(b)$.

Let us denote by x the suffix of $\varphi^{k-1}(0)$ of length ℓ and y the prefix of $\varphi^{k-1}(0)$, whose length is $\lambda - \ell$ in the first subcase, $\lambda - \ell - 2^{k-1}$ in the second one. The case b.1) gives us

$$\begin{aligned} P_\ell = \{&(x,y),(\overline{x},y),(x,\overline{y}),(\overline{x},\overline{y}),(x\varphi^{k-1}(1),\varphi^{k-1}(1)y),\\ &(x\varphi^{k-1}(1),\varphi^{k-1}(0)\overline{y}),(\overline{x}\varphi^{k-1}(0),\varphi^{k-1}(1)y),(\overline{x}\varphi^{k-1}(0),\varphi^{k-1}(0)\overline{y})\} \end{aligned}$$

while the case b.2) gives

$$\begin{aligned} P_\ell = \{&(x,\varphi^{k-1}(1)y),(\overline{x},\varphi^{k-1}(1)y),(x,\varphi^{k-1}(0)\overline{y}),(\overline{x},\varphi^{k-1}(0)\overline{y}),\\ &(x\varphi^{k-1}(1),y),(x\varphi^{k-1}(1),\overline{y}),(\overline{x}\varphi^{k-1}(0),y),(\overline{x}\varphi^{k-1}(0),\overline{y})\}. \end{aligned}$$

Both of them lead to the conclusion that $\#\left(P_\ell/\equiv_k\right) = 6$.

□

As a consequence of Corollary 3.4.2, Theorem 3.3.14 and Theorem 3.4.3, we get the expected result.

Theorem 3.4.4. *Let k be a positive integer. For all $n \leq 2^k - 1$, we have*

$$b_{\mathbf{t}}^{(k)}(n) = p_{\mathbf{t}}(n).$$

For all $n \geq 2^k$, we have

$$b_{\mathbf{t}}^{(k)}(n) = \begin{cases} 3\cdot 2^k - 3, & \text{if } n \equiv 0 \pmod{2^k};\\ 3\cdot 2^k - 4, & \text{otherwise.} \end{cases}$$

3.5 Possible generalizations

The Thue–Morse word is part of a larger family: it is a Parikh-constant morphism. A morphism σ defined over the alphabet $\mathcal{A}$ is *Parikh-constant* if all the images $\sigma(a)$, $a \in \mathcal{A}$, are Abelian equivalent (recall Definition 1.3.1). In [109] it is shown that every fixed point of a Parikh-constant morphism has a k-binomial complexity that is bounded by a constant depending on k. Therefore, one can ask if it is possible to compute, as in Theorem 3.4.4, the exact value of $b_{\mathbf{x}}^{(k)}(n)$ for any such word $\mathbf{x}$. However,

the techniques developed here and more specially in Sections 3.2 to 3.4 are devoted to the Thue–Morse word and cannot be extended in a straightforward manner. Some computer experiments were carried on generalizations of the Thue–Morse word over an arbitrary alphabet (see [26] or [96] for studies of these words), fixed points of morphisms

$$\varphi_m : \{0,1,\dots,m-1\}^* \to \{0,1,\dots,m-1\}^* : i \mapsto i(i+1)\cdots(m-1)01\cdots(i-1),$$

but the expression of a formula describing the k-binomial complexity seems to be more intricate. Let $\mathbf{t_3} = \lim_{n\to+\infty} \varphi_3(0)$. By computer experiments it seems that $b^{(2)}_{\mathbf{t_3}}(n) = p_{\mathbf{t_3}}(n)$ for $n < 3^2$ and, for $n \geq 3^2$,

$$b^{(2)}_{\mathbf{t_3}}(n) = \begin{cases} 49, & \text{if } n \equiv 0 \pmod{3^2}; \\ 48, & \text{if } n \equiv 3^1 \text{ or } 2\cdot 3^1 \pmod{3^2}; \\ 45, & \text{otherwise.} \end{cases}$$

Similarly, it seems that $b^{(3)}_{\mathbf{t_3}}(n) = p_{\mathbf{t_3}}(n)$ for $n < 3^3$ and, for $n \geq 3^3$,

$$b^{(3)}_{\mathbf{t_3}}(n) = \begin{cases} 175, & \text{if } n \equiv 0 \pmod{3^3}; \\ 174, & \text{if } n \equiv 3^2 \text{ or } 2\cdot 3^2 \pmod{3^3}; \\ 171, & \text{otherwise.} \end{cases}$$

Then, considering $\mathbf{t_4} = \lim_{n\to+\infty} \varphi_4(0)$, we get with a computer that $b^{(2)}_{\mathbf{t_4}}$ takes different values if $n \equiv 0 \pmod{4^2}$, if $n \equiv 2\cdot 4^1 \pmod{4^2}$, if $n \equiv 4^1$ or $3\cdot 4^1 \pmod{4^2}$ or if none of these equivalences hold.

So we thought that a pattern arises: $b^{(k)}_{\mathbf{t_m}}$ can take a finite number of values, among which there are special values if $n \equiv i\cdot m^{k-1} \pmod{m^k}$, $i \in [m-1]_0$, and a constant value in all other cases. But this reasoning felt apart with $\mathbf{t_5} = \lim_{n\to+\infty} \varphi_5(0)$ for which we copy the first 50 values we got for $b^{(2)}_{\mathbf{t_5}}(n)$:

1, 5, 16, 28, 40, 52, 68, 84, 94, 104, 116, 128, 132, 136, 148, 160, 173, 160, 148, 136,
132, 128, 116, 104, 94, 160, 160, 160, 168, 160, 160, 160, 173, 160, 160, 160, 168, 160,
160, 160, 170, 160, 160, 160, 168, 160, 160, 160, 173, 160

It seems that the pattern appearing depends on multiples of 4 and not multiples of 5. Therefore, a sharp description of the constants related to a given Parikh-constant morphism appears to be challenging, since we cannot succeed even for the generalized Thue–Morse case. In particular, when the distinct symbols occurring in $\sigma(a)$ do not all have the same frequency, the problem is clearly open.

4 | The Tribonacci word

In the previous chapter, we investigated the k-binomial complexity of the Thue–Morse word, following the interesting result of Proposition 1.4.5. This chapter will get interest into the other result from [109] stated here as Proposition 1.4.4. We know that every Sturmian word has a k-binomial complexity equal to its factor complexity, for $k \geq 2$. We thus decided to study the well-known Tribonacci word $\mathbf{T}$, which can be seen as a generalization to a ternary alphabet of a Sturmian word. Up to our knowledge, $\mathbf{T}$ does not seem to be a fixed point of a Parikh-constant morphism. Computer experimentations show that we likely have $b_{\mathbf{T}}^{(k)} = p_{\mathbf{T}}$ for any $k \geq 2$, as in the Sturmian case. The results presented in this chapter thus have the aim to prove this last affirmation.

The Tribonacci word $\mathbf{T}$ is the fixed point, starting by 0, of the following morphism:

$$\tau : \{0,1,2\}^* \to \{0,1,2\}^* : \begin{cases} 0 \mapsto 01; \\ 1 \mapsto 02; \\ 2 \mapsto 0. \end{cases}$$

It has the following property: for each $n \in \mathbb{N}_0$, there exist exactly one right special factor and one left special factor of length n (it means that for each $n \in \mathbb{N}_0$, there exist exactly one $\ell_n \in \mathrm{Fac}_n(\mathbf{T})$, exactly one $r_n \in \mathrm{Fac}_n(\mathbf{T})$, and at least four letters $a_\ell, b_\ell, a_r, b_r \in \mathcal{A}$ such that $a_\ell \neq b_\ell$, $a_\ell\ell_n, b_\ell\ell_n \in \mathrm{Fac}_{n+1}(\mathbf{T})$, $a_r \neq b_r$ and $r_n a_r, r_n b_r \in \mathrm{Fac}_{n+1}(\mathbf{T})$). Moreover, these factors are extendable by the three letters of the alphabet, that is, $0\ell_n, 1\ell_n, 2\ell_n \in \mathrm{Fac}_{n+1}(\mathbf{T})$ and $r_n 0, r_n 1, r_n 2 \in \mathrm{Fac}_{n+1}(\mathbf{T})$. Hence, it is easy to deduce the factor complexity of $\mathbf{T}$: for any $n \in \mathbb{N}_0$,

$$p_{\mathbf{T}}(n) = 2n + 1.$$

The main result of this chapter is that $b_{\mathbf{T}}^{(k)}(n) = p_{\mathbf{T}}(n)$ for every $k \geq 2$ and every n. As for the Sturmian case, it thus suffices to show that two different factors of $\mathbf{T}$ are never 2-binomially equivalent. Surprisingly, classical combinatorial techniques seemed to be unsuccessful. We make an extensive use of the concepts of *templates* and their ancestors, similarly to what can be found in [1, 2, 31] where avoidance of Abelian

repetitions is considered. Most of the results are coming from [68] by Marie Lejeune, Michel Rigo and Matthieu Rosenfeld. I presented this work in *WORDS* 2019. An extended version containing all the details [71] is published in Advances in Applied Mathematics. The chapter is organized as follows: we recall in the first section the notion of Kronecker product, and we define extended Parikh vectors $\underline{\mathbf{\Phi}}$ in such a way that $u \sim_2 v$ if and only if $\underline{\mathbf{\Phi}}(u) = \underline{\mathbf{\Phi}}(v)$. In Section 4.2 we define and adapt the notions of templates and ancestors to our purpose. To solve our problem, we need to show the finiteness of some set of realizable ancestors. To that end, we first get in Section 4.3 several bounds related to Parikh vectors of factors of the Tribonacci word. Consequently, we deduce bounds on the realizable ancestors. We put together the results of these last two sections to establish the main theorem in Section 4.4. Similarly to [1, 2, 31, 74, 103], our proof is a computer-assisted one. We give and comment the `Mathematica` code in Appendix A. We finish the chapter by some possible further extensions.

Contents

4.1 The Kronecker product

Definition 4.1.1. Let u be a finite word over $\{0, \ldots, m-1\}$. We will make an extensive use of its *extended Parikh vector* denoted by $\underline{\mathbf{\Phi}}(u)$ and defined as follows. We set

$$\underline{\mathbf{\Phi}}(u) := \left(|u|_0, \ldots, |u|_{m-1}, \binom{u}{00}, \binom{u}{01}, \ldots, \binom{u}{(m-1)(m-1)} \right)^{\mathsf{T}};$$

it is a column vector of size $m(m+1)$ and we assume that the m^2 subwords of length 2 are lexicographically ordered.

Take the word $u = 10010201010$ which is a factor of length 11 occurring in the Tribonacci word. Its extended Parikh vector is given by

$$\underline{\mathbf{\Phi}}(u) = \Big(6, 4, 1, 15, 11, 3, 13, 6, 2, 3, 2, 0\Big)^{\mathsf{T}}.$$

Clearly, some of these entries are redundant, e.g., for all $a \in \mathcal{A}$, $\binom{w}{aa} = \binom{|w|_a}{2}$ where on the right-hand side we have a classical binomial coefficient. With this notation, $\underline{\mathbf{\Phi}}(u) = \underline{\mathbf{\Phi}}(v)$ if and only if $u \sim_2 v$.

For a vector $\underline{\mathbf{d}} \in \mathbb{Z}^n$, $n \geq m$, we let $\underline{\mathbf{d}}|_m$ denote the vector in $\mathbb{Z}^m$ made of the first m coordinates of $\underline{\mathbf{d}}$. Recall that $\underline{\mathbf{\Psi}}$ is the classical Parikh vector; over an alphabet of size m, $\underline{\mathbf{\Phi}}(u)|_m = \underline{\mathbf{\Psi}}(u)$.

We let $A \otimes B$ denote the usual Kronecker product of two matrices $A = (a_{jk})_{\substack{1 \leq j \leq m \\ 1 \leq k \leq n}} \in \mathbb{Z}^{m \times n}$ and $B \in \mathbb{Z}^{p \times q}$. It is the block-matrix in $\mathbb{Z}^{mp \times nq}$ defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}.$$

Let us recall some well-known properties of the Kronecker product [49].

Lemma 4.1.2. *For all matrices A, B, C, D, and for every scalar ℓ, as long as the involved products and sums of matrices are defined, the following equalities hold:*

- *(mixed-product property)* $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$;
- *(left-linearity)* $(A + B) \otimes D = (A \otimes D) + (B \otimes D)$;
- *(right-linearity)* $A \otimes (C + D) = (A \otimes C) + (A \otimes D)$;
- *(associativity)* $\ell(A \otimes B) = (\ell A \otimes B) = (A \otimes \ell B)$;
- *(determinant)* $\det(A \otimes B) = \det(A)^m \det(B)^n$ *when A and B are square matrices of size n and m respectively.*

Let ℓ be an integer. We let $P_\ell \in \mathbb{Z}^{(\ell(\ell+1)) \times \ell^2}$ denote the matrix such that for all i, j,

$$[P_\ell]_{i,j} = \begin{cases} 1, & \text{if } i = j + \ell; \\ 0, & \text{otherwise.} \end{cases}$$

This matrix, when applied on a column vector of size ℓ^2, adds ℓ zeros at the beginning of this vector. We start with two straightforward lemmas and the introduction of an extended Parikh matrix.

Lemma 4.1.3. *Let u and v be two words over an alphabet of size m. We have*

$$\underline{\mathbf{\Phi}}(uv) = \underline{\mathbf{\Phi}}(u) + \underline{\mathbf{\Phi}}(v) + P_m\big(\underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(v)\big).$$

Proof. The first two terms in the statement take into account the separate contributions of u and v to the different coefficients. Nevertheless, subwords of length 2 can also be obtained by taking their first letter in u and their second one in v. This is exactly the contribution of the third term. Observe that $\underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(v)$ is a column vector of size m^2. Applying P_m will add m zeros on top because the contribution of individual letters has already been taken into account in the first two terms. □

The classical *Parikh matrix* M'_σ associated with a morphism σ is a useful tool in combinatorics on words (not to be confused with the notion of Parikh matrix of a word introduced in 2000 by Mateescu et al. [83]). Over an ordered m-letter alphabet $\{0, \ldots, m-1\}$, it is defined from its columns as a $m \times m$ matrix

$$M'_\sigma = \begin{pmatrix} \underline{\mathbf{\Psi}}(\sigma(0)) & \cdots & \underline{\mathbf{\Psi}}(\sigma(m-1)) \end{pmatrix}$$

and it readily satisfies

$$\underline{\mathbf{\Psi}}(\sigma(u)) = M'_\sigma \underline{\mathbf{\Psi}}(u), \quad \forall u \in \mathcal{A}^*.$$

For the Tribonacci morphism, it is given by

$$M'_\tau = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{4.1}$$

Definition 4.1.4. Mimicking the Parikh matrix and its use, one can define an extended Parikh matrix M_σ associated with a morphism σ defined over an ordered m-letter alphabet. It is a $m(m+1) \times m(m+1)$ matrix satisfying

$$\underline{\mathbf{\Phi}}(\sigma(u)) = M_\sigma \underline{\mathbf{\Phi}}(u), \quad \forall u \in \mathcal{A}^*. \tag{4.2}$$

The existence of the extended Parikh matrix satisfying (4.2) is ensured by the next result.

Lemma 4.1.5. *Let M'_σ be the Parikh matrix associated with some morphism σ. The extended Parikh matrix of σ has the following form:*

$$M_\sigma = \begin{pmatrix} M'_\sigma & 0 & \cdots & 0 \\ \star & & & \\ \star & & M'_\sigma \otimes M'_\sigma & \\ \star & & & \end{pmatrix},$$

where $\star$ represents elements we don't need to determine for our result. In particular, if the alphabet is of size m, then $\det(M_\sigma) = \det(M'_\sigma)^{2m+1}$. Moreover, $M_\sigma P_m = P_m(M'_\sigma \otimes M'_\sigma)$. If M'_σ is non-singular, then M_σ is non-singular and M_σ^{-1} is a block-triangular matrix of the same form as M_σ with diagonal blocks $M_\sigma'^{-1}$ and $M_\sigma'^{-1} \otimes M_\sigma'^{-1}$.

Proof. Since the first m components of $\underline{\mathbf{\Phi}}(u)$ give the usual Parikh vector, M'_σ is the upper-left corner of M_σ. For the last m^2 components of $\underline{\mathbf{\Phi}}(u)$ dealing with binomial coefficients of subwords of length 2, it is shown in [67] that, for all $a, b \in \mathcal{A}$,

$$\binom{\sigma(u)}{ab} = \sum_{c\in\mathcal{A}} \binom{\sigma(c)}{ab} |u|_c + \sum_{x_1x_2\in\mathcal{A}^2} \binom{\sigma(x_1)}{a}\binom{\sigma(x_2)}{b}\binom{u}{x_1x_2}.$$

In this expression, the first sum corresponds to the $m \times m$ submatrices marked as $\star$ and the second sum exactly corresponds to the Kronecker product $M'_\sigma \otimes M'_\sigma$. Indeed, if we index M'_σ on $\mathcal{A}$ and $M'_\sigma \otimes M'_\sigma$ on $\mathcal{A}^2$, we have

$$\binom{\sigma(x_1)}{a}\binom{\sigma(x_2)}{b} = \left[M'_\sigma\right]_{a,x_1} \left[M'_\sigma\right]_{b,x_2} = \left[M'_\sigma \otimes M'_\sigma\right]_{ab,x_1x_2}.$$

□

This extended Parikh matrix was also used in [102] (for avoidance problems). We will use several times this combinatorial lemma.

Lemma 4.1.6. *Let $n, m \in \mathbb{N}^+$, $a \in \mathbb{C}$, $c, v_1, \ldots, v_m \in \mathbb{C}^n$ and $\mathcal{C}$ be the convex hull of $\{v_1, \ldots, v_m\}$. Then*

$$\max_{y\in\mathcal{C}} |a + c \cdot y| = \max_{y\in\{v_1,\ldots,v_m\}} |a + c \cdot y|.$$

Proof. Since the convex hull is a compact set and the application $y \mapsto |a + c \cdot y|$ is continuous on $\mathcal{C}$, the maximum is well defined and realized in a point x of $\mathcal{C}$. Formally, we thus have $|a + c \cdot x| = \max_{y\in\mathcal{C}} |a + c \cdot y|$. By definition, for any $x \in \mathcal{C}$, there exist $\alpha_1, \ldots, \alpha_m$ in the set $[0, 1]$ such that $1 = \sum_{i=1}^m \alpha_i$ and $x = \sum_{i=1}^m \alpha_i v_i$. Then, using linearity and triangular inequality, we get

$$\begin{aligned}
\max_{y\in\mathcal{C}} |a + c \cdot y| &= |a + c \cdot x| \\
&= \left|a + c \cdot \sum_{i=1}^m \alpha_i v_i\right| \\
&= \left|\sum_{i=1}^m (\alpha_i (a + c \cdot v_i))\right| \\
&\leq \sum_{i=1}^m \alpha_i |a + c \cdot v_i| \\
&\leq \sum_{i=1}^m \alpha_i \max_{y\in\{v_1,\ldots,v_m\}} |a + c \cdot y| \\
\max_{y\in\mathcal{C}} |a + c \cdot y| &\leq \max_{y\in\{v_1,\ldots,v_m\}} |a + c \cdot y|.
\end{aligned}$$

Since $\{v_1, \ldots, v_m\} \subseteq \mathcal{C}$, $\max_{y\in\mathcal{C}} |a + c \cdot y| = \max_{y\in\{v_1,\ldots,v_m\}} |a + c \cdot y|$ which concludes the proof. □

4.2 Templates and ancestors

For this section, let $\sigma : \mathcal{A}^* \to \mathcal{A}^*$ be any primitive (prolongable) morphism. Let M'_σ be its Parikh matrix and M_σ be its extended Parikh matrix. We let $m := \#\mathcal{A}$ and $\mathcal{L}(\sigma)$ be the *language of* σ, that is, the set of factors of any of its non-empty fixed points (if σ is primitive, they all have the same language). For the Tribonacci morphism τ, we have $\mathcal{L}(\tau) = \mathrm{Fac}(\mathbf{T})$ and $\mathrm{PPref}(\tau) = \{\varepsilon, 0\}$, $\mathrm{PSuff}(\tau) = \{\varepsilon, 1, 2\}$. Such a notation can be extended to σ^n. If $u \in \mathcal{L}(\sigma)$, there exist a shortest $p_u \in \mathrm{PPref}(\sigma)$, a shortest $s_u \in \mathrm{PSuff}(\sigma)$ and $u' \in \mathcal{L}(\sigma)$ such that $\sigma(u') = p_u u s_u$.

In the following definition, the index **b** (resp., **e**) stands for beginning (resp., end).

Definition 4.2.1. A *template* is a 5-tuple of the form $t = [\underline{\mathbf{d}}, \underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}, a_1, a_2]$ where $a_1, a_2 \in \mathcal{A}$, $\underline{\mathbf{d}} \in \mathbb{Z}^{m(m+1)}$ and $\underline{\mathbf{D_b}}, \underline{\mathbf{D_e}} \in \mathbb{Z}^m$. A pair of words (u, v) is a *realization* of (or *realizes*) the template t if

- $\underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v) = \underline{\mathbf{d}} + P_m\big(\underline{\mathbf{D_b}} \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{D_e}}\big)$, and;
- there exist u' and v' such that $u = u'a_1$ and $v = v'a_2$.

A template t is *realizable by* σ if there is a pair of words in $\mathcal{L}(\sigma)$ that realize t.

Given two factors u and v, the template of the form $[\underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v), \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2]$ is obviously realizable by σ, where a_1 (resp., a_2) is the last letter of u (resp., v).

Due to the presence of P_m in the above definition, note that if a template is realizable by a pair (u, v), then the corresponding vector $\underline{\mathbf{d}}$ is such that $\underline{\mathbf{d}}|_m = \underline{\mathbf{\Psi}}(u) - \underline{\mathbf{\Psi}}(v)$.

Example 4.2.2. Take $u = 00102010$ and $v = 10201020$ two factors of the Tribonacci word. We have

$$\underline{\mathbf{\Phi}}(u) = (5,\ 2,\ 1,\ 10,\ 6,\ 3,\ 4,\ 1,\ 1,\ 2,\ 1,\ 0)^\mathsf{T}$$

$$\underline{\mathbf{\Phi}}(v) = (4,\ 2,\ 2,\ 6,\ 2,\ 4,\ 6,\ 1,\ 3,\ 4,\ 1,\ 1)^\mathsf{T}$$

and the difference is given by

$$\underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v) = (1,\ 0,\ -1,\ 4,\ 4,\ -1,\ -2,\ 0,\ -2,\ -2,\ 0,\ -1)^\mathsf{T}.$$

Taking

$$\underline{\mathbf{d}} = (1,\ 0,\ -1,\ -1,\ -1,\ 4,\ 1,\ 0,\ 1,\ -3,\ -1,\ 0)^\mathsf{T}$$

and

$$\underline{\mathbf{D_b}} = (0,\ -1,\ 0)^\mathsf{T} \text{ and } \underline{\mathbf{D_e}} = (1,\ 1,\ -1)^\mathsf{T}$$

leads to

$$\underline{\mathbf{D_b}} \otimes \underline{\mathbf{\Psi}}(u) = (0,\ -1,\ 0)^\mathsf{T} \otimes (5,\ 2,\ 1)^\mathsf{T} = (0,\ 0,\ 0,\ -5,\ -2,\ -1,\ 0,\ 0,\ 0)^\mathsf{T}$$

and

$$\underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{D_e}} = (5,\ 2,\ 1)^\mathsf{T} \otimes (1,\ 1,\ -1)^\mathsf{T} = (5,\ 5,\ -5,\ 2,\ 2,\ -2,\ 1,\ 1,\ -1)^\mathsf{T}.$$

This shows that we have a template $[\underline{\mathbf{d}}, \underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}, 0, 0]$ realizable by τ. Actually, for any alphabet of size m, for any choice of words u, v and vectors $\underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}$ in $\mathbb{Z}^m$, there exists a convenient $\underline{\mathbf{d}} \in \mathbb{Z}^{m(m+1)}$.

Lemma 4.2.3. *Let σ be a primitive morphism. Let*

$$\mathcal{T} := \{[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2] : a_1 \neq a_2\}.$$

The factor complexity and the 2-binomial complexity of any fixed point of σ are equal if and only if all templates from $\mathcal{T}$ are non-realizable by σ.

Proof. The factor complexity is not the same as the 2-binomial complexity if and only if there exists a pair of factors (u, v) such that $u \neq v$ and $\underline{\mathbf{\Phi}}(u) = \underline{\mathbf{\Phi}}(v)$.

The two words of any realization of an element in $\mathcal{T}$ are 2-binomially equivalent and are different since they do not have the same last letter. Thus, if there is a realization of an element of $\mathcal{T}$, then the factor complexity and the 2-binomial complexity are not equal.

Now, for the other direction, suppose that the two complexity functions are not equal: we have a pair of words (u, v) such that $u \neq v$ and $\underline{\mathbf{\Phi}}(u) = \underline{\mathbf{\Phi}}(v)$. Since $u \neq v$ and $|u| = |v|$, there exist $u', v', s \in \mathcal{A}^*$ and $a, b \in \mathcal{A}$ with $a \neq b$ such that $u = u'as$ and $v = v'bs$ (observe that s is the longest common suffix of u and v). Then $\underline{\mathbf{\Phi}}(u'a) = \underline{\mathbf{\Phi}}(v'b)$ so the pair $(u'a, v'b)$ realizes $[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a, b]$, which belongs to $\mathcal{T}$. □

The idea in the next definition is that any long factor of a fixed point of a morphism must be the image of a shorter factor, up to (short) prefix and suffix. So the relation corresponds to the various relationships among the binomial coefficients that must hold if this is to be the case.

Definition 4.2.4. Let $t' = [\underline{\mathbf{d}'}, \underline{\mathbf{D}'_\mathbf{b}}, \underline{\mathbf{D}'_\mathbf{e}}, a'_1, a'_2]$ and $t = [\underline{\mathbf{d}}, \underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}, a_1, a_2]$ be two templates and σ be a morphism. We say that t' *is a parent by σ of t* if there exist $p_u, p_v \in \mathrm{PPref}(\sigma)$ and $s_u, s_v \in \mathrm{PSuff}(\sigma)$ such that

- $\underline{\mathbf{d}'}$ is given by

$$\begin{aligned} M_\sigma \underline{\mathbf{d}'} = \underline{\mathbf{d}} &+ \underline{\mathbf{\Phi}}(p_u s_u) - \underline{\mathbf{\Phi}}(p_v s_v) + P_m\big(\underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{d}}|_m + \underline{\mathbf{d}}|_m \otimes \underline{\mathbf{\Psi}}(s_v)\big) \\ &- P_m\big((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(p_u s_u) + \underline{\mathbf{\Psi}}(p_u s_u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))\big); \end{aligned}$$

- the value of $\underline{\mathbf{D}'_{\mathbf{b}}}$ is given by $M'_\sigma \underline{\mathbf{D}'_{\mathbf{b}}} = \underline{\mathbf{D}_{\mathbf{b}}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)$;
- the value of $\underline{\mathbf{D}'_{\mathbf{e}}}$ is given by $M'_\sigma \underline{\mathbf{D}'_{\mathbf{e}}} = \underline{\mathbf{D}_{\mathbf{e}}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v)$;
- $a_1 s_u$ is a suffix of $\sigma(a'_1)$;
- $a_2 s_v$ is a suffix of $\sigma(a'_2)$.

We let $\mathrm{Par}_\sigma(t)$ denote the set of parents by σ of t.

Remark 4.2.5. Observe that for any given template t, $\mathrm{Par}_\sigma(t)$ is finite and easy to compute as long as M_σ and M'_σ are non-singular. Indeed, the sets $\mathrm{PPref}(\sigma)$ and $\mathrm{PSuff}(\sigma)$ are finite. For the Tribonacci word, $\#\,\mathrm{PPref}(\tau) = 2$, $\#\,\mathrm{PSuff}(\tau) = 3$, hence $\#\,\mathrm{Par}_\tau(t) \leq 36$. At this stage, it is not required for a parent to be realizable.

More interestingly there is a link between preimages of the realization by σ of a template and realization by σ of the parents of the template. We make that link explicit in the following lemma.

Lemma 4.2.6. *Let σ be a morphism. Assume that $\det(M'_\sigma) = \pm 1$. Let t be a template, $u, v, v', u' \in \mathcal{L}(\sigma)$, $p_u, p_v \in \mathrm{PPref}(\sigma)$ and $s_u, s_v \in \mathrm{PSuff}(\sigma)$ such that*

- *$\sigma(u') = p_u u s_u$ and $\sigma(v') = p_v v s_v$;*
- *s_u is a proper suffix of the image of the last letter of u';*
- *s_v is a proper suffix of the image of the last letter of v';*
- *(u, v) realizes t.*

Then there exists a parent t' of t such that (u', v') realizes t'.

Proof. Let $t = [\underline{\mathbf{d}}, \underline{\mathbf{D}_{\mathbf{b}}}, \underline{\mathbf{D}_{\mathbf{e}}}, a_1, a_2]$ be a template realized by (u, v). Let us compute $M_\sigma(\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v'))$. We first make use of (4.2):

$$\begin{aligned} M_\sigma(\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v')) &= \underline{\mathbf{\Phi}}(\sigma(u')) - \underline{\mathbf{\Phi}}(\sigma(v')) \\ &= \underline{\mathbf{\Phi}}(p_u u s_u) - \underline{\mathbf{\Phi}}(p_v v s_v) \\ &= \underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v) + \underline{\mathbf{\Phi}}(p_u s_u) - \underline{\mathbf{\Phi}}(p_v s_v) \\ &\quad + P_m\big(\underline{\mathbf{\Psi}}(p_u) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s_u)\big) \\ &\quad - P_m\big(\underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{\Psi}}(v) + \underline{\mathbf{\Psi}}(v) \otimes \underline{\mathbf{\Psi}}(s_v)\big). \end{aligned}$$

The last line comes from the fact that, for all $a, b \in \mathcal{A}$,

$$\binom{p_u u s_u}{ab} = \binom{p_u s_u}{ab} + \binom{u}{ab} + |p_u|_a\,|u|_b + |u|_a\,|s_u|_b.$$

Now we can use the equality $\underline{\mathbf{\Psi}}(v) = \underline{\mathbf{\Psi}}(u) - \underline{\mathbf{d}}|_m$ and the definition of a template to express $\underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v)$ from t. The vector $M_\sigma(\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v'))$ is equal to

$$\begin{aligned}
&\underline{\mathbf{d}} + P_m(\underline{\mathbf{D_b}} \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{D_e}}) + \underline{\mathbf{\Phi}}(p_u s_u) - \underline{\mathbf{\Phi}}(p_v s_v) \\
&\quad + P_m\big(\underline{\mathbf{\Psi}}(p_u) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s_u)\big) \\
&\quad - P_m\big(\underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s_v) - \underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{d}}|_m - \underline{\mathbf{d}}|_m \otimes \underline{\mathbf{\Psi}}(s_v)\big).
\end{aligned}$$

Rearranging the terms gives us a new expression of $M_\sigma(\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v'))$:

$$\begin{aligned}
&\underline{\mathbf{d}} + \underline{\mathbf{\Phi}}(p_u s_u) - \underline{\mathbf{\Phi}}(p_v s_v) + P_m\big(\underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{d}}|_m + \underline{\mathbf{d}}|_m \otimes \underline{\mathbf{\Psi}}(s_v)\big) \\
&\quad + P_m\big((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))\big) \\
&= \underline{\mathbf{d}} + \underline{\mathbf{\Phi}}(p_u s_u) - \underline{\mathbf{\Phi}}(p_v s_v) + P_s\big(\underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{d}}|_m + \underline{\mathbf{d}}|_m \otimes \underline{\mathbf{\Psi}}(s_v)\big) \\
&\quad - P_m\big((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(p_u s_u) + \underline{\mathbf{\Psi}}(p_u s_u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))\big) \\
&\quad + P_m\big((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(p_u u s_u) + \underline{\mathbf{\Psi}}(p_u u s_u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))\big)
\end{aligned}$$

where for the last line, we simply recall that $\underline{\mathbf{\Psi}}(u) = \underline{\mathbf{\Psi}}(p_u u s_u) - \underline{\mathbf{\Psi}}(p_u s_u)$. Since $det(M'_\sigma) = \pm 1$, we also have $\det(M_\sigma) = \pm 1$ and thus the following vector has integer entries

$$\begin{aligned}
\underline{\mathbf{d}'} := M_\sigma^{-1}\Big(&\underline{\mathbf{d}} + \underline{\mathbf{\Phi}}(p_u s_u) - \underline{\mathbf{\Phi}}(p_v s_v) + P_m\big(\underline{\mathbf{\Psi}}(p_v) \otimes \underline{\mathbf{d}}|_m + \underline{\mathbf{d}}|_m \otimes \underline{\mathbf{\Psi}}(s_v)\big) \\
&- P_m\big((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(p_u s_u) + \underline{\mathbf{\Psi}}(p_u s_u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))\big)\Big).
\end{aligned}$$

Then, recalling the form of M_σ given in Lemma 4.1.5 and classical properties of Kronecker product (Lemma 4.1.2), we get

$$\begin{aligned}
\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v') &= \underline{\mathbf{d}'} + M_\sigma^{-1} P_m\big((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(p_u u s_u) \\
&\quad + \underline{\mathbf{\Psi}}(p_u u s_u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))\big) \\
&= \underline{\mathbf{d}'} + P_m\big((M_\sigma'^{-1} \otimes M_\sigma'^{-1})((\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v)) \otimes \underline{\mathbf{\Psi}}(p_u u s_u)) \\
&\quad + (M_\sigma'^{-1} \otimes M_\sigma'^{-1})(\underline{\mathbf{\Psi}}(p_u u s_u) \otimes (\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v)))\big) \\
&= \underline{\mathbf{d}'} + P_m((M_\sigma'^{-1}(\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v))) \otimes (M_\sigma'^{-1}\underline{\mathbf{\Psi}}(p_u u s_u)) \\
&\quad + (M_\sigma'^{-1}\underline{\mathbf{\Psi}}(p_u u s_u)) \otimes (M_\sigma'^{-1}(\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v))) \\
&= \underline{\mathbf{d}'} + P_m\big((M_\sigma'^{-1}(\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v))) \otimes \underline{\mathbf{\Psi}}(u') \\
&\quad + \underline{\mathbf{\Psi}}(u') \otimes (M_\sigma'^{-1}(\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v)))\big)
\end{aligned}$$

where on the last line, we make use of (4.2). Finally, if we let

$$\underline{\mathbf{D'_b}} := M_\sigma'^{-1}(\underline{\mathbf{D_b}} + \underline{\mathbf{\Psi}}(p_u) - \underline{\mathbf{\Psi}}(p_v))$$

and

$$\underline{\mathbf{D'_e}} := M_\sigma'^{-1}(\underline{\mathbf{D_e}} + \underline{\mathbf{\Psi}}(s_u) - \underline{\mathbf{\Psi}}(s_v)),$$

we get

$$\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v') = \underline{\mathbf{d}'} + P_m\big(\underline{\mathbf{D}'_{\mathbf{b}}} \otimes \underline{\mathbf{\Psi}}(u') + \underline{\mathbf{\Psi}}(u') \otimes \underline{\mathbf{D}'_{\mathbf{e}}}\big). \tag{4.3}$$

Let a'_1 be the last letter of u', and a'_2 be the last letter of v'. Since s_u is a proper suffix of $\sigma(a'_1)$, $a_1 s_u$ is a suffix of $\sigma(a'_1)$. Similarly $a_2 s_v$ is a suffix of $\sigma(a'_2)$.

From the definitions of realizable template and parent, it is clear that the template $t' = [\underline{\mathbf{d}'}, \underline{\mathbf{D}'_{\mathbf{b}}}, \underline{\mathbf{D}'_{\mathbf{e}}}, a'_1, a'_2]$ is a parent of t and is realized by (u', v'). □

This motivates the following definition.

Definition 4.2.7. A template t' is an *ancestor by σ* (resp., *realizable ancestor*) of a template t if there exists a sequence of $n \geq 1$ templates (resp., realizable templates) $t_1, t_2, \ldots, t_n$ with $t = t_1$ and $t_n = t'$, such that for all $i \in [n-1]$, t_{i+1} is a parent by σ of t_i. For a template t, we denote by $\mathrm{RAnc}_\sigma(t)$ the set of all the realizable ancestors by σ of t. We may omit "by σ" when the morphism is clear from the context.

Example 4.2.8. Take the words $u = 00102010$ and $v = 10201020$ and the template of the previous example. Let $u' = 20100$ and $v' = 01010$ be two factors of the Tribonacci word. We have

$$\tau(u') = u1,\ \tau(v') = 0v1$$

and we set $p_u = \varepsilon$, $s_u = 1$, $p_v = 0$ and $s_v = 1$. Using Definition 4.2.4, we compute

$$\underline{\mathbf{D}'_{\mathbf{b}}} = (-1,\ 0,\ 0)^{\mathsf{T}} \quad \text{and} \quad \underline{\mathbf{D}'_{\mathbf{e}}} = (1,\ -1,\ 1)^{\mathsf{T}}.$$

Hence, we get

$$\underline{\mathbf{d}'} = (0,\ -1,\ 1,\ 0,\ -2,\ 0,\ -1,\ -1,\ 0,\ 3,\ 1,\ 0)^{\mathsf{T}}$$

and by Lemma 4.2.6, this parent template is a template realized by (u', v'). It is a realizable ancestor. Applying (4.3) leads to

$$\underline{\mathbf{\Phi}}(u') - \underline{\mathbf{\Phi}}(v') = (0,\ -1,\ 1,\ 0,\ -2,\ 0,\ -1,\ -1,\ 0,\ 3,\ 1,\ 0)^{\mathsf{T}}.$$

Since 2010 is a right-special factor of the Tribonacci word, we could also have chosen $u' = 20101$ (or 20102). In that case, this changes s_u to 2 (or ε) and thus leads to other vectors $\underline{\mathbf{D}'_{\mathbf{b}}}, \underline{\mathbf{D}_{\mathbf{e}}}', \underline{\mathbf{d}'}$ and other parent templates realizable by τ.

Let $|\sigma| = \max_{a \in \mathcal{A}} |\sigma(a)|$, usually called the *width* of σ.

Proposition 4.2.9. *Let L be a positive integer. Let σ be a primitive morphism and t_0 be a template. If there exists a pair of words in $\mathcal{L}(\sigma)$ that is a realization of t_0, then*

- *either t_0 has a realization $(u, v) \in \mathcal{L}(\sigma) \times \mathcal{L}(\sigma)$ such that $\min(|u|, |v|) \leq L$, or;*
- *there exists a realization $(u, v) \in \mathcal{L}(\sigma) \times \mathcal{L}(\sigma)$ of a template t of $\mathrm{RAnc}_\sigma(t_0)$ with $L \leq \min(|u|, |v|) \leq |\sigma| L$.*

Proof. Let (u, v) be a pair of factors of $\mathcal{L}(\sigma)$ realizing t_0. If $\min(|u|, |v|) \leq L$, there is nothing left to prove. Assume therefore that $\min(|u|, |v|) > L$.

Since v is a factor of $\mathcal{L}(\sigma)$, there are sequences of words $v = v_1, v_2, \ldots, v_n \in \mathcal{A}^*$, $p_1, \ldots, p_{n-1} \in \mathrm{PPref}(\sigma)$ and $s_1, \ldots, s_{n-1} \in \mathrm{PSuff}(\sigma)$ such that, for all $i < n$, $\sigma(v_{i+1}) = p_i v_i s_i$ and $L \leq |v_n| \leq |\sigma|L$. Moreover we may force that, for all $i < n$, s_i is a proper suffix of the image of the last letter of v_{i+1}.

Similarly, since u is a factor of $\mathcal{L}(\sigma)$, there are sequences of words $u = u_1$, $u_2, \ldots, u_\ell \in \mathcal{A}^*$, $p'_1, \ldots, p'_{\ell-1} \in \mathrm{PPref}(\sigma)$ and $s'_1, \ldots, s'_{\ell-1} \in \mathrm{PSuff}(\sigma)$ such that, for all $i < \ell$, $\sigma(u_{i+1}) = p'_i u_i s'_i$ and $L \leq |u_\ell| \leq L|\sigma|$.

Let $m = \min(n, \ell)$. We can simply apply Lemma 4.2.6 inductively m times. We obtain a template t' which is an ancestor of t_0 and is realized by (u_m, v_m). Since $m = \min(n, \ell)$ we obtain that $L \leq \min(|u_m|, |v_m|) \leq |\sigma|L$. This concludes the proof. □

4.3 Bounding realizable templates for the Tribonacci word

The Parikh matrix M'_τ for the Tribonacci morphism was given in (4.1). Since it is primitive, we may use Perron–Frobenius theorem which asserts in particular that a real square matrix with positive entries has a unique largest real eigenvalue (called the *Perron eigenvalue*) and that the corresponding eigenvector can be chosen to have strictly positive components [98, 47]. Let $\theta \approx 1.839$ be the Perron eigenvalue of M_τ; it is the dominant root of the polynomial $x^3 - x^2 - x - 1$. Let $\mathbf{w}$ be an infinite word over $\mathcal{A}$ and $a \in a$. The *density of a in* $\mathbf{w}$ is defined as the limit

$$\lim_{n \to +\infty} \frac{|\mathbf{w}_1 \cdots \mathbf{w}_n|_a}{n},$$

if it exists. Densities of letters $0, 1, 2$ in $\mathbf{T}$ exist and are denoted respectively by α_0, α_1 and α_2. Moreover, $\underline{\boldsymbol{\alpha}} = \begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 \end{pmatrix}^\mathsf{T}$ is an eigenvector of τ associated with θ. A proof of these results can be found in [77].

4.3.1 Bounds on extended Parikh vectors

From [106, Prop. 2.4], we know that, for every factor u of $\mathbf{T}$,

$$\big||u|_i - \alpha_i |u|\big| < 1.5, \quad i \in \{0, 1, 2\}.$$

Let $\Delta = \{(\delta_0, \delta_1) : -1.5 \leq \delta_0, \delta_1, \delta_0 + \delta_1 \leq 1.5\}$ be the set represented in Figure 4.1. We will use this result in the following form.

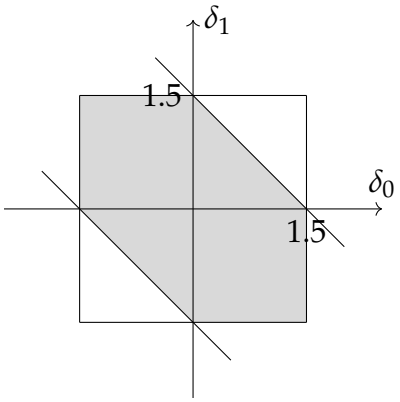


Figure 4.1: Representing the set Δ.

Corollary 4.3.1. *For every factor u of $\mathbf{T}$, there exists $(\delta_0, \delta_1) \in \Delta$ such that*

$$\underline{\mathbf{\Psi}}(u) = |u| \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} + \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix}.$$

We deduce the following lemma.

Lemma 4.3.2. *For all factors u of the Tribonacci word, we have*

$$\big||\tau(u)| - \theta\,|u|\big| \leq 1.5.$$

Proof. From Corollary 4.3.1, there exists $(\delta_0, \delta_1) \in \Delta$ such that

$$\underline{\mathbf{\Psi}}(u) = |u| \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} + \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix}.$$

We thus have

$$\begin{aligned} \underline{\mathbf{\Psi}}(\tau(u)) = M'_\tau \underline{\mathbf{\Psi}}(u) &= |u|\, M'_\tau \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} + M'_\tau \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} \\ &= |u|\,\theta \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \delta_0 \\ \delta_1 \end{pmatrix}. \end{aligned}$$

The length of $\tau(u)$ is obtained by computing the dot product of this result with $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$. Hence, we get

$$|\tau(u)| = (\alpha_0 + \alpha_1 + \alpha_2)\,\theta\,|u| + \delta_0 + \delta_1 = \theta\,|u| + \delta_0 + \delta_1.$$

□

Corollary 4.3.3. *Let u be a factor of $\mathbf{T}$. For all non-negative integers n, we have*

$$\big||\tau^n(u)| - \theta^n|u|\big| \leq \frac{1.5\,\theta^n}{\theta - 1}.$$

Proof. Since $\theta > 1$, the formula is clearly true for $n = 0$. We proceed by induction on n. First, using the triangular inequality gives

$$\begin{aligned}\big||\tau^n(u)| - \theta^n|u|\big| &= \Big|\Big(|\tau^n(u)| - \theta^{n-1}|\tau(u)|\Big) + \Big(\theta^{n-1}|\tau(u)| - \theta^n|u|\Big)\Big| \\ \big||\tau^n(u)| - \theta^n|u|\big| &\leq \big||\tau^{n-1}(\tau(u))| - \theta^{n-1}|\tau(u)|\big| + \theta^{n-1}\big||\tau(u)| - \theta|u|\big|.\end{aligned}$$

Using the induction hypothesis on the first term and Lemma 4.3.2 on the second term, we get

$$\big||\tau^n(u)| - \theta^n|u|\big| \leq \frac{1.5\,\theta^{n-1}}{\theta - 1} + \theta^{n-1}\,1.5 \leq \frac{1.5\,\theta^n}{\theta - 1}.$$

□

As an immediate corollary, we make note of the following fact.

Corollary 4.3.4. *Let u be a factor of $\mathbf{T}$. For all non-negative integers n, we have*

$$|u| \geq \frac{|\tau^n(u)|}{\theta^n} - \frac{1.5}{\theta - 1}.$$

The following property will also be useful. Note that all along this chapter, we will use left eigenvectors of M_τ.

Lemma 4.3.5. *Let λ be an eigenvalue of M_τ such that $|\lambda| < 1$ and let $\underline{\mathbf{r}}$ be a left eigenvector of M_τ associated with λ. Then for every word $u \in \{0,1,2\}^*$,*

$$\underline{\mathbf{r}} \cdot P_3\left(\underline{\mathbf{\Psi}}(u) \otimes \begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2\end{pmatrix}^{\mathsf{T}}\right) = \underline{\mathbf{r}} \cdot P_3\left(\begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2\end{pmatrix}^{\mathsf{T}} \otimes \underline{\mathbf{\Psi}}(u)\right) = 0.$$

Proof. By linearity of the Kronecker product, it is enough to show that this is true for every letter a. Recall that the matrix M_τ is given by Lemma 4.1.5 and Equality (4.1). One can check that its characteristic polynomial can be factorized as

$$\left(X^3 - 3X^2 - X - 1\right)\left(X^3 - X^2 - X - 1\right)\left(X^3 + X^2 + X - 1\right)^2.$$

The first factor has roots $\theta^2 \approx 3.382$ and two complex conjugates of modulus less than 1. The second factor is the characteristic polynomial of M'_τ. It has θ as dominant root and two complex conjugate roots of modulus less than 1. Finally, the third polynomial has a real root ≈ 0.543 and two complex conjugate roots of modulus larger than 1. Thus we have four simple eigenvalues of modulus less than 1 and one eigenvalue of geometric multiplicity 2. One can find six linearly independent eigenvectors such that any left eigenvector $\underline{\mathbf{r}}$ of M_τ is a linear combination of these vectors. All these (exact) computations can be carried on in a Computer Algebra System such as `Mathematica`:

```
> dens = Eigenvectors[mprime][[1]]
         / Apply[Plus,Eigenvectors[mprime][[1]]]
> N[%]
[Out] {0.543689, 0.295598, 0.160713}
```

Let j be the index of an eigenvalue of modulus less than one. We use the transpose of M_τ to take into account left eigenvectors.

```
> Dot[JordanDecomposition[Transpose[m]][[1]][[All, j]],
   PadLeft[Flatten[KroneckerProduct[{1,0,0}, dens]],12]
> FullSimplify[%]
[Out] 0
```

□

Notice that this is not a lucky coincidence. It comes from the fact that $\begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2\end{pmatrix}$ is a right eigenvector of M'_τ and that all the left eigenvectors of M_τ can be expressed from the left eigenvectors of M'_τ using the Kronecker product.

We can show a first bound.

Lemma 4.3.6. *Let $\underline{\mathbf{r}}$ be a row vector in $\mathbb{C}^{12}$. For all integers n, words $p \in \mathrm{PPref}(\tau^n)$ and $s \in \mathrm{PSuff}(\tau^n)$, there exist two positive real numbers $c_1(\underline{\mathbf{r}}, p, s)$ and $c_2(\underline{\mathbf{r}}, p, s)$ such that, for all words u, if pus is a factor of $\mathbf{T}$ then*

$$\left|\underline{\mathbf{r}} \cdot \left(\underline{\mathbf{\Phi}}(ps) + P_3\left(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s)\right)\right)\right| \leq c_1(\underline{\mathbf{r}}, p, s)|u| + c_2(\underline{\mathbf{r}}, p, s).$$

Moreover, if

$$\underline{\mathbf{r}} \cdot P_3\left(\underline{\mathbf{\Psi}}(p) \otimes \begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2\end{pmatrix}^{\mathsf{T}} + \begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2\end{pmatrix}^{\mathsf{T}} \otimes \underline{\mathbf{\Psi}}(s)\right) = 0,$$

then $c_1(\underline{\mathbf{r}}, p, s) = 0$.

Proof. Let

$$\begin{pmatrix}\delta_0 \\ \delta_1 \\ \delta_2\end{pmatrix} = \underline{\mathbf{\Psi}}(u) - |u| \begin{pmatrix}\alpha_0 \\ \alpha_1 \\ \alpha_2\end{pmatrix}.$$

For all $p' \in \mathrm{PSuff}(p)$ and $s' \in \mathrm{PPref}(s)$, $p'us'$ is a factor of $\mathbf{T}$. From Corollary 4.3.1, we deduce that for all $i \in \{0, 1, 2\}$,

$$\begin{array}{rcccl} -1.5 & \leq & |p'us'|_i - \alpha_i|p'us'| & \leq & 1.5 \\ -1.5 & \leq & |p's'|_i - \alpha_i|p's'| + \delta_i & \leq & 1.5 \\ -1.5 - |p's'|_i + \alpha_i|p's'| & \leq & \delta_i & \leq & 1.5 - |p's'|_i + \alpha_i|p's'|. \end{array}$$

Let

$$l_i = \max_{\substack{s' \in \mathrm{PPref}(s) \\ p' \in \mathrm{PSuff}(p)}} -1.5 - |p's'|_i + \alpha_i|p's'|$$

and

$$u_i = \min_{\substack{s' \in \mathrm{PPref}(s) \\ p' \in \mathrm{PSuff}(p)}} 1.5 - |p's'|_i + \alpha_i |p's'| \,.$$

Then, we can deduce the following bounds: $l_i \leq \delta_i \leq u_i$.

For the sake of notation, we write $\underline{\boldsymbol{\delta}} = \begin{pmatrix} \delta_0 & \delta_1 & \delta_2 \end{pmatrix}^\intercal$ as it was done with $\underline{\boldsymbol{\alpha}} = \begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 \end{pmatrix}^\intercal$. It gives

$$\begin{aligned}
&|\underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s)))| \\
&= |\underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3\left(\underline{\mathbf{\Psi}}(p) \otimes [\underline{\boldsymbol{\alpha}}\,|u| + \underline{\boldsymbol{\delta}}] + [\underline{\boldsymbol{\alpha}}\,|u| + \underline{\boldsymbol{\delta}}] \otimes \underline{\mathbf{\Psi}}(s)\right))| \\
&\leq \underbrace{|\underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3\left(\underline{\mathbf{\Psi}}(p) \otimes \underline{\boldsymbol{\delta}} + \underline{\boldsymbol{\delta}} \otimes \underline{\mathbf{\Psi}}(s)\right))|}_{=: f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, \delta_2)} + |w| \underbrace{|\underline{\mathbf{r}} \cdot P_3\left(\underline{\mathbf{\Psi}}(p) \otimes \underline{\boldsymbol{\alpha}} + \underline{\boldsymbol{\alpha}} \otimes \underline{\mathbf{\Psi}}(s)\right)|}_{=: c_1(\underline{\mathbf{r}}, p, s)}.
\end{aligned}$$

We need $c_2(\underline{\mathbf{r}}, p, s)$ to bound f for all possible values of the δ_i, so we can take

$$c_2(\underline{\mathbf{r}}, p, s) := \max_{\substack{l_0 \leq \delta_0 \leq u_0 \\ l_1 \leq \delta_1 \leq u_1 \\ l_2 \leq \delta_2 \leq u_2 \\ \delta_0 + \delta_1 + \delta_2 = 0}} f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, \delta_2).$$

Then,

$$|\underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + \underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s))| \leq c_1(\underline{\mathbf{r}}, p, s)|u| + c_2(\underline{\mathbf{r}}, p, s).$$

□

Observe that

$$\begin{aligned}
&\max_{\substack{l_0 \leq \delta_0 \leq u_0 \\ l_1 \leq \delta_1 \leq u_1 \\ l_2 \leq \delta_2 \leq u_2 \\ \delta_0 + \delta_1 + \delta_2 = 0}} f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, \delta_2) \leq \\
&\qquad \min \Bigg\{ \max_{\substack{l_0 \leq \delta_0 \leq u_0 \\ l_1 \leq \delta_1 \leq u_1 \\ l_2 \leq \delta_2 \leq u_2}} f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, \delta_2), \max_{\substack{l_0 \leq \delta_0 \leq u_0 \\ l_1 \leq \delta_1 \leq u_1}} f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, -\delta_0 - \delta_1), \\
&\qquad \max_{\substack{l_0 \leq \delta_0 \leq u_0 \\ l_2 \leq \delta_2 \leq u_2}} f(\underline{\mathbf{r}}, p, s, \delta_0, -\delta_0 - \delta_2, \delta_2), \max_{\substack{l_1 \leq \delta_1 \leq u_1 \\ l_2 \leq \delta_2 \leq u_2}} f(\underline{\mathbf{r}}, p, s, -\delta_1 - \delta_2, \delta_1, \delta_2) \Bigg\}.
\end{aligned}$$

Moreover, for any p and s, there exist complex numbers a, b, c, d such that $f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, \delta_2) = |a + b\delta_0 + c\delta_1 + d\delta_2|$. By Lemma 4.1.6, the maximum is then reached on a vertex. Thus, for instance for the first of these maxima we have

$$\max_{\substack{l_0 \leq \delta_0 \leq u_0 \\ l_1 \leq \delta_1 \leq u_1}} f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, -\delta_0 - \delta_1) = \max_{\binom{\delta_0}{\delta_1} \in \left\{ \binom{l_0}{l_1}, \binom{l_0}{u_1}, \binom{u_0}{l_1}, \binom{u_0}{u_1} \right\}} f(\underline{\mathbf{r}}, p, s, \delta_0, \delta_1, -\delta_0 - \delta_1).$$

A similar conclusion can be reached for the three other maxima and we can easily compute a good upper bound on $c_2(\underline{\mathbf{r}}, p, s)$.

The second part of the statement of the previous lemma is useful for the following result.

Lemma 4.3.7. *In the statement of Lemma 4.3.6, if $\underline{\mathbf{r}}$ is a left eigenvector of M_τ associated with an eigenvalue of modulus less than 1, then $c_1(\underline{\mathbf{r}}, p, s) = 0$.*

Proof. Using Lemma 4.3.5, we get, for all words $p \in \mathrm{PPref}(\tau^n)$ and $s \in \mathrm{PSuff}(\tau^n)$,

$$\underline{\mathbf{r}} \cdot P_3 \left(\underline{\mathbf{\Psi}}(p) \otimes \underline{\boldsymbol{\alpha}} + \underline{\boldsymbol{\alpha}} \otimes \underline{\mathbf{\Psi}}(s) \right) = 0.$$

This concludes the proof. □

We can now obtain two different kinds of bounds on extended Parikh vectors of factors of the Tribonacci word. First we essentially take care of the large eigenvalues.

Proposition 4.3.8. *Let $\underline{\mathbf{r}}$ be a left eigenvector of M_τ having λ as associated eigenvalue. If $|\lambda| < \theta$, then there exists a constant $C(\underline{\mathbf{r}})$ such that, for all factors u of $\mathbf{T}$,*

$$\frac{|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)|}{|u|} \leq C(\underline{\mathbf{r}}).$$

Proof. Let n be a positive integer. We work with τ^n because it gives us a degree of freedom to obtain smaller bounds, but working with τ is enough to get a bound.

Let $u = u_0$ be a factor of $\mathbf{T}$. There exist a non-empty factor of the Tribonacci word u_1, $p \in \mathrm{PPref}(\tau^n(a))$ and $s \in \mathrm{PSuff}(\tau^n(b))$, where a (resp., b) is the first (resp., last) letter of u_1, such that $pus = \tau^n(u_1)$. We say that u_1 is the *preimage* of u_0 (by τ^n). In particular $\underline{\mathbf{\Phi}}(pus) = \underline{\mathbf{\Phi}}(\tau^n(u_1))$. Using the same arguments as in Lemma 4.1.3, one has

$$\underline{\mathbf{\Phi}}(pus) = \underline{\mathbf{\Phi}}(u) + \underline{\mathbf{\Phi}}(ps) + P_3(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s)).$$

Hence, we have

$$\begin{aligned}
\underline{\mathbf{\Phi}}(u) &= \underline{\mathbf{\Phi}}(\tau^n(u_1)) - \underline{\mathbf{\Phi}}(ps) - P_3(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s)) \\
\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u) &= \underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(\tau^n(u_1)) - \underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s))) \\
\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u) &= \underline{\mathbf{r}} \cdot (M_\tau^n \underline{\mathbf{\Phi}}(u_1)) - \underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s))) \\
\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u) &= \lambda^n \underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u_1) - \underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s))) \\
|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| &\leq |\lambda|^n |\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u_1)| + \left| \underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(ps) + P_3 \left(\underline{\mathbf{\Psi}}(p) \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{\Psi}}(s) \right)) \right|.
\end{aligned}$$

From Lemma 4.3.6, we get

$$|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| \leq |\lambda|^n |\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u_1)| + c_1(\underline{\mathbf{r}}, p, s) \, |u| + c_2(\underline{\mathbf{r}}, p, s). \tag{4.4}$$

Let ℓ be some integer with $\ell > \theta^n\left(2+\frac{1.5}{\theta-1}\right)$. Assume that $|u| \geq \ell$. From Corollary 4.3.3, $|u_1| \geq 2$. Moreover, if $u_1 = a_1a_2\cdots a_{|u_1|-1}a_{|u_1|}$, then we can apply Corollary 4.3.3 on $a_2\cdots a_{|u_1|-1}$:

$$\begin{aligned}
|\tau^n(a_2a_3\ldots a_{|u_1|-1})| &\geq \theta^n\left(|u_1|-2\right) - \theta^n\frac{1.5}{\theta-1}\\
|u|\left(1+\frac{\theta^n\left(2+\frac{1.5}{\theta-1}\right)}{|u|}\right) &\geq \theta^n|u_1|\\
|u|\,\frac{\ell+\theta^n\left(2+\frac{1.5}{\theta-1}\right)}{\ell} &\geq \theta^n|u_1|\\
|u| &\geq \theta^n|u_1|\,\frac{\ell}{\ell+\theta^n\left(2+\frac{1.5}{\theta-1}\right)}.
\end{aligned}$$

For the sake of notation, let $\iota(\ell,n) := \frac{\ell}{\ell+\theta^n\left(2+\frac{1.5}{\theta-1}\right)}$. Otherwise stated, $|u| \geq \theta^n|u_1|\iota(\ell,n)$. We can now compute the following bound:

$$\begin{aligned}
\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} &\leq \frac{|\lambda|^n|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u_1)|}{|u|} + c_1(\underline{\mathbf{r}},p,s) + \frac{c_2(\underline{\mathbf{r}},p,s)}{|u|}\\
\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} &\leq \frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\,\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u_1)|}{|u_1|} + c_1(\underline{\mathbf{r}},p,s) + \frac{c_2(\underline{\mathbf{r}},p,s)}{\ell}.
\end{aligned}$$

Let $c_3(\underline{\mathbf{r}}) = \max\left\{c_1(\underline{\mathbf{r}},p,s) + \frac{c_2(\underline{\mathbf{r}},p,s)}{\ell} : p \in \mathrm{PPref}(\tau^n), s \in \mathrm{PSuff}(\tau^n)\right\}$. We get

$$\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} \leq \frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\,\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u_1)|}{|u_1|} + c_3(\underline{\mathbf{r}}). \tag{4.5}$$

The reader may notice that on the right-hand side, we have a factor of the same form as the left-hand side so, if $|u_1| \geq \ell$, we are tempted to apply again the same inequality on u_1. Let u_2 be the preimage of u_1 (as defined in the beginning of this proof). We can define a sequence $u = u_0, u_1, u_2, \ldots$ of factors of $\mathbf{T}$ such that u_{i+1} is the preimage of u_i by τ^n.

By induction on (4.5), one can find a factor u_i of $\mathbf{T}$ with $|u_i| \leq \ell$ such that

$$\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} \leq \left(\frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\right)^i\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u_i)|}{|u_i|} + c_3(\underline{\mathbf{r}})\sum_{k=0}^{i-1}\left(\frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\right)^k.$$

By assumption, $|\lambda| < \theta$, so numerically computing the modulus of the eigenvalues of M_τ distinct from θ and θ^2, we obtain $|\lambda|/\theta < .74$. For all n, there exists a large enough ℓ such that $\iota(\ell,n) > .74$. Indeed for a fixed n, $\lim_{\ell\to+\infty}\iota(\ell,n) = 1$. For such a choice, we have

$$\frac{|\lambda|^n}{\iota(\ell,n)\theta^n} \leq \frac{|\lambda|}{\iota(\ell,n)\theta} < 1.$$

Now that we have shown that a convenient pair (ℓ, n) exists, in practice, it is enough to choose a pair such that

$$\frac{|\lambda|^n}{\iota(\ell,n)\theta^n} < 1. \tag{4.6}$$

Since

$$\left(\frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\right)^i < \frac{|\lambda|^n}{\iota(\ell,n)\theta^n} \text{ and } \sum_{k=0}^{i-1}\left(\frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\right)^k < \frac{1}{1-\frac{|\lambda|^n}{\iota(\ell,n)\theta^n}},$$

we obtain the following bound:

$$\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} \leq \frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u_i)|}{|u_i|} + c_3(\underline{\mathbf{r}})\,\frac{\iota(\ell,n)\theta^n}{\iota(\ell,n)\theta^n - |\lambda|^n}.$$

Finally, the quantity $\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u_i)|}{|u_i|}$ is bounded by

$$\max_{\substack{u\in\mathcal{L}(\tau)\\ |u|\leq\ell}} \frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|}.$$

To conclude, for any positive integers n and ℓ with

$$\frac{|\lambda|^n}{\iota(\ell,n)\theta^n} < 1 \quad \text{and} \quad \ell > \theta^n\left(2+\frac{1.5}{\theta-1}\right) \tag{4.7}$$

and any factor u of the Tribonacci word (without length restriction on u), we have

$$\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} \leq \max\left\{\max_{\substack{u\in\mathcal{L}(\tau)\\ |u|\leq\ell}}\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|}, \frac{|\lambda|^n}{\iota(\ell,n)\theta^n}\max_{\substack{u\in\mathcal{L}(\tau)\\ |u|\leq\ell}}\frac{|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)|}{|u|} + c_3(\underline{\mathbf{r}})\,\frac{\iota(\ell,n)\theta^n}{\iota(\ell,n)\theta^n-|\lambda|^n}\right\}.$$

As already discussed for any fixed n, there exists a large enough ℓ such that this is true. Since there are only finitely many factors of bounded size, this quantity is clearly defined and easy to compute. This concludes the proof. □

Note that a good choice of n and ℓ yields better bounds for the computations that have to be carried on. In particular, we want ℓ as large as possible, unfortunately ℓ has to be limited to the hundreds in order for the computations to be feasible. Discussions about the choice of n and ℓ are postponed at the end of this chapter.

The bound given by Proposition 4.3.8 will only be useful for the eigenvalues λ such that $|\lambda| \geq 1$. When $|\lambda| < 1$, the following result is stronger.

Proposition 4.3.9. *Let $\underline{\mathbf{r}}$ be an eigenvector of M_τ and λ be the associated eigenvalue. If $|\lambda| < 1$, then there exists a constant $C(\underline{\mathbf{r}})$ such that for all factors u of $\mathbf{T}$,*

$$|\underline{\mathbf{r}}\cdot\underline{\mathbf{\Phi}}(u)| \leq C(\underline{\mathbf{r}}).$$

Proof. We can reproduce the beginning of the proof of Proposition 4.3.8 and get (4.4). But since $|\lambda| < 1$, Lemma 4.3.7 implies that $c_1(\underline{\mathbf{r}}, p, s) = 0$. So, we have

$$|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| \leq |\lambda|^n |\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u_1)| + c_2(\underline{\mathbf{r}}, p, s)$$

where u_1 is a preimage of u. We can iterate the process by taking preimages and we have a sequence of factors $u = u_0, u_1, u_2, \ldots$. We deduce by induction that for any integer i,

$$|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| \leq |\lambda|^{ni} |\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u_i)| + \sum_{j=0}^{i-1} |\lambda|^{jn} \max_{\substack{p \in \mathrm{PPref}(\tau^n) \\ s \in \mathrm{PSuff}(\tau^n)}} c_2(\underline{\mathbf{r}}, p, s).$$

Moreover, there exists an integer k, such that for all $i \geq k$, $|u_i| = 1$. Then, for all $i \geq k$,

$$|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| \leq |\lambda|^{ni} \max_{a \in \mathcal{A}} |\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(a)| + \sum_{j=0}^{i-1} |\lambda|^{jn} \max_{\substack{p \in \mathrm{PPref}(\tau^n) \\ s \in \mathrm{PSuff}(\tau^n)}} c_2(\underline{\mathbf{r}}, p, s).$$

This implies

$$|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| \leq \lim_{i \to \infty} \left(|\lambda|^{ni} \max_{a \in \mathcal{A}} |\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(a)| + \sum_{j=0}^{i-1} |\lambda|^{jn} \max_{\substack{p \in \mathrm{PPref}(\tau^n) \\ s \in \mathrm{PSuff}(\tau^n)}} c_2(\underline{\mathbf{r}}, p, s) \right)$$

$$|\underline{\mathbf{r}} \cdot \underline{\mathbf{\Phi}}(u)| \leq \max_{\substack{p \in \mathrm{PPref}(\tau^n) \\ s \in \mathrm{PSuff}(\tau^n)}} \frac{c_2(\underline{\mathbf{r}}, p, s)}{1 - |\lambda|^n}$$

and concludes the proof. □

4.3.2 Bounds on templates

This subsection contains several lemmas giving necessary conditions on templates to be realizable by τ.

Lemma 4.3.10. *Let λ be an eigenvalue of M_τ such that $|\lambda| < 1$. For every left eigenvector $\underline{\mathbf{r}}$ of M_τ associated with λ and for every realizable template $t = [\underline{\mathbf{d}}, \underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}, a_1, a_2]$,*

$$\min_{(\delta_0, \delta_1) \in \Delta} \left| \underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3 \left(\underline{\mathbf{D_b}} \otimes \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} + \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} \otimes \underline{\mathbf{D_e}} \right) \right) \right| \leq 2C(\underline{\mathbf{r}})$$

where $C(\underline{\mathbf{r}})$ is the constant from Proposition 4.3.9.

Proof. From Proposition 4.3.9, for any two factors $u, v \in \mathcal{L}(\tau)$, we have $|\underline{\mathbf{r}} \cdot (\underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v))| \leq 2C(\underline{\mathbf{r}})$. On the other hand, t is realizable so there are two factors u and v of the Tribonacci word such that

$$\underline{\mathbf{\Phi}}(u) - \underline{\mathbf{\Phi}}(v) = \underline{\mathbf{d}} + P_3(\underline{\mathbf{D_b}} \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{D_e}}).$$

This implies

$$\left|\underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3(\underline{\mathbf{D_b}} \otimes \underline{\mathbf{\Psi}}(u) + \underline{\mathbf{\Psi}}(u) \otimes \underline{\mathbf{D_e}})\right)\right| \leq 2C(\underline{\mathbf{r}}).$$

From Corollary 4.3.1, there are $(\delta_0, \delta_1) \in \Delta$ such that

$$\underline{\mathbf{\Psi}}(u) = |u| \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} + \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix}.$$

We write again $\underline{\boldsymbol{\alpha}} = \begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 \end{pmatrix}^\mathsf{T}$ and $\underline{\boldsymbol{\delta}} = \begin{pmatrix} \delta_0 & \delta_1 & -\delta_0 - \delta_1 \end{pmatrix}^\mathsf{T}$.

It gives

$$2C(\underline{\mathbf{r}}) \geq \left|\underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3\left(\underline{\mathbf{D_b}} \otimes (\underline{\boldsymbol{\alpha}}|u| + \underline{\boldsymbol{\delta}}) + (\underline{\boldsymbol{\alpha}}|u| + \underline{\boldsymbol{\delta}}) \otimes \underline{\mathbf{D_e}}\right)\right)\right|.$$

From Lemma 4.3.5, we get

$$\underline{\mathbf{r}} \cdot P_3\left(\underline{\mathbf{D_b}} \otimes \underline{\boldsymbol{\alpha}}|u|\right) = \underline{\mathbf{r}} \cdot P_3\left(\underline{\boldsymbol{\alpha}}|u| \otimes \underline{\mathbf{D_e}}\right) = 0$$

and thus we finally obtain

$$\begin{aligned} 2C(\underline{\mathbf{r}}) &\geq \left|\underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3\left(\underline{\mathbf{D_b}} \otimes \underline{\boldsymbol{\delta}} + \underline{\boldsymbol{\delta}} \otimes \underline{\mathbf{D_e}}\right)\right)\right| \\ &\geq \min_{(\delta_0,\delta_1)\in\Delta} \left|\underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3\left(\underline{\mathbf{D_b}} \otimes \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} + \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} \otimes \underline{\mathbf{D_e}}\right)\right)\right|. \end{aligned}$$

□

This bound is not so easy to use because of the complicated minimum. It can be computed using tools from optimization. However, we can simply use this bound as follows.

First, for the sake of notation, let

$$f(\delta_0, \delta_1) = \underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3\left(\underline{\mathbf{D_b}} \otimes \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} + \begin{pmatrix} \delta_0 \\ \delta_1 \\ -\delta_0 - \delta_1 \end{pmatrix} \otimes \underline{\mathbf{D_e}}\right)\right).$$

Then

$$\min_{(\delta_0,\delta_1)\in\Delta} |f(\delta_0, \delta_1)| \geq \sqrt{\min_{(\delta_0,\delta_1)\in\Delta} \mathrm{Re}\left(f(\delta_0, \delta_1)\right)^2 + \min_{(\delta_0,\delta_1)\in\Delta} \mathrm{Im}\left(f(\delta_0, \delta_1)\right)^2}.$$

Let I_{Re} and I_{Im} be intervals such that

$$I_{Re} = \left[\min_{(\delta_0,\delta_1)\in\Delta} \mathrm{Re}\left(f(\delta_0, \delta_1)\right), \max_{(\delta_0,\delta_1)\in\Delta} \mathrm{Re}\left(f(\delta_0, \delta_1)\right)\right]$$

and

$$I_{Im} = \left[\min_{(\delta_0,\delta_1)\in\Delta} \mathrm{Im}\left(f(\delta_0,\delta_1)\right), \max_{(\delta_0,\delta_1)\in\Delta} \mathrm{Im}\left(f(\delta_0,\delta_1)\right) \right].$$

Then

$$\min_{(\delta_0,\delta_1)\in\Delta} |f(\delta_0,\delta_1)| \geq \sqrt{\min_{y\in I_{Re}} y^2 + \min_{y\in I_{Im}} y^2}.$$

Thus any template for which this last quantity is greater than $2C(\underline{\mathbf{r}})$ is not realizable.

Observe that each of the four interval bounds is reached for a vertex of the polytope, that is $\begin{pmatrix}\delta_0\\ \delta_1\end{pmatrix} \in \left\{ \begin{pmatrix}1.5\\ -1.5\end{pmatrix}, \begin{pmatrix}1.5\\ 0\end{pmatrix}, \begin{pmatrix}0\\ 1.5\end{pmatrix}, \begin{pmatrix}-1.5\\ 1.5\end{pmatrix}, \begin{pmatrix}-1.5\\ 0\end{pmatrix}, \begin{pmatrix}0\\ -1.5\end{pmatrix} \right\}$. This is due to the fact that f is linear (and thus convex) over the convex set Δ. The proof of this fact is identical to the proof of Lemma 4.1.6.

This allows us to remove many templates from the set of templates, but this is not enough to obtain a finite set, so we need to somehow use the bounds on the other eigenvectors as well.

Lemma 4.3.11. *Let L be a positive integer. Let λ be an eigenvalue of M_τ such that $|\lambda| < \theta$. Then, for all eigenvectors $\underline{\mathbf{r}}$ of M_τ associated with λ, there exists a constant $C(\underline{\mathbf{r}})$ such that for any template $t = [\underline{\mathbf{d}}, \underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}, a_1, a_2]$ realized by a pair of factors of the Tribonacci word (u, v) with $|u| \geq L$, we have*

$$\left|\underline{\mathbf{r}} \cdot P_3\left(\underline{\mathbf{D_b}} \otimes \underline{\boldsymbol{\alpha}} + \underline{\boldsymbol{\alpha}} \otimes \underline{\mathbf{D_e}}\right)\right| \leq \frac{2L - \sum_{i=1}^{3} \underline{\mathbf{d}}_i}{L} C(\underline{\mathbf{r}}) + \max_{(\delta_0,\delta_1)\in\Delta} \frac{\left|\underline{\mathbf{r}} \cdot \left(\underline{\mathbf{d}} + P_3\left(\underline{\mathbf{D_b}} \otimes \underline{\boldsymbol{\delta}} + \underline{\boldsymbol{\delta}} \otimes \underline{\mathbf{D_e}}\right)\right)\right|}{L}.$$

Proof. From Proposition 4.3.8, there is a constant $C(\underline{\mathbf{r}})$ such that for any two factors $u, v \in \mathcal{L}(\tau)$, we have $|\underline{\mathbf{r}} \cdot (\underline{\boldsymbol{\Phi}}(u) - \underline{\boldsymbol{\Phi}}(v))| \leq (|u| + |v|)C(\underline{\mathbf{r}})$. On the other hand, t is realized by a pair of factors (u, v) of $\mathbf{T}$ with $|u| \geq L$ such that

$$\underline{\boldsymbol{\Phi}}(u) - \underline{\boldsymbol{\Phi}}(v) = \underline{\mathbf{d}} + P_3(\underline{\mathbf{D_b}} \otimes \underline{\boldsymbol{\Psi}}(u) + \underline{\boldsymbol{\Psi}}(u) \otimes \underline{\mathbf{D_e}}).$$

This implies

$$\left|\underline{\mathbf{r}} \cdot (\underline{\mathbf{d}} + P_3(\underline{\mathbf{D_b}} \otimes \underline{\boldsymbol{\Psi}}(u) + \underline{\boldsymbol{\Psi}}(u) \otimes \underline{\mathbf{D_e}}))\right| \leq (|u| + |v|)C(\underline{\mathbf{r}}).$$

From Corollary 4.3.1, there is $(\delta_0, \delta_1) \in \Delta$ such that

$$\underline{\boldsymbol{\Psi}}(u) = |u| \begin{pmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\end{pmatrix} + \begin{pmatrix}\delta_0\\ \delta_1\\ -\delta_0 - \delta_1\end{pmatrix}.$$

We write again $\underline{\boldsymbol{\alpha}} = \begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2\end{pmatrix}^\mathsf{T}$ and $\underline{\boldsymbol{\delta}} = \begin{pmatrix}\delta_0 & \delta_1 & -\delta_0 - \delta_1\end{pmatrix}^\mathsf{T}$.

It gives

$$(|u|+|v|)C(\underline{\mathbf{r}}) \geq \left|\underline{\mathbf{r}}\cdot\left(\underline{\mathbf{d}}+P_3\left(\underline{\mathbf{D_b}}\otimes(\underline{\boldsymbol{\alpha}}|u|+\underline{\boldsymbol{\delta}})+(\underline{\boldsymbol{\alpha}}|u|+\underline{\boldsymbol{\delta}})\otimes\underline{\mathbf{D_e}}\right)\right)\right|.$$

We can now apply the triangular inequality and divide each side by $|u|$:

$$\left|\underline{\mathbf{r}}\cdot P_3\left(\underline{\mathbf{D_b}}\otimes\underline{\boldsymbol{\alpha}}+\underline{\boldsymbol{\alpha}}\otimes\underline{\mathbf{D_e}}\right)\right| \leq \frac{|u|+|v|}{|u|}C(\underline{\mathbf{r}}) + \frac{\left|\underline{\mathbf{r}}\cdot\left(\underline{\mathbf{d}}+P_3\left(\underline{\mathbf{D_b}}\otimes\underline{\boldsymbol{\delta}}+\underline{\boldsymbol{\delta}}\otimes\underline{\mathbf{D_e}}\right)\right)\right|}{|u|}.$$

Finally, we can use the inequality $|u| \geq L$:

$$\begin{aligned}&\left|\underline{\mathbf{r}}\cdot P_3\left(\underline{\mathbf{D_b}}\otimes\underline{\boldsymbol{\alpha}}+\underline{\boldsymbol{\alpha}}\otimes\underline{\mathbf{D_e}}\right)\right| \leq \\ &\frac{2L-\sum_{i=1}^{3}\underline{\mathbf{d}}_i}{L}C(\underline{\mathbf{r}}) + \max_{(\delta_0,\delta_1)\in\Delta}\frac{\left|\underline{\mathbf{r}}\cdot\left(\underline{\mathbf{d}}+P_3\left(\underline{\mathbf{D_b}}\otimes\underline{\boldsymbol{\delta}}+\underline{\boldsymbol{\delta}}\otimes\underline{\mathbf{D_e}}\right)\right)\right|}{L}.\end{aligned} \tag{4.8}$$

This concludes the proof. □

The quantity of the left-hand side and the first term on the right-hand side in (4.8) are straightforward to compute. For the last term, Lemma 4.1.6 tells us that the maximum is in fact necessarily reached on a vertex of the polytope, that is

$$\begin{aligned}\max_{(\delta_0,\delta_1)\in\Delta}\frac{\left|\underline{\mathbf{r}}\cdot\left(\underline{\mathbf{d}}+P_3\left(\underline{\mathbf{D_b}}\otimes\underline{\boldsymbol{\delta}}+\underline{\boldsymbol{\delta}}\otimes\underline{\mathbf{D_e}}\right)\right)\right|}{L} \leq \\ \max_{\binom{\delta_0}{\delta_1}\in\left\{\binom{1.5}{0},\binom{1.5}{-1.5},\binom{0}{1.5},\binom{-1.5}{0},\binom{-1.5}{1.5},\binom{0}{-1.5}\right\}}\frac{\left|\underline{\mathbf{r}}\cdot\left(\underline{\mathbf{d}}+P_3\left(\underline{\mathbf{D_b}}\otimes\underline{\boldsymbol{\delta}}+\underline{\boldsymbol{\delta}}\otimes\underline{\mathbf{D_e}}\right)\right)\right|}{L}.\end{aligned}$$

4.4 Proof of the main result

With all these lemmas, we are ready to show our main result.

Theorem 4.4.1. *Two factors of the Tribonacci word are 2-binomially equivalent if and only if they are equal.*

Proof. Let $\mathcal{T} = \{[\underline{\mathbf{0}},\underline{\mathbf{0}},\underline{\mathbf{0}},a_1,a_2] : a_1 \neq a_2\}$. Let us show that no template from $\mathcal{T}$ is realizable. Let $L = 15$. We can easily check with a computer that no pair of factors of $\mathbf{T}$ with $\min(|u|,|v|) \leq L$ realizes a template t from the set $\mathcal{T}$ (see Section A.6 of the first appendix). Indeed, since for all $t \in \mathcal{T}$, $\underline{\mathbf{d}} = \underline{\mathbf{0}}$, $\underline{\mathbf{D_b}} = \underline{\mathbf{0}}$ and $\underline{\mathbf{D_e}} = \underline{\mathbf{0}}$, we know that a pair of words (u,v) realizes t if and only if $\underline{\boldsymbol{\Phi}}(u) - \underline{\boldsymbol{\Phi}}(v) = 0$. It just suffices to check that for all $n \leq L$, $b^{(2)}_{\mathbf{T}}(n) = p_{\mathbf{T}}(n)$.

Now, from Proposition 4.2.9, if $t \in \mathcal{T}$ is realized then one of its ancestors is realized by a pair (u,v) with $L \leq \min(|u|,|v|) \leq 2L$.

Lemmas 4.3.10 and 4.3.11 give us two sets of inequalities that any template realized by a pair (u, v) of factors of Tribonacci with $|u| \geq L$ must respect. Let $\mathcal{X}$ be the set of templates that respect the bounds. Let $A_0 = \mathcal{T}$ and, for all i, let $A_{i+1} = \{\mathrm{Par}_\tau(t) \cap \mathcal{X} : t \in A_i\}$. Then clearly $\mathrm{RAnc}_\tau(t) \subseteq \bigcup_{i \in \mathbb{N}} A_i$. Each A_i can be easily computed and it can be checked by a computer program that the set $\bigcup_{i \in \mathbb{N}} A_i$ is finite, as done in Section A.5 of the appendices.

We can finally check with a computer that there is no pair (u, v) of factors of $\mathbf{T}$ with $L \leq \min(|u|, |v|) \leq 2L$ that realizes any element of $\bigcup_{i \in \mathbb{N}} A_i$ (once again, see Section A.6 of the appendix). Thus no template of $\mathcal{T}$ is realizable. By Lemma 4.2.3, we can conclude that the 2-binomial complexity of the Tribonacci word is equal to its factor complexity. □

Accompanying this chapter is an implementation in `Mathematica` of all the computations described in this theorem as well as in the previous lemmas and propositions. An online version [70] is available, but the code is also given with some additional explanation in Appendix A. We also have an independent `C++` implementation that is much faster, but uses machine floating point arithmetic whose accuracy cannot be guaranteed (in this case, however, we obtain exactly the same set of templates). Diagonalizing the matrix of Tribonacci gives 4 eigenvectors to which Lemma 4.3.10 can be applied. Since there are two pairs of conjugate complex vectors, it is useless to keep more than one of each pair. However, by taking a linear combination of these two, we get another eigenvector to which we can apply Lemma 4.3.10 (in practice we only do that once, but we could take as many vectors as we want from this 2-dimensional space). Taking conjugates into account, we only keep 4 of the 6 eigenvectors that correspond to an eigenvalue of norm less than 1. For each of these 7 eigenvectors, we choose[1] $\ell = 600$ and the best $1 \leq n \leq 6$ when applying Lemma 4.3.10 or Lemma 4.3.11. What remains is done as described in the manuscript. We obtain a set of 241544 templates.

4.5 Possible extensions

We used an algorithm to show that the 2-binomial complexity of the Tribonacci word is equal to its factor complexity. It seems that our method can be turned into an algorithm that can decide under some mild conditions whether the factor complexity of a given morphic word is equal to its k-binomial complexity. In fact, by keeping track of the first letter of each word in templates, the "if" in Proposition 4.2.9 can be

[1] Remember that we work on τ^n and that increasing n and ℓ tend to give us better bounds but increases the computation time.

replaced by an "if and only if" (some technicalities could allow us to apply it even if the matrix is singular). Moreover, with arguments similar to the ideas from [103], one could show that we also have bounds on the eigenvectors that correspond to larger eigenvalues and that the number of templates that respect these bounds is always finite (one might need no eigenvalues has norm 1).

Observe that the notion of template was first introduced in the context of avoidance of Abelian powers [31] and, as one could expect, it seems that our technique also gives a decision algorithm for the avoidability of k-binomial powers in morphic words (and even avoidability of patterns in the k-binomial sense).

Let us now consider two independant extensions.

Arnoux–Rauzy words

Arnoux–Rauzy words (over a three-letter alphabet) can be seen as a generalization of Sturmian words; they have exactly one left and one right special factor of each length, extendable by the three letters of the alphabet [7, 108], and hence their factor complexity is $p(n) = 2n + 1$ for all $n \in \mathbb{N}_0$. The most famous Arnoux–Rauzy word is $\mathbf{T}$, this is why we got interested into its k-binomial complexity. One particular property of $\mathbf{T}$ is that it is 2-balanced, i.e., for any $n \in \mathbb{N}$, $u, v \in \mathrm{Fac}_n(\mathbf{T})$ and $a \in \mathcal{A}$, we have

$$||u|_a - |v|_a| \leq 2.$$

This is not the case in general for Arnoux–Rauzy words [21]: there exist an Arnoux–Rauzy word $\mathbf{w}$ that is not N-balanced for any N, i.e., for any $N \in \mathbb{N}$, there exist $n \in \mathbb{N}$, $u, v \in \mathrm{Fac}_n(\mathbf{w})$ and $a \in \mathcal{A}$ such that

$$||u|_a - |v|_a| > N.$$

In [21] the authors give an algorithm allowing us to build a sequence of Arnoux–Rauzy words $(\mathbf{w_i})_{i\geq 2}$ such that for any i, $\mathbf{w_i}$ is not $(i-1)$-balanced. Inspired by this result, we computed the first infinite words of this sequence and their associated k-binomial complexity. The interested reader can find in Appendix B an algorithm giving sequence $(\mathbf{w_i})_{i\geq 2}$ as described, as well as a `Mathematica` code computing the 2-binomial complexity of the first elements of this sequence. We conjecture that

$$b^{(k)}_{\mathbf{w_i}}(n) = 2n + 1,$$

for all $i \geq 2$, $k \geq 2$ and $n \in \mathbb{N}_0$.

Hence it seems that 2-balancedness of $\mathbf{T}$ helps simplifying our results but is not a necessary condition for having $b^{(k)}_{\mathbf{w}} = p_{\mathbf{w}}$, $k \geq 2$, for Arnoux–Rauzy words.

The period-doubling word

One question we may address is the following: does there exists a variant of the Pansiot theorem (Theorem 1.4.3) for k-binomial equivalence? This question is of interest since up to now, we have only seen two types behaviors for the k-binomial complexity of infinite words: words having a bounded k-binomial complexity (such as the Thue–Morse word and all fixed points of Parikh-constant morphisms), and words having a 2-binomial complexity equal to the factor one (as Sturmian words, the Tribonacci word and probably all Arnoux–Rauzy words). We thus wonder if there exists a word $\mathbf{w}$ such that

- $b_{\mathbf{w}}^{(k)}$ is unbounded for all $k \in \mathbb{N}$, and;
- for all $k \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that $b_{\mathbf{w}}^{(k)}(n) < p_{\mathbf{w}}(n)$.

We decided with Rigo and Stipulanti to study the k-binomial complexity of the period-doubling word $\mathbf{p}$. This infinite word is the fixed point starting with 0 of the morphism

$$\sigma_p : \{0,1\}^* \to \{0,1\}^* : \begin{cases} 0 & \mapsto & 01; \\ 1 & \mapsto & 00, \end{cases}$$

hence $\mathbf{p} = \lim_{n\to+\infty} \sigma_p^n(0)$. In [6, Example 6.3.4] it is shown that $\mathbf{p}_n = \nu_2(n) \bmod 2$, where $\nu_2(i)$ is the exponent of the largest power of 2 dividing i. Various properties of this fixed point were studied: see for example [113] where palindromes, rich, privileged, trapezoidal, and balanced factors of $\mathbf{p}$ were considered; see [32] for another reference concerning its palindromes.

We computed the first values of $b_{\mathbf{p}}^{(2)}$ and $b_{\mathbf{p}}^{(3)}$ with a computer. They are given in Figure 4.2 for $n \in [70]_0$. We then conjectured that these two functions are different from the factor complexity. We were able to prove the following result.

Proposition 4.5.1 (unpublished). *For all $n \geq 0$, we have*

$$b_{\mathbf{p}}^{(2)}(2^n) = p_{\mathbf{p}}(2^n)$$

and

$$b_{\mathbf{p}}^{(2)}(2^n + r) < p_{\mathbf{p}}(2^n + r)$$

for all $r \in [2^n - 1]$.

The second assertion is easily proved by induction on the length of factors of $\mathbf{p}$, since σ_p is 2-uniform[2]. The factor complexity of $\mathbf{p}$ is a folklore result: $p_{\mathbf{p}}(0) = 1$, $p_{\mathbf{p}}(1) = 2$, $p_{\mathbf{p}}(2) = 3$, $p_{\mathbf{p}}(3) = 5$,

$$p_{\mathbf{p}}(2n) = 2p_{\mathbf{p}}(n), \quad \forall n \geq 2,$$

[2]It means that $|\sigma_p(a)| = 2$ for all $a \in \mathcal{A}$.

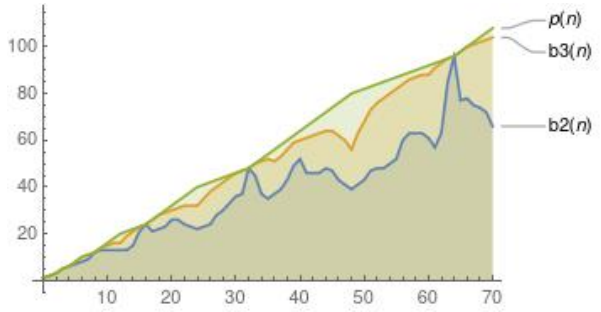


Figure 4.2: Factor, 2- and 3-binomial complexities of the period-doubling word for $n \leq 70$.

and, for all $n \geq 2$,

$$p_{\mathbf{p}}(2n+1) = 2p_{\mathbf{p}}(n) + \begin{cases} 2, & \text{if } \exists i \in \mathbb{N}_0 \text{ s.t. } 2^{i+1} \leq n < 2^{i+1} + 2^i; \\ 1, & \text{if } \exists i \in \mathbb{N}_0 \text{ s.t. } 2^{i+1} + 2^i \leq n < 2^{i+2}. \end{cases}$$

Thus for the equality in the previous proposition we need to prove that

$$\#\left\{ \left(\binom{u}{0}, \binom{u}{01} \right) : u \in \mathrm{Fac}_n(\mathbf{p}) \right\} = 3 \cdot 2^{n-1}.$$

Noticing that $\binom{u}{0}$ can only take two different values and that it limits the possible values for $\binom{u}{01}$, it remains to enumerate all possibilities. Let us add the fact that we are not able to precisely compute $b_{\mathbf{p}}^{(2)}$ but it seems that this function is not 2-regular (we refer the reader to [11] for a detailed description of 2-regular sequences), which could be considered as quite surprising since $\mathbf{p}$ is a 2-automatic sequence[3]. Let us mention that this conjecture is similar to the one from Anna Frid in [46]: her computational experiments suggest that, as for the 2-binomial complexity, the prefix palindromic length of the period-doubling word is not regular.

Concerning $b_{\mathbf{p}}^{(3)}$, it seems that a similar result holds.

Conjecture 4.5.2. *We have*

$$b_{\mathbf{p}}^{(3)}(2^n + r) < p_{\mathbf{p}}(2^n + r)$$

for $r \in [2^n - 2] \backslash \{1\}$ *when* $n \geq 4$ *is even and for* $r \in [2^n - 3] \backslash \{1, 2\}$ *when* $n \geq 3$ *is odd. Furthermore* $b_{\mathbf{p}}^{(3)}(2^n + r) = p_{\mathbf{p}}(2^n + r)$ *for the remaining residues and* $b_{\mathbf{p}}^{(3)}(n) < p_{\mathbf{p}}(n)$ *for* $n \leq 8$.

[3]It means that there exists a DFA $M = (\mathcal{A}, \mathcal{A}, \delta, \{0\}, \mathcal{A})$ such that, for any $n \in \mathbb{N}$, the base-2 representation of n is accepted by M and the final state is $\mathbf{p}_n$.

At this point the period-doubling word seems interesting since its 2- and 3-binomial complexities behave differently from what we already knew. But we then discovered with a computer that

$$b_{\mathbf{p}}^{(4)}(n) = p_{\mathbf{p}}(n)$$

for all $n \leq 128$ and we thus conjecture the equality for all $n \in \mathbb{N}$. It behaves completely differently from $b_{\mathbf{p}}^{(2)}$ and $b_{\mathbf{p}}^{(3)}$.

It thus leads to the following naive question: for any (purely morphic) infinite word $\mathbf{w}$ such that $b_{\mathbf{w}}^{(2)}$ is not bounded, does there always exist $K \in \mathbb{N}$ such that

$$b_{\mathbf{w}}^{(K)} = p_{\mathbf{w}}$$

(and thus $b_{\mathbf{w}}^{(k)} = p_{\mathbf{w}}$ for all $k \geq K$) or, conversely, can we find a purely morphic infinite word $\mathbf{w}$ (with $b_{\mathbf{w}}^{(2)}$ unbounded) for which

$$b_{\mathbf{w}}^{(k)} < p_{\mathbf{w}}$$

for all $k \in \mathbb{N}$?

5 | Reconstructing words from their binomial coefficients

In this last chapter we are not interested directly in the k-binomial equivalence or complexity, but we study part of the famous *reconstruction problem* of words from their subwords.

The general scheme for a so-called reconstruction problem is the following: given a sufficient amount of information about substructures of a hidden discrete structure, can one uniquely determine this structure? In particular, what are the fragments about the structure needed to recover it all. For instance, a square matrix of size at least 5 can be reconstructed from its principal minors given in any order [81].

In graph theory, given some subgraphs of a graph (these subgraphs may share some common vertices and edges), can one uniquely rebuild the original graph? Given a finite undirected graph $G = (V, E)$ with n vertices, consider the multiset made of the n induced subgraphs of G obtained by deleting exactly one vertex from G. In particular, one knows how many isomorphic subgraphs of a given class appear. Two graphs leading to the same multiset (generally called a *deck*) are said to be *hypomorphic*. A conjecture due to Kelly and Ulam states that two hypomorphic graphs with at least three vertices are isomorphic [60, 92]. A similar conjecture in terms of edge-deleted subgraphs has been proposed by Harary [52]. These conjectures are known to hold true for several families of graphs.

A finite word can be seen as an edge- or vertex-labelled linear tree. So variants of the graph reconstruction problem can be considered and are of independent interest. Participants of the Oberwolfach meeting on Combinatorics on Words in 2010 [14] gave a list of 18 important open problems in the field. Amongst them, the twelfth problem is stated as *reconstruction from subwords of given length*. Recall from Definition 1.2.1 that the k-deck of a word is the multiset of all its subwords of length k.

Definition 5.0.1. Let k, n be natural numbers. Words of length n over a given alphabet are said to be *k-reconstructible* whenever the k-deck uniquely determines any word of length n.

The challenge is to determine the function $f(n) = k$ where k is the least integer

for which words of length n are k-reconstructible. This problem has been studied by several authors and one of the first trace goes back to 1973 [56]. Results in that direction have been obtained by M.-P. Schützenberger (with the so-called *Schützenberger's Guessing game*) and L. Simon [118]. They show that words of length n sharing the same multiset of subwords of length up to $\lfloor n/2 \rfloor + 1$ are the same. Consequently, words of length n are $(\lfloor n/2 \rfloor + 1)$-reconstructible. In [62], this upper bound has been improved: Krasikov and Roditty have shown that words of length n are k-reconstructible for $k \geq \lfloor 16\sqrt{n}/7 \rfloor + 5$. On the other hand Dudik and Schulmann [36] provide a lower bound: if words of length n are k-reconstructible, then $k \geq 3^{(\sqrt{2/3}-o(1))\log_3^{1/2} n}$. Bounds were also considered in [80]. Algorithmic complexity of the reconstruction problem is discussed, for instance, in [35]. Note that the different types of reconstruction problems have application in phylogenetic networks, see, e.g., [123], or in the context of molecular genetics [39] and coding theory [73].

Another motivation, close to combinatorics on words, stems from the study of k-binomial equivalence of finite words and k-binomial complexity of infinite words as we did in this thesis. Given two words x and y of the same length, one can address the following problem: decide whether or not x and y are k-binomially equivalent? A polynomial time decision algorithm based on automata and a probabilistic algorithm have been addressed in [44]. A variation of our work would be to find, given k and n, a minimal set of subwords for which the knowledge of the number of occurrences in x and y permits to decide k-binomial equivalence.

Over an alphabet of size q, there are q^k pairwise distinct length-k words. If we relax the requirement of only considering subwords of the same length, another interesting question is to look for a minimal (in terms of cardinality) multiset of subwords to reconstruct entirely a word. The general problem addressed in this chapter is therefore the following one.

Problem 5.0.2. *Let $\mathcal{A}$ be a given alphabet and n a natural number. We want to reconstruct a hidden word $u \in \mathcal{A}^n$. To that aim, we are allowed to pick a word v_i and ask questions of the type "What is the value of $\binom{u}{v_i}$?". Based on the answers to questions related to $\binom{u}{v_1}, \ldots, \binom{u}{v_i}$, we can decide which will be the next question (i.e., decide which word will be v_{i+1}). We want to have the shortest sequence $(v_1, \ldots, v_k)$ uniquely determining u by knowing the values of $\binom{u}{v_1}, \ldots, \binom{u}{v_k}$.*

We naturally look for a value of k less than the upper bound for k-reconstructibility. We consider an alphabet equipped with a total order on the letters. Words of the form $a^n b$ with letters $a < b$ and $n \in \mathbb{N}_0$ are a special form of Lyndon words, the so-called *right-bounded-block* words.

We consider the reconstruction problem from the information given by the occurrences of right-bounded-block words as subwords of a word of length n. In Sec-

tion 5.2 we show how to reconstruct a word uniquely from $m+1$ binomial coefficients of right-bounded-block words where m is the minimum number of occurrences of a and b in the word. We also prove that this is less than the upper bound given in [62]. In Section 5.3 we reduce the problem for arbitrary finite alphabets $\{1,\dots,q\}$ to the binary case. Here we show that at most $\sum_{i=1}^{q-1}|w|_i\,(q-i+1) \leq q|w|$ binomial coefficients suffice to uniquely reconstruct w with $|w|_i$ being the number of occurrences of letter i in w. Again, we compare this bound to the best known one for the classical reconstruction problem (from words of a given length), and in the last subsection of the chapter we discuss the algorithmic complexity of our solution. Some results are taken from [41], written by Pamela Fleischmann, Marie Lejeune, Florin Manea, Dirk Nowotka and Michel Rigo, and published in its proceeding version after its acceptance in *Developments in Language Theory* 2020. The extended version was accepted in the International Journal of the Foundation of Computer Science and will be published soon. Subsections 5.3.1 and 5.3.3 differ from the initial papers and were rewritten in an aim to better suit to our particular problem.

Contents

5.1 Presentation of the problem

A word $u \in \mathcal{A}^*$ is called *uniquely reconstructible* (or *uniquely determined*) by the set $S \subset \mathcal{A}^*$ if for all words $v \in \mathcal{A}^*\backslash\{u\}$, there exists $s \in S$ such that $\binom{u}{s} \neq \binom{v}{s}$.

Consider $S = \{ab, ba\}$. Then $u = abba$ is not uniquely reconstructible by S since $\left(\binom{u}{ab}, \binom{u}{ba}\right) = (2,2)$ is also the 2-vector of binomial coefficients of $baab$. On the other hand $S = \{a, ab, abb\}$ reconstructs u uniquely. The following remark gives immediate results for binary alphabets.

Remark 5.1.1. Let $\mathcal{A} = \{a, b\}$ and $u \in \mathcal{A}^n$. If $|u|_a \in \{0, n\}$ then u contains either only b or a and by the given length n of u, u is uniquely determined by $S = \{a\}$. This fact is in particular an equivalence: $u \in \mathcal{A}^n$ can be uniquely determined by $\{a\}$ if and only if $|u|_a \in \{0, n\}$. If $|u|_a \in \{1, n-1\}$, u is not uniquely determined by $\{a\}$ as witnessed by ab and ba for $n = 2$. It is immediately clear that the additional information $\binom{u}{ab}$ leads to unique determinism of u.

We define a query of the following form: $Q(u, v)$ stands for "What is the value of $\binom{u}{v}$?". Let us fix $\mathcal{A}$ and $n \in \mathbb{N}$. We are interested, in this chapter, in the minimal number of queries we have to ask, in a sequential way, to be sure to uniquely determine u. By sequential, we mean that the answers to queries $Q(u, v_1), \ldots, Q(u, v_i)$ can influence the choice of v_{i+1} for the next query $Q(u, v_{i+1})$.

5.2 Binary case

As always, we first ask the question on binary alphabets. Hence set $\mathcal{A} = \{a, b\}$ all along this section.

5.2.1 An algorithm involving right-bounded-block words

In this subsection we present a method to reconstruct a binary word uniquely from binomial coefficients of right-bounded-block words. Let $n \in \mathbb{N}$ be a natural number and $u \in \{a, b\}^n$ a word. Since the word length n is assumed to be known, $|u|_a$ is known if $|u|_b$ is given and vice versa.

Moreover we assume without loss of generality that $|u|_a \leq |u|_b$ and that $|u|_b$, and thus $|u|_a$, is known (otherwise substitute each a by b and each b by a, apply the following reconstruction method and revert the substitution). This implies that u is of the form:

$$b^{s_1} a b^{s_2} \ldots b^{s_{|u|_a}} a b^{s_{|u|_a+1}} \tag{5.1}$$

for $s_i \in \mathbb{N}_0$ and $i \in [|u|_a + 1]$ with

$$\sum_{i \in [|u|_a+1]} s_i = n - |u|_a = |u|_b$$

and thus we get for $\ell \in [|u|_a]_0$,

$$\binom{u}{a^\ell b} = \sum_{i=\ell+1}^{|u|_a+1} \binom{i-1}{\ell} s_i. \tag{5.2}$$

Remark 5.2.1. Notice that for fixed $\ell \in [|u|_a]_0$ and $c_i = \binom{i-1}{\ell}$ for $i \in [|u|_a + 1] \setminus [\ell]$, we have $c_i < c_{i+1}$ and especially $c_{\ell+1} = 1$ and $c_{\ell+2} = \ell + 1$.

Equation (5.2) shows that reconstructing a word uniquely from binomial coefficients of right-bounded-block words equates to solve a system of Diophantine equations. The knowledge of $\binom{u}{b}, \dots, \binom{u}{a^\ell b}$ provides $\ell+1$ equations. If the equation of $\binom{u}{a^\ell b}$ has a unique solution for $\{s_{\ell+1}, \dots, s_{|u|_a+1}\}$ (in this case we say, by language abuse, that $\binom{u}{a^\ell b}$ is *unique*), then the system in row echelon form has a unique solution and thus the binary word is uniquely reconstructible. Notice that $\binom{u}{a^{|u|_a} b}$ is always unique since Equation (5.2) can be rewritten $\binom{u}{a^{|u|_a} b} = s_{|u|_a+1}$ for $\ell = u_a$.

Example 5.2.2. Consider $n = 10$ and an unknown word $u \in \mathcal{A}^n$. If we first ask the query $Q(u, b)$, we obtain $|u|_b = 6$ and thus $|u|_a = 4$. This leads to

$$u = b^{s_1} a b^{s_2} a b^{s_3} a b^{s_4} a b^{s_5}$$

with $s_1 + s_2 + s_3 + s_4 + s_5 = 6$. If we now add the information that $\binom{u}{ab} = 4$, asking $Q(u, ab)$, we get

$$s_2 + 2s_3 + 3s_4 + 4s_5 = 4.$$

Unknowns s_i are not uniquely determined. Adding the fact that $\binom{u}{aab}$ is equal to 2 with the query $Q(u, aab)$, we get

$$s_3 + 3s_4 + 6s_5 = 2$$

and thus $s_4 = s_5 = 0, s_3 = 2$. Injecting it in the previous equations gives us $s_2 = 0$ and $s_1 = 4$. Hence the word $u = bbbbaabbaa$ is uniquely determined by asking three queries, concerning the set of right-bounded-block words $\{b, ab, aab\}$.

Let u be an unknown word with $|u|_a \leq |u|_b$. We denote by j_u the minimal $j \in \mathbb{N}_0$ such that u is uniquely determined from $\{b, ab, a^2b, \dots, a^j b\}$.

Example 5.2.3. In Table 5.1 we give the values of j_u for every binary word u of length 5.

Theorem 5.2.4. *Let $u \in \mathcal{A}^n$ be an unknown word with $|u|_a \leq |u|_b$. Let $j \in [|u|_a]_0$. If $\binom{u}{a^j b}$ is unique, then $j_u \leq j$, i.e., u is uniquely determined by $\{b, ab, a^2b, \dots, a^j b\}$.*

Proof. If $\binom{u}{a^j b}$ is unique, coefficients $s_{j+1}, \dots, s_{|u|_a+1}$ are uniquely determined from

$$\binom{u}{a^\ell b} = \sum_{i=\ell+1}^{|u|_a+1} \binom{i-1}{\ell} s_i.$$

Substituting backwards the known values in the first j Equations (5.2) (for $\ell = j-1, j-2, \dots, 0$), we can successively obtain the unique solutions for $s_j, \dots, s_1$. □

Corollary 5.2.5. *Let $u \in \mathcal{A}^*$ be an unknown word such that $|u|_a \leq |u|_b$. The value j_u is exactly the minimal $j \in \mathbb{N}_0$ such that $\binom{u}{a^j b}$ is unique. Hence $j_u \leq |u|_a$.*

j_u	0	1		2	
	aaaaa	*aaaab*	*abbbb*	*aabba*	*abbab*
	bbbbb	*aaaba*	*babbb*	*abaab*	*abbba*
		aabaa	*bbabb*	*ababa*	*baabb*
		abaaa	*bbbab*	*abbaa*	*babab*
		baaaa	*bbbba*	*baaab*	*babba*
				baaba	*bbaab*
		aaabb	*aabbb*		
		aabab	*ababb*		
		babaa	*bbaba*		
		bbaaa	*bbbaa*		

Table 5.1: Values of j_u for words u of length 5.

Theorem 5.2.6. *Let $n \in \mathbb{N}$ and $u \in \mathcal{A}^n$ be an unknown word. We can determine u in a unique way by asking a maximum of $\lfloor \frac{n}{2} \rfloor + 1$ queries.*

Proof. The first query to ask is $Q(u, a)$ (or, in an equivalent way, $Q(u, b)$) to determine the 1-deck of u. If $|u|_a \leq |u|_b$, Corollary 5.2.5 ensures that u is determined after asking j_u more queries, where $j_u \leq |u|_a$. In this case, $|u|_a \leq \lfloor \frac{n}{2} \rfloor$. If $|u|_b \leq |u|_a$, inverting the role of a and b, we can claim that u is reconstructed after a maximal number of $|u|_b$ queries, with $|u|_b \leq \lfloor \frac{n}{2} \rfloor$. □

Let us now study particular cases where less queries are needed. First note that for any $\ell \in \mathbb{N}$ and any $u \in \mathcal{A}^*$ the value of $\binom{u}{a^\ell b}$ is in the set $\left[\binom{|u|_a}{\ell}|u|_b\right]_0$. We start by a combinatorial lemma.

Lemma 5.2.7. *Let $n \in \mathbb{N}$, $k \in [n]_0$, $j \in [k+1]$ and $c_1, \ldots, c_{k+1}, s_1, \ldots, s_{k+1} \in \mathbb{N}_0$ be such that $c_i < c_{i+1}$ for $i \in [k]$ and $\sum_{i=1}^{k+1} s_i = n - k$. The sum*

$$\sum_{i=j}^{k+1} c_i s_i$$

is maximal if and only if $s_{k+1} = n - k$, and consequently $s_i = 0$ for all $i \in [k]$.

Proof. The case $k = 0$ is trivial. Consider the case $n = k$, i.e., $\sum_{i=1}^{k+1} s_i = 0$. This implies immediately $s_i = 0$ for all $i \in [k+1]$ and the equivalence holds. Assume for the rest of the proof that $0 < k < n$. If $s_{k+1} = n - k$, then $s_i = 0$ for all $i \leq k$ and

$$\sum_{i=j}^{k+1} c_i s_i = c_{k+1}(n - k).$$

Let us assume that the maximal value for $\sum_{i=j}^{k+1} c_i s_i$ can be obtained in another way and that there exist $s'_1, \dots, s'_{k+1} \in \mathbb{N}_0$, $\ell \in [n-k]$ such that $\sum_{i=1}^{k+1} s'_i = n-k$ and $s'_{k+1} = n-k-\ell$. Thus

$$c_{k+1}(n-k) \leq \sum_{i=j}^{k+1} c_i s'_i = \left(\sum_{i=j}^{k} c_i s'_i \right) + c_{k+1}(n-k-\ell).$$

This implies $\sum_{i=j}^{k} c_i s'_i \geq c_{k+1}\ell$. Since the coefficients are strictly increasing we get

$$\sum_{i=j}^{k} c_i s'_i \leq c_k \sum_{i=j}^{k} s'_i < c_{k+1}\ell,$$

hence the contradiction. □

Proposition 5.2.8. *Let u be an unknown word of $\mathcal{A}^n$ and let ℓ be in the set $[|u|_a - 1]_0$. If*

$$\binom{u}{a^\ell b} \in [\ell]_0 \cup \left\{ \binom{|u|_a - 1}{\ell} r + \binom{|u|_a}{\ell} (|u|_b - r) : r \in [|u|_b]_0 \right\},$$

then $\binom{u}{a^\ell b}$ is unique.

Proof. Let us write u in the form of Equation (5.1). We will make an intensive use of (5.2); set, for all $i \in [|u|_a + 1] \backslash [\ell]$,

$$c_i = \binom{i-1}{\ell}.$$

Consider firstly $\binom{u}{a^\ell b} \in [\ell]_0$. By Remark 5.2.1 we have $c_{\ell+1} = 1$, $c_{\ell+2} = \ell + 1$ and we know that $c_i < c_{i+1}$, hence we obtain immediately $s_i = 0$ for $i \in [|u|_a + 1] \backslash [\ell + 1]$. The only solution of Equation 5.2 is thus $s_{\ell+1} = \binom{u}{a^\ell b}$ and the claim is proven.

Let now be $r \in [|u|_b]_0$, assume that

$$\binom{u}{a^\ell b} = \binom{|u|_a - 1}{\ell} r + \binom{|u|_a}{\ell} (|u|_b - r).$$

If $r = 0$, the value of $\binom{u}{a^\ell b}$ is maximal, $s_{|u|_a+1} = |u|_b$ and $s_i = 0$ for $i \in [|u|_a]_0$ is the only possibility, by the previous lemma.

Assume now, by contradiction, that $\binom{u}{a^\ell b}$ is not unique and that r is positive. This last fact implies $s_{|u|_a+1} < |u|_b - r$. Assume that $s_{|u|_a+1} = |u|_b - r'$ for $r' > r$. Thus there exists $x \in \mathbb{N}$ such that

$$\binom{u}{a^\ell b} = \binom{|u|_a}{\ell} s_{|u|_a+1} + x,$$

$$\text{i.e., } x = \frac{(|u|_a - 1)!(|u|_a r' - \ell r)}{\ell!(|u|_a - \ell)!}.$$

On the one hand, since $r' > r$ we have

$$x > \frac{(|u|_a - 1)!(|u|_a r' - \ell r')}{\ell!(|u|_a - \ell)!} \tag{5.3}$$

but on the other hand,

$$x \leq s_{|u|_a} \binom{|u|_a - 1}{\ell},$$

and because $s_{|u|_a} + s_{|u|_a+1} \leq |u|_b$,

$$x \leq r' \binom{|u|_a - 1}{\ell} = \frac{(|u|_a - 1)!(|u|_a r' - \ell r')}{\ell!(|u|_a - \ell)!}. \tag{5.4}$$

A contradiction raises from Equations (5.3) and (5.4). □

Proposition 5.2.9. *The word $u \in \mathcal{A}^n$ is uniquely determined by $|u|_b$ and $\binom{u}{ab}$ if and only if one of the following occurs:*

- *$|u|_b = 0$ or $|u|_b = n$ (and then obviously $\binom{u}{ab} = 0$);*
- *$|u|_b = 1$ or $|u|_b = n - 1$ and $\binom{u}{ab}$ is arbitrary in $[n-1]_0$, or;*
- *$|u|_b \in [n-2]\backslash\{1\}$ and $\binom{u}{ab} \in \{0, 1, |u|_a|u|_b - 1, |u|_a|u|_b\}$.*

Proof. Let us first prove that u is uniquely determined in these cases. It is obvious if $|u|_b = 0$ or $|u|_b = n$ since the word is composed of the same letter repeated n times.

If $|u|_b = n - 1$, then

$$u = b^{s_1} a b^{n-1-s_1} \quad \text{and} \quad \binom{u}{ab} = n - 1 - s_1.$$

Therefore u is uniquely determined. If $|u|_b = 1$, then

$$u = b^{s_1} a b^{s_2} \cdots a b^{s_n}$$

with exactly one of the s_i being non zero and, in fact, equal to one. We have

$$\binom{u}{ab} = \sum_{i=2}^{n} (i-1) s_i$$

and, if $\binom{u}{ab}$ is given, then $s_{\binom{u}{ab}+1} = 1$ is the only non-zero variable.

Consider now $|u|_a \in [n-2]\backslash\{1\}$, i.e.,

$$u = b^{s_1} a b^{s_2} \cdots b^{s_{|u|_a}} a b^{s_{|u|_a+1}}.$$

Thus $\binom{u}{ab} = 0$ implies $s_1 = |u|_b$ and $s_2 = 0, \ldots, s_{|u|_a+1} = 0$ while $\binom{u}{ab} = 1$ implies $s_2 = 1$, $s_1 = |u|_b - 1$ and $s_3 = 0, \ldots, s_{|u|_a+1} = 0$.

By Lemma 5.2.7, we know that $\binom{u}{ab}$ is maximal if and only if $s_{|u|_a+1} = |u|_b$ and all the other s_i are equal to zero. In that case, the value of the sum equals $|u|_a|u|_b$. Therefore, if $\binom{u}{ab} = |u|_a|u|_b$, the word u is uniquely determined. Finally, if $\binom{u}{ab} = |u|_a|u|_b - 1$, we must have $s_{|u|_a+1} \leq |u|_b - 1$. If we choose $s_{|u|_a+1} = |u|_b - 1$, it remains that

$$\sum_{i=1}^{|u|_a} s_i = 1 \quad \text{and} \quad \sum_{i=2}^{|u|_a} (i-1)s_i = |u|_a - 1.$$

We must have $s_{|u|_a} = 1$ and the other ones equal to zero. In fact, choosing $s_{|u|_a+1} = |u|_b - 1$ is the only possibility: if otherwise $s_{|u|_a+1} = |u|_b - r$ with $r > 1$, we obtain

$$\sum_{i=2}^{|u|_a} (i-1)s_i \geq r|u|_a - 1$$

with $\sum_{i=1}^{|u|_a} s_i = r$. It is easy to check with Lemma 5.2.7 that these conditions are incompatible.

We now need to prove that u cannot be uniquely determined if

$$|u|_a \in [n-2]\backslash\{1\} \quad \text{and} \quad \binom{u}{ab} \in [|u|_a|u|_b - 2]\backslash\{1\}.$$

To this aim we will give two different sets of values for the unknowns s_i. The first decomposition is the greedy one. Let us put

$$s_i = \begin{cases} \left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor & \text{if } i = |u|_a + 1; \\ 1 & \text{if } i = (\binom{u}{ab} \bmod |u|_a) + 1; \\ 0 & \text{otherwise.} \end{cases}$$

Let us finally modify the value of s_1 (which is, at this stage, equal to 0 or 1) by adding the value needed. By $\sum_{i=1}^{|u|_a+1} s_i = n - |u|_a$ we get

$$s_1 \leftarrow s_1 + |u|_b - \left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor - 1.$$

This implies

$$\begin{aligned} \sum_{i=1}^{|u|_a+1} s_i &= 1 + |u|_b - \left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor - 1 + \left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor \\ &= |u|_b \end{aligned}$$

and $s_i \geq 0$ for all i. Moreover we have

$$\begin{aligned} \sum_{i=2}^{|u|_a+1} (i-1)s_i &= \left(\binom{u}{ab} \bmod |u|_a \right) + |u|_a \left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor \\ &= \binom{u}{ab}. \end{aligned}$$

Now we provide a second decomposition for the s_i. First, let us assume that $2 \leq \binom{u}{ab} < |u|_a$. In that case, the greedy algorithm sets

$$s_{\binom{u}{ab}+1} = 1, \quad s_1 = |u|_b - 1 \quad \text{and} \quad s_i = 0 \ \ \forall i \notin \left\{1, \binom{u}{ab} + 1\right\}.$$

Let us now set $s_1 = |u|_b - 2$ and all the other s_i to 0. Then, update

$$s_{\binom{u}{ab}} \leftarrow s_{\binom{u}{ab}} + 1 \quad \text{and} \quad s_2 \leftarrow s_2 + 1$$

(in the case where $\binom{u}{ab} = 2$, s_2 will be equal to 2 after these manipulations). We obtain that the sum in (5.2) is equal to $1 + (\binom{u}{ab} - 1)$ as needed.

Finally, if $\binom{u}{ab} \geq |u|_a$, then $s_{|u|_a+1}$ was non-zero in the greedy decomposition, and the idea is to reduce it of a value of 1. Let us set

$$s_{|u|_a+1} = \left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor - 1 \quad \text{and} \quad s_i = 0 \ \ \forall i \neq |u|_a + 1.$$

Then, let us update some values:

$$s_{(\binom{u}{ab} \bmod |u|_a)+2} \leftarrow s_{(\binom{u}{ab} \bmod |u|_a)+2} + 1 \quad \text{and} \quad s_{|u|_a} \leftarrow s_{|u|_a} + 1$$

if $(\binom{u}{ab} \bmod |u|_a) \neq |u|_a - 1$, and

$$s_{|u|_a} = 2, \quad s_2 = 1$$

otherwise. Finally, set s_1 to the right value, i.e.,

$$s_1 = |u|_b - \sum_{i=2}^{|u|_a+1} s_i.$$

It can be easily checked that, in both cases, $s_1 \geq 0$ (notice that $(\binom{u}{ab} \bmod |u|_a)$ is equal to $|u|_a - 1$ and implies $\left\lfloor \frac{\binom{u}{ab}}{|u|_a} \right\rfloor \leq |u|_b - 2$) and that all s_i sum up to $|u|_b$. Similarly, we can check that $\sum_{i=2}^{|u|_a+1}(i-1)s_i$ is equal to $\binom{u}{ab}$ in both cases.

To sum up, we gave two different decompositions for the s_i in cases where $|u|_a \in [n-2]\backslash\{1\}$ and $\binom{u}{ab} \in [|u|_a(n - |u|_a) - 2]\backslash\{1\}$. That implies that u cannot be uniquely determined in those cases. □

5.2.2 Comparing the number of queries to the classical reconstruction problem

By [62] an upper bound on the number of binomial coefficients to uniquely reconstruct the word $u \in \mathcal{A}^n$ is given by the amount of the binomial coefficients of the

$(\lfloor \frac{16}{7}\sqrt{n} \rfloor + 5)$-deck. Notice that implicitly the full deck is assumed to be known. As proven in Subsection 1.2.1, Lyndon words up to this length suffice. It can be shown using the unweighted version of the Polya's Enumeration Theorem [100] that there are

$$N_{\#\mathcal{A}}(i) := \frac{1}{i}\sum_{d|i}\mu(d)(\#\mathcal{A})^{\frac{i}{d}}$$

Lyndon words of length i over an arbitrary alphabet $\mathcal{A}$, where $\mu : \mathbb{N} \to \{-1, 0, 1\}$ is the classical Möbius function [61]: the value $\mu(n)$ is non-zero if and only if n is square-free. More precisely, if $n = p_1^{\ell_1} \cdots p_k^{\ell_k}$ where $p_1, \ldots, p_k$ are distinct prime numbers and where $\ell_i \in \mathbb{N}$ for all $i \in [k]$, then

$$\mu(n) = \begin{cases} 0 & \text{if there exists } i \in [k] \text{ such that } \ell_i \geq 2; \\ (-1)^k & \text{otherwise.} \end{cases}$$

The function $N_q(i)$ here above is sometimes called the *Moreau's necklace-counting function* [85].

The combination of both results presented in [62, 105] states that, for $n > 6$,

$$\sum_{i=1}^{\lfloor \frac{16}{7}\sqrt{n} \rfloor + 5} \frac{1}{i}\sum_{d|i}\mu(d) \cdot 2^{\frac{i}{d}} \tag{5.5}$$

queries are sufficient for a unique reconstruction of any binary word of length n.

By Proposition 5.2.9 we need exactly one query for $n \leq 3$, two if $n = 4$ and $n - 1$ for $n \in \{5, 6\}$. For $n > 6$ Theorem 5.2.11 shows that we need strictly less queries than (5.5). Let us first give the proof of a combinatorial lemma we will need here and in the case of an arbitrary alphabet. This result is inspired from [40, Lemma 2.4] but their statement and proof were wrong, so we adapt them here.

Lemma 5.2.10. *For all $q \geq 2$ and for all $i \in \mathbb{N}$, we have*

$$N_q(i) \geq \frac{1}{i}\left(q^i - \frac{q^{\frac{i}{2}+1} - 1}{q - 1}\right).$$

Proof. Let us show that

$$\sum_{d|i}\mu(d)q^{\frac{i}{d}} \geq q^i - \frac{q^{\frac{i}{2}+1} - 1}{q - 1}.$$

Starting from the right-hand side we get

$$\begin{aligned} q^i - \frac{q^{\frac{i}{2}+1} - 1}{q - 1} &\leq q^i - \frac{q^{\lfloor \frac{i}{2} \rfloor + 1} - 1}{q - 1} \\ &= q^i - \sum_{d=0}^{\lfloor \frac{i}{2} \rfloor} q^d \end{aligned}$$

Note that all divisors of i that are different from i are less than or equal to $\lfloor \frac{i}{2} \rfloor$. Hence

$$\sum_{d=0}^{\lfloor \frac{i}{2} \rfloor} q^d = \sum_{\substack{d \mid i, \\ d \neq i}} q^d + \sum_{\substack{d=0 \\ d \nmid i}}^{\lfloor \frac{i}{2} \rfloor} q^d$$

and we get, by keeping only the divisors of i,

$$q^i - \frac{q^{\frac{i}{2}+1} - 1}{q-1} \leq q^i + \sum_{\substack{d \mid i, \\ d \neq i}} (-1) q^d \leq \mu(1) q^i + \sum_{\substack{d \mid i, \\ d \neq i}} \mu\left(\frac{i}{d}\right) q^d.$$

Hence we finally conclude since

$$\mu(1) q^i + \sum_{\substack{d \mid i, \\ d \neq i}} \mu\left(\frac{i}{d}\right) q^d = \sum_{d \mid i} \mu\left(\frac{i}{d}\right) q^d = \sum_{d \mid i} \mu(d) q^{\frac{i}{d}}.$$

□

We can now show that we need less queries than (5.5).

Theorem 5.2.11. *Let $n > 6$ and let $u \in \mathcal{A}^n$. We can reconstruct u uniquely with at most $\lfloor \frac{n}{2} \rfloor + 1$ queries, which is strictly smaller than* (5.5).

Proof. By the previous lemma, we have, for all $i \geq 4$,

$$\begin{aligned} N_2(i) \geq & \frac{1}{i} \left(2^i - \frac{2^{\frac{i}{2}+1} - 1}{2-1} \right) \\ & = \frac{1}{i} \left(2^i - 2^{\frac{i}{2}+1} + 1 \right) \\ & = \frac{1}{i} \left(2^{\frac{i}{2}+1} (2^{\frac{i}{2}-1} - 1) + 1 \right) \\ \geq & \frac{2^{\frac{i}{2}+1}}{i}. \end{aligned}$$

For dealing with the values of i less than 4, note that

$$\sum_{i=1}^{5} N_2(i) \geq \sum_{i=1}^{5} \frac{2^{\frac{i}{2}+1}}{i},$$

since the left-hand side equals

$$2^1 + \frac{1}{2}(2^2 - 2) + \frac{1}{3}(2^3 - 2) + \frac{1}{4}(2^4 - 2^2) + \frac{1}{5}(2^5 - 2) = 2 + 1 + 2 + 3 + 6 = 14$$

while the right-hand side is equal to

$$2\sqrt{2}+2+\frac{4\sqrt{2}}{3}+2+\frac{8\sqrt{2}}{5}\leq 2\sqrt{2}+2+2+2+3\leq 13.$$

For more convenience set $\alpha(x)=\frac{16}{7}\sqrt{x}+5$ and $\alpha^{\lfloor\cdot\rfloor}(x)=\lfloor\frac{16}{7}\sqrt{x}\rfloor+5$ for all $x>0$. We obtain, since $\alpha^{\lfloor\cdot\rfloor}(n)\geq 5$ and since $\alpha^{\lfloor\cdot\rfloor}(n)+1\geq\alpha(n)$,

$$\begin{aligned}\sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} N_2(i) &\geq \sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} \frac{2^{\frac{i}{2}+1}}{i}\\ &\geq \frac{2}{\alpha(n)}\,\frac{\sqrt{2}^{\alpha(n)}-1}{\sqrt{2}-1}\\ &\geq \frac{1}{\alpha(n)}\,\frac{\sqrt{2}^{\alpha(n)}-1}{\sqrt{2}-1}.\end{aligned}$$

We want to show that this quantity is at least equal to $\frac{n+1}{2}$. Let us define

$$f(x)=\frac{1}{\alpha(x)}\,\frac{\sqrt{2}^{\alpha(x)}-1}{\sqrt{2}-1}-\frac{x+1}{2}$$

for all $x>0$, which is the continuous extension on $\mathbb{R}^+$ of the quantity we are interested in. Let us formally show that $f(x)>0$ for all $x\geq 5$. We have $f(5)\approx 4.7$ and we will in fact show that f is increasing for $x\geq 5$. Since this function is differentiable, we get

$$f'(x)=\frac{1}{\alpha(x)}\,\sqrt{2}^{\alpha(x)}\,\frac{\ln(\sqrt{2})}{\sqrt{2}-1}\,\frac{8}{7\sqrt{x}}-\frac{1}{\alpha(x)^2}\,\frac{8}{7\sqrt{x}}\,\frac{\sqrt{2}^{\alpha(x)}-1}{\sqrt{2}-1}-\frac{1}{2}.$$

Since $\sqrt{x}=\frac{7\alpha(x)-35}{16}$ and by reducing to the common denominator that is positive, we have to show that

$$\begin{aligned}&2\alpha(x)\sqrt{2}^{\alpha(x)}128\ln(\sqrt{2})-256(\sqrt{2}^{\alpha(x)}-1)-7(7\alpha(x)-35)\alpha(x)^2(\sqrt{2}-1)\\ &=\sqrt{2}^{\alpha(x)}(128\ln(2)\alpha(x)-256)+256-49\alpha(x)^3(\sqrt{2}-1)+245\alpha(x)^2(\sqrt{2}-1)\end{aligned}\quad(5.6)$$

is strictly positive. Because $\alpha(x)\geq 7$ for $x\geq 1$ and $\ln(2)>0.69308$, the value of (5.6) is greater than

$$\sqrt{2}^{\alpha(x)}365+256-49\alpha(x)^3(\sqrt{2}-1)+245\alpha(x)^2(\sqrt{2}-1).$$

Set

$$g(y)=\sqrt{2}^{y}365+256-49y^3(\sqrt{2}-1)+245y^2(\sqrt{2}-1);$$

we will show that $g(y)$ is positive for all $y\geq 10.05$, which means that $f'(x)$ is positive for all x such that $\alpha(x)\geq 10.05$, i.e., for all $x\geq 5$.

We have

$$\begin{aligned} g'(y) &= 365\sqrt{2}^y \ln(\sqrt{2}) - 147(\sqrt{2}-1)y^2 + 490(\sqrt{2}-1)y, \\ g''(y) &= 365\sqrt{2}^y (\ln(\sqrt{2}))^2 - 294(\sqrt{2}-1)y + 490(\sqrt{2}-1), \\ g'''(y) &= 365\sqrt{2}^y (\ln(\sqrt{2}))^3 - 294(\sqrt{2}-1), \end{aligned}$$

and $g'''(7) > 50$, $g''(8.5) > 2$, $g'(10.05) > 8$ and finally $g(10.05) > 1787$. Since $g'''(y)$ is increasing and positive in 7, $g''(y)$ is increasing for $y \geq 7$. Therefore $g'(y)$ is increasing for $y \geq 8.5$ and finally $g(y)$ is increasing for $y \geq 10.05$ and positive. □

5.3 Extension to a general alphabet

In this section we address the problem of reconstructing words over arbitrary alphabets from their subwords. Let $\mathcal{A} = \{a_1, \ldots, a_q\}$ be an alphabet equipped with the ordering $a_i < a_j$ for $1 \leq i < j \leq q$.

Let $\{a, b\}$ be a binary subalphabet of $\mathcal{A}$. The *binary projection of u over the subalphabet* $\{a, b\}$ is the word $\pi_{a,b}(u) \in \{a, b\}^*$, where $\pi_{a,b}$ is the following morphism:

$$\pi_{a,b} : \begin{cases} a & \mapsto & a; \\ b & \mapsto & b; \\ c & \mapsto & \varepsilon \text{ for all } c \notin \{a, b\}. \end{cases}$$

For reconstructing a word $u \in \mathcal{A}^n$, we first determine its binary projections over all subalphabets of size 2. It can be done easily, following the results of Section 5.2. We then would like to obtain an algorithm whose aim is to reconstruct u uniquely from its set of binary projections. Such a process exists and is presented in Subsection 5.3.1. We then discuss the number of queries in Subsection 5.3.2 and the total complexity of the algorithm in Subsection 5.3.3.

5.3.1 Reconstructing a word from its binary projections

The algorithm we are going to describe involves two notions that we are first going to present.

Definition 5.3.1. Let $u^{(1)}, \ldots, u^{(\ell)}$ be words of $\mathcal{A}^*$ and let $K = (k_a)_{a \in \mathcal{A}}$ be a sequence of natural numbers. A *K-marking of $u^{(1)}, \ldots, u^{(\ell)}$* is an application ψ defined on

$$\{(j, i) \in \mathbb{N} \times \mathbb{N} : j \in [\ell], i \in [|u^{(j)}]\},$$

whose image is in $\mathbb{N}$ and such that, for all $j \in [\ell], i, m \in [|u^{(j)}|]$,

- if $u_i^{(j)} = a$, then $\psi(j, i) \leq k_a$;

- if $i < m$ and $u_i^{(j)} = u_m^{(j)}$, then $\psi(j,i) < \psi(j,m)$.

We denote by $\Psi_K(u^{(1)},\dots,u^{(\ell)})$ (or Ψ_K if it is clear from the context) the set of all K-markings of $u^{(1)},\dots,u^{(\ell)}$.

A K-marking can be seen as a map from letters of $u^{(1)},\dots,u^{(\ell)}$ to $\mathbb{N}$.

Note that, for a fixed sequence K, a K-marking of $u^{(1)},\dots,u^{(\ell)}$ only exists if, for all $a \in \mathcal{A}$,

$$k_a \geq \max_{j\in[\ell]} |u^{(j)}|_a. \tag{5.7}$$

If $u^{(1)},\dots,u^{(\ell)}$ are given, we say that a sequence K is *minimal* if all inequalities here above are equalities. We denote the minimal sequence K by K_{min}. In what follows we will only consider sequences K satisfying all Inequalities (5.7).

Example 5.3.2. Let us take $\mathcal{A} = \{a,b,c\}$, $K = (k_a = 2, k_b = 3, k_c = 2)$, $u^{(1)} = bcab$ and $u^{(2)} = aba$. Any K-marking of $u^{(1)}, u^{(2)}$ is defined on

$$\{(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),(2,3)\}$$

and since $u_1^{(1)} = u_4^{(1)}$, we have $\psi(1,1) < \psi(1,4)$. For the same reason, $\psi(2,1) < \psi(2,3)$. The fact that, for any $a \in \mathcal{A}$, $u_i^{(j)} = a$ implies $\psi(j,i) \leq k_a$ leaves us with few K-markings. Here is one of them:

j	1				2		
i	1	2	3	4	1	2	3
$u_i^{(j)}$	b	c	a	b	a	b	a
$\psi(j,i)$	1	2	1	3	1	1	2

Moreover, $K_{min} = (k_a = 2, k_b = 2, k_c = 1)$, which leaves us with only 4 possible K_{min}-markings:

j	1				2		
i	1	2	3	4	1	2	3
$u_i^{(j)}$	b	c	a	b	a	b	a
$\psi_1(j,i)$	1	1	1	2	1	1	2
$\psi_2(j,i)$	1	1	1	2	1	2	2
$\psi_3(j,i)$	1	1	2	2	1	1	2
$\psi_4(j,i)$	1	1	2	2	1	2	2

Here comes the second definition.

Definition 5.3.3. Let $G = (V, E)$ be a directed graph. A *topological sort* (or *topological ordering*) of G is a total ordering $v_1 < \cdots < v_n$ of the vertices of G, such that if $(v_i, v_j) \in E$, then $i < j$.

We call a *topological path*, or *path associated to the topological sort* the word $v_1 \cdots v_n \in V^n$.

Here are some folklore facts, see [12] for example.

Proposition 5.3.4. *Let G be a directed graph.*

1. *G admits a topological sort if and only if it is acyclic.*
2. *If $v_1 < \cdots < v_n$ is a topological sort of G, then for any $i \in [n-1]$ such that (v_i, v_{i+1}) is not an edge of G, then the sorting $v_1 < \cdots < v_{i-1} < v_{i+1} < v_i < v_{i+2} < \cdots < v_n$ obtained by exchanging v_i and v_{i+1} is still a topological sort of G.*
3. *If G is acyclic, its topological sort is unique if and only if G has a hamiltonian path*[1].

We denote by $\mathcal{P}(G)$ the set of all paths associated to a topological sort of G.

We now define a graph associated to a given K-marking, for which we are going to study its topological sorts.

Definition 5.3.5. Let $u^{(1)}, \ldots, u^{(\ell)}$ be words of $\mathcal{A}^*$, K be a sequence of $\mathbb{N}^{\mathcal{A}}$ and ψ be a K-marking of $u^{(1)}, \ldots, u^{(\ell)}$. The *graph associated to the K-marking ψ*, denoted G_ψ, is defined as follows. Its set of vertices is

$$\{(a)_i : a \in \mathcal{A}, i \in [k_a]\}.$$

Let $a, b \in \mathcal{A}$ and $m \in [k_a]$, $n \in [k_b]$. The pair of vertices $((a)_m, (b)_n)$ is a directed edge of G_ψ if

- $a = b$ and $n = m + 1$, or;
- there exists $j \in [\ell]$ and $i \in [|u^{(j)}| - 1]$ such that $u_i^{(j)} = a$, $u_{i+1}^{(j)} = b$, $\psi(j, i) = m$ and $\psi(j, i+1) = n$.

Example 5.3.6. Let us continue the previous example. For $K = (2, 3, 2)$ and the given K-marking ψ, the associated graph is given in Figure 5.1. This graph does not admit a topological sort, since there is a cycle involving vertices $(a)_1$,$(b)_1$ and $(c)_2$. Changing the value of $\psi(2, 2)$ from 1 to 2 gives a new K-marking ψ' for which the graph $G_{\psi'}$ is acyclic (see Figure 5.2).

Note that $(b)_1$ and $(c)_1$ are two vertices having no incoming edge. For this reason, they have to come first in any topological sort of $G_{\psi'}$. Since $(a)_2$ and $(b)_3$ have

[1] Recall that, in a graph, an *hamiltonian path* is a path going through every vertex once.

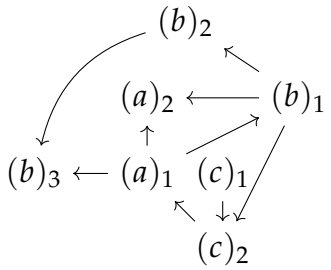


Figure 5.1: Graph G_ψ associated to the K-marking ψ.

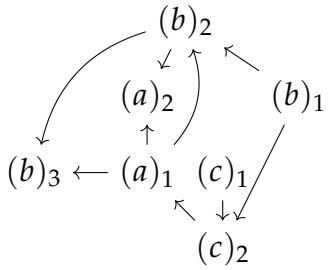


Figure 5.2: Graph $G_{\psi'}$ associated to the K-marking ψ'.

no outgoing edge, they have to come at the end of any topological sort. Moreover, there is a path $(c)_2 \to (a)_1 \to (b)_2$. Hence, there are exactly four topological paths associated to $G_{\psi'}$, given by

$$\mathcal{P}(G_{\psi'}) = \{(b)_1(c)_1(c)_2(a)_1(b)_2(a)_2(b)_3,\ (c)_1(b)_1(c)_2(a)_1(b)_2(a)_2(b)_3,\\ (b)_1(c)_1(c)_2(a)_1(b)_2(b)_3(a)_2,\ (c)_1(b)_1(c)_2(a)_1(b)_2(b)_3(a)_2\}.$$

Observe that when dropping the subscripts, we obtain words *bccabab*, *cbcabab*, *bccabba*, *cbcabba* that all have the following common properties: $u^{(1)}$ and $u^{(2)}$ are subwords of them, and they have $k_a = 2$ occurrences of a, $k_b = 3$ occurrences of b and $k_c = 2$ occurrences of c. We are going to show in what follows that considering every topological path of every graph associated to a K-marking of $u^{(1)}, \dots, u^{(\ell)}$ will give us all words u having $u^{(1)}, \dots, u^{(\ell)}$ as subwords and such that $|u|_a = k_a$ for all $a \in \mathcal{A}$.

As highlighted in the previous example, not all graphs associated to K-markings admit a topological sort. Some of them can also admit more than one. Therefore, we will denote by $\Psi_K^{t.s.}(u^{(1)}, \dots, u^{(\ell)})$ (resp., $\Psi_K^{u.t.s.}(u^{(1)}, \dots, u^{(\ell)})$) the set of all K-markings of $(u^{(1)}, \dots, u^{(\ell)})$ such that their associated graph admits a topological sort (resp., a unique topological sort).

Let us set

$$C_K(u^{(1)}, \dots, u^{(\ell)}) = \{u \in \mathcal{A}^* : u^{(1)}, \dots, u^{(\ell)} \text{ are subwords of } u \text{ and } |u|_a = k_a\ \forall a \in \mathcal{A}\}.$$

To establish the correspondence between paths in G_ψ, which are words over the alphabet $\{(a)_i : a \in \mathcal{A}, i \in [k_a]\}$, and words in $\mathcal{A}^*$, we need to use the following projection:

$$\Gamma : (a)_i \mapsto a.$$

Note that if p, p' are topological paths in G_ψ, then $p \neq p'$ implies $\Gamma(p) \neq \Gamma(p')$. Indeed, for any letter $a \in \mathcal{A}$, occurrences of $(a)_1, \ldots, (a)_{k_a}$ appear in this order in both words p and p'.

Lemma 5.3.7. *Let $\psi \in \Psi_K^{t.s.}(u^{(1)}, \ldots, u^{(\ell)})$. Let p be a path of $\mathcal{P}(G_\psi)$. The word $\Gamma(p)$ is in the set $C_K(u^{(1)}, \ldots, u^{(\ell)})$.*

Proof. The topological sort of G_ψ we consider is fixed by p. By construction of G_ψ, words

$$(u_1^{(j)})_{\psi(j,1)} \cdot (u_2^{(j)})_{\psi(j,2)} \cdots (u^{(j)}_{|u^{(j)}|})_{\psi(j,|u^{(j)}|)},$$

$j \in [\ell]$, are paths in this graph. By definition of a topological sort, any such word is a subword of the topological path p. Hence $u^{(j)}$ is a subword of $\Gamma(p)$ for every j.

Moreover, there is exactly one occurrence of any vertex of G_ψ in p. Hence, $\Gamma(p)$ contains k_a occurrences of the letter a, for all $a \in \mathcal{A}$. □

Lemma 5.3.8. *Let $u^{(1)}, \ldots, u^{(\ell)}$ be words of $\mathcal{A}^*$ and let K be a sequence of $\mathbb{N}^{\mathcal{A}}$. Let u be a word of $C_K(u^{(1)}, \ldots, u^{(\ell)})$. There exists $\psi \in \Psi_K^{t.s.}(u^{(1)}, \ldots, u^{(\ell)})$ and $p \in \mathcal{P}(G_\psi)$ such that $\Gamma(p) = u$.*

Proof. Let us write $u = u_1 \cdots u_n$. We will associate a natural number to any letter of u, with the following map β. Let $i \in [n]$ and a be the letter u_i. We set $\beta(i) = |u_1 \cdots u_i|_a$. As an example, if $u = abcabac$, then β is given here below:

u	a	b	c	a	b	a	c
n	1	2	3	4	5	6	7
$\beta(n)$	1	1	1	2	2	3	2

Then, for any $s \in [\ell]$, choose an occurrence of $u^{(s)}$ as subword of u, i.e., find a subsequence $1 \leq \iota_1^{(s)} < \iota_2^{(s)} < \cdots < \iota^{(s)}_{|u^{(s)}|} \leq n$ such that

$$u_{\iota_1^{(s)}} u_{\iota_2^{(s)}} \cdots u_{\iota^{(s)}_{|u^{(s)}|}} = u^{(s)}.$$

Such subsequences exist for every $s \in [\ell]$ since $u \in C_K(u^{(1)}, \ldots, u^{(\ell)})$.

Take ψ such that $\psi(s,i) = \beta(\iota_i^{(s)})$ for all s, i. It is direct that $\psi \in \Psi_K$. Let p be the word $(u_1)_{\beta(1)} \cdots (u_n)_{\beta(n)}$ written with vertices of G_ψ. It suffices to show that p is a topological path of G_ψ, since $\Gamma(p) = u$. Assume to the contrary that p is not a

topological path of G_ψ. Hence there exists in the graph, an edge $e = ((u_j)_{\beta(j)}, (u_i)_{\beta(i)})$ with $j > i$.

We cannot have $u_i = u_j$. Indeed, in this case, the presence of the edge e would imply that $\beta(j) < \beta(i)$. But by definition of β and the fact that $u_i = u_j$, $\beta(j) > \beta(i)$ since $j > i$, hence a contradiction.

Therefore, u_i and u_j are two different letters of $\mathcal{A}$. The existence of e in the graph induces that $u_j u_i$ is a factor of $u^{(s)}$ for a $s \in [\ell]$, and thus a subword of u. Moreover, by choosing this subsequence $(\iota_i^{(s)})_i$ and by definition of ψ, the $\beta(j)^{\text{th}}$ occurrence of u_j comes before the $\beta(i)^{\text{th}}$ occurrence of u_i in u.

On the other hand, $j > i$ implies that the $\beta(j)^{\text{th}}$ occurrence of u_j comes after the $\beta(i)^{\text{th}}$ occurrence of u_i in u. Here is the contradiction, and p is a topological path of G_ψ. □

These two lemmas give the correspondence between words of $C_K(u^{(1)}, \dots, u^{(\ell)})$ and paths from $\bigcup_{\psi \in \Psi_K} \mathcal{P}(G_\psi)$ and we can thus deduce the following result.

Proposition 5.3.9. *Let $u^{(1)}, \dots, u^{(\ell)} \in \mathcal{A}^*$ and $K \in \mathbb{N}^{\mathcal{A}}$. There exists $u \in C_K(u^{(1)}, \dots, u^{(\ell)})$ if and only if there exists $\psi \in \Psi_K^{t.s.}(u^{(1)}, \dots, u^{(\ell)})$.*

We can add a condition to ensure that the word u of the previous proposition is unique.

Proposition 5.3.10. *Let $u^{(1)}, \dots, u^{(\ell)} \in \mathcal{A}^*$ and $K \in \mathbb{N}^{\mathcal{A}}$. There exists a unique word $u \in C_K(u^{(1)}, \dots, u^{(\ell)})$ if and only if $\bigcup_{\psi \in \Psi_K^{t.s.}} \mathcal{P}(G_\psi)$ contains a unique element.*

Proof. To show that the condition is necessary, assume that there exist p, p' in the set $\bigcup_{\psi \in \Psi_K^{t.s.}} \mathcal{P}(G_\psi)$, with $p \neq p'$. Hence $\Gamma(p) \neq \Gamma(p')$ and, by Lemma 5.3.7, $C_K(u^{(1)}, \dots, u^{(\ell)})$ contains at least two words.

Let us now assume that there are two different words u and u' in $C_K(u^{(1)}, \dots, u^{(\ell)})$. By Lemma 5.3.8, there exist $\psi, \psi' \in \Psi_K^{t.s.}$ and $p \in \mathcal{P}(G_\psi)$, $p' \in \mathcal{P}(G_{\psi'})$ such that $\Gamma(p) = u$ and $\Gamma(p') = u'$. But $u \neq u'$ implies $p \neq p'$ and $\bigcup_{\psi \in \Psi_K^{t.s.}} \mathcal{P}(G_\psi)$ contains more than one element. □

The previous proposition can be rephrased: the word u is unique if and only if every K-marking whose associated graph admits a topological sort is such that this topological sort is unique, and moreover, the topological path is the same from a K-marking to another. Note that it is not asked to have $\#\Psi_K^{t.s.} = \#\Psi_K^{u.t.s.} = 1$, even if this condition is sufficient.

In this section we are interested in reconstructing a word from all its binary projections. Here comes the most useful result.

Proposition 5.3.11. *Let $u^{(1)}, \dots, u^{(\ell)}$ be all binary projections of a word $u \in \mathcal{A}^*$. There exists a unique $\psi \in \Psi_K(u^{(1)}, \dots, u^{(\ell)})$. Moreover, $\mathcal{P}(G_\psi)$ contains a unique element.*

Proof. Let us write $K_{min} = (k_{min,a})_{a\in\mathcal{A}}$. Since $u^{(1)}, \dots, u^{(\ell)}$ are all binary projections of the same word, we have, for any $a \in \mathcal{A}$, either $|u^{(j)}|_a = 0$, or $|u^{(j)}|_a = k_{min,a}$. Hence, the K_{min}-marking ψ is given as in the proof of Lemma 5.3.8 and, since every $u^{(j)}$ appears only once as a subword of u, ψ is unique. We also get that G_ψ admits a topological sort $p = (u_1)_{\beta(1)} \cdots (u_n)_{\beta(n)}$. To show the unicity, let us prove that this path is hamiltonian. Let $i \in [n-1]$. The edge $((u_i)_{\beta(i)}, (u_{i+1})_{\beta(i+1)}))$ is in G_ψ. It is obvious if $u_i = u_{i+1}$ and, otherwise, it comes from the fact that the factor u_iu_{i+1} appears in the binary projection of u over the subalphabet $\{u_i, u_{i+1}\}$. □

Corollary 5.3.12. *Any word can be reconstructed uniquely from its binary projections over subalphabets of $\mathcal{A}$.*

Proof. The result is direct from Propositions 5.3.10 and 5.3.11. □

In Subsection 5.3.3 we present an algorithm, based on a depth-first search of G_ψ, that computes in linear time a topological sort of a graph.

5.3.2 Comparing the number of queries with the classical reconstruction problem

We first give the minimal number of queries we need to ask to determine u uniquely. Recall that $\mathcal{A} = \{a_1, \dots, a_q\}$.

Proposition 5.3.13. *Let $u \in \mathcal{A}^n$ be an unknown word. Suppose that $|u|_a$ is known for every $a \in \mathcal{A}$. Let ι be a permutation of $[q]$ such that*

$$|u|_{a_{\iota(1)}} \leq |u|_{a_{\iota(2)}} \leq \cdots \leq |u|_{a_{\iota(q)}}.$$

Then u can be uniquely determined by asking

$$(q-1) + \sum_{i\in[q]} |u|_{a_{\iota(i)}}(q-i) \tag{5.8}$$

queries.

Proof. To know $|u|_a$ for every $a \in \mathcal{A}$ and determine ι, we need to ask $q-1$ queries. Then we consider every subalphabet $\mathcal{A}_{i,j} := \{a_{\iota(i)}, a_{\iota(j)}\}$ with $i < j$. By Corollary 5.2.5, we can reconstruct $\pi_{a_{\iota(i)},a_{\iota(j)}}(u)$ asking at most $|u|_{a_{\iota(i)}}$ queries. For any $i \in [q]$, they are $q-i$ subalphabets $\mathcal{A}_{i,j}$ with $i < j$. Putting all the results together gives Quantity (5.8). □

As explained at the beginning of Subsection 5.2.2, the results from [62] and [105] yield that, for $n > 6$,

$$\sum_{i=1}^{\lfloor \frac{16}{7}\sqrt{n}\rfloor+5} \frac{1}{i} \sum_{d|i} \mu(d) \cdot q^{\frac{i}{d}} \tag{5.9}$$

queries are sufficient for a unique reconstruction of any word of length n.

Theorem 5.3.14. *Let u and ι be as in the previous proposition. If $n \geq q-1$, u can be determined by asking at most*

$$\sum_{i\in[q]} |u|_{a_{\iota(i)}}(q-i+1) \tag{5.10}$$

queries. This quantity is less than qn.

Proof. We are going to show that the quantity in Equation (5.8) is less than or equal to (5.10), which is sufficient. We have

$$\begin{aligned}(5.8) &\leq n + \sum_{i\in[q]} |u|_{a_{\iota(i)}}(q-i) \\ &= \sum_{i\in[q]} |u|_{a_{\iota(i)}} + \sum_{i\in[q]} |u|_{a_{\iota(i)}}(q-i) \\ &= (5.10).\end{aligned}$$

It remains to show that (5.10) is less than qn. Note that, by definition of ι, we have

$$|u|_{a_{\iota(1)}} \leq \left\lfloor \frac{n}{q} \right\rfloor, \quad |u|_{a_{\iota(2)}} \leq \left\lfloor \frac{n}{q-1} \right\rfloor, \quad \ldots, \quad |u|_{a_{\iota(q)}} \leq n,$$

and for any $i > 1$, if $|u|_{a_{\iota(i)}} = \left\lfloor \frac{n}{q+1-i} \right\rfloor$ (i.e., the equality holds) then $|u|_{a_{\iota(j)}} = 0$ for every $j < i$. Hence at least one of the previous inequalities is strict. We thus have

$$\begin{aligned}(5.10) &< \sum_{i\in[q]} \left\lfloor \frac{n}{q+1-i} \right\rfloor (q-i+1) \\ &\leq \sum_{i\in[q]} n \\ &= qn.\end{aligned}$$

□

Before showing that our number of queries is less than (5.9), we need several combinatorial lemmas.

Lemma 5.3.15. *Let $N_q(i)$ denote the number of Lyndon words of length $i \in \mathbb{N}$ on $\{a_1, \ldots, a_q\}$ (with $q \geq 2$). We have*

$$N_q(i) > \frac{1}{i}\frac{(q+1)^{\frac{i}{2}} - 1}{q}. \tag{5.11}$$

Proof. We first show that the result is true for $i \in [4]$. When $i = 1$, the statement is equivalent to $q^2 > \sqrt{q+1} - 1$, which is always true for $q \geq 2$. When $i = 2$, we must have

$$\mu(1)q^2 + \mu(2)q > \frac{q+1-1}{q} \Leftrightarrow q^2 - q - 1 > 0,$$

which is also true for $q \geq 2$. When $i = 3$, we have to prove that

$$q^3 - q > \frac{(q+1)\sqrt{q+1} - 1}{q} \Leftrightarrow q^4 - q^2 - (q+1)\sqrt{q+1} + 1 > 0.$$

We have

$$\begin{aligned} q^4 - q^2 - (q+1)\sqrt{q+1} + 1 &> q^4 - q^2 - (q+1)^2 + 1 \\ &= q^4 - 2q^2 - 2q = q(q^3 - 2q - 2) > 0 \end{aligned}$$

for all $q \geq 2$. Finally, for $i = 4$, we have to prove that

$$q^4 - q^2 > \frac{(q+1)^2 - 1}{q} \Leftrightarrow q^4 - q^2 - q - 2 > 0,$$

which is true for $q \geq 2$.

Let us now prove the statement for all $i \geq 5$. By Lemma 5.2.10, we have

$$N_q(i) \geq \frac{1}{i}\left(q^i - \frac{q^{\frac{i}{2}+1} - 1}{q-1}\right).$$

We have to show that

$$\begin{aligned} q^i - \frac{q^{\frac{i}{2}+1} - 1}{q-1} &> \frac{(q+1)^{\frac{i}{2}} - 1}{q} \\ \Leftrightarrow q^i - \frac{q^{\frac{i}{2}+1} - 1}{q-1} - \frac{(q+1)^{\frac{i}{2}} - 1}{q} &> 0. \end{aligned} \tag{5.12}$$

It is sufficient to prove that

$$q^i - \frac{q^{\frac{i}{2}+1}}{q-1} - \frac{(q+1)^{\frac{i}{2}}}{q} > 0.$$

Let us consider two cases, depending on if $\frac{q^{\frac{i}{2}+1}}{q-1} \leq \frac{(q+1)^{\frac{i}{2}}}{q}$ or not.

1. If $\frac{q^{\frac{i}{2}+1}}{q-1} \leq \frac{(q+1)^{\frac{i}{2}}}{q}$, we have

$$q^i - \frac{q^{\frac{i}{2}+1}}{q-1} - \frac{(q+1)^{\frac{i}{2}}}{q} \geq \frac{q^{i+1}}{q} - \frac{2(q+1)^{\frac{i}{2}}}{q}$$

so we are going to prove that the latter quantity is positive.

Let us first consider the case where i is even. There exists $n \in \mathbb{N}$ such that $i = 2n$. We have

$$\begin{aligned} q^{2n+1} &= (q^2)^n q \\ &= 2(q^2)^n + (q-2)(q^2)^n \\ &> 2(q+1)^n + (q-2)(q^2)^n \\ &\geq 2(q+1)^n, \end{aligned}$$

hence the conclusion. If i is odd, there exists $n \in \mathbb{N}$ such that $i = 2n-1$. We thus have to show

$$\begin{aligned} & q^{2n} - 2(q+1)^{n-\frac{1}{2}} > 0 \\ \Leftrightarrow\ & \sqrt{q+1}\, q^{2n} - 2(q+1)^n > 0. \end{aligned}$$

We have

$$\begin{aligned} \sqrt{q+1}\, q^{2n} &= 2(q^2)^n + (\sqrt{q+1} - 2)(q^2)^n \\ &> 2(q+1)^n, \end{aligned}$$

if $q \geq 3$. If $q = 2$, the fact that

$$4^n - 2 \cdot 3^{n-\frac{1}{2}} > 0,$$

for all $n \geq 1$, is trivial.

2. If $\frac{q^{\frac{i}{2}+1}}{q-1} > \frac{(q+1)^{\frac{i}{2}}}{q}$, we have

$$q^i - \frac{q^{\frac{i}{2}+1}}{q-1} - \frac{(q+1)^{\frac{i}{2}}}{q} > \frac{q^{i+1}}{q} - \frac{2q^{\frac{i}{2}+1}}{q-1}.$$

Let us thus show that

$$\begin{aligned} & \frac{q^{i+1}}{q} - \frac{2q^{\frac{i}{2}+1}}{q-1} > 0 \\ \Leftrightarrow\ & q^{i+1}(q-1) - 2q^{i/2+2} > 0 \\ \Leftrightarrow\ & q^{\frac{i}{2}+2}(q^{i+1-\frac{i}{2}-2}(q-1) - 2) > 0 \\ \Leftrightarrow\ & q^{\frac{i}{2}-1}(q-1) > 2. \end{aligned}$$

For any fixed $i \geq 5$, the function $f_i : q \mapsto q^{\frac{i}{2}-1}(q-1)$ is increasing if $q > 1$. Moreover, for a fixed $q \geq 2$, the function $f_q : i \mapsto q^{\frac{i}{2}-1}(q-1)$ is also increasing if $i \geq 5$. We have $f_5(2) = 2\sqrt{2} > 2$, hence the conclusion for all $i \geq 5$ and $q \geq 2$.

□

Lemma 5.3.16. *Let us define*

$$f(i,q) = \frac{1}{i}\frac{(q-1)(q+1)^{i/2} - q^{i/2+1} + 1}{q(q-1)}$$

for every $i > 0$ and every $q \geq 2$. For every $i, q \in \mathbb{N}$ such that $i \geq 7$ and $q \geq 2$, we have

$$f(i,q) < f(i,q+1).$$

Proof. Fix $i, q \in \mathbb{N}$ such that $i \geq 7$ and $q \geq 2$. We have

$$\begin{aligned}
&f(i,q+1) > f(i,q) \\
\Leftrightarrow &\left(q(q+2)^{i/2} - (q+1)^{i/2+1} + 1\right)(q-1) > \left((q-1)(q+1)^{i/2} - q^{i/2+1} + 1\right)(q+1) \\
\Leftrightarrow &(q+2)^{i/2}q(q-1) - 2(q-1)(q+1)^{i/2+1} + q^{i/2+2} + q^{i/2+1} - 2 > 0. \qquad (5.13)
\end{aligned}$$

Let us assume that i is odd. There exists $n \geq 3$ such that $i = 2n+1$. The left-hand side of Inequality (5.13) can be rewritten and we have to prove that

$$h(n,q) := (q+2)^n\sqrt{q+2}\,q(q-1) - 2(q-1)(q+1)^{n+1}\sqrt{q+1} + q^{n+2}\sqrt{q} + q^{n+1}\sqrt{q} - 2$$

is positive for every $q \geq 2$ and $n \geq 3$. Figure 5.3 displays functions h for $q = 2, 3, 4$ and $n \in [3, 5]$.

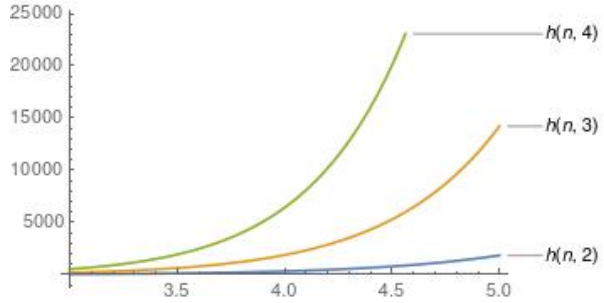


Figure 5.3: Functions $h(n,2)$, $h(n,3)$ and $h(n,4)$.

We get

$$h(n,q+1) - h(n,q) = -q(q+1)\underbrace{(q^{n+\frac{1}{2}} - 3(q+1)^{n+\frac{1}{2}} + 3(q+2)^{n+\frac{1}{2}} - (q+3)^{n+\frac{1}{2}})}_{:=z(n,q)}$$

Hence it is sufficient to prove that

1. $z(n,q) < 0$ for all $q \geq 2$ and $n \geq 3$, and;
2. $h(n,2) > 0$ for all $n \geq 3$.

For the second item, note that

$$h(n,2) = 4^{n+1} - 2 \cdot 3^{n+1}\sqrt{3} + 2^{n+2}\sqrt{2} + 2^{n+1}\sqrt{2} - 2 > 0$$

for all $n \geq 4$ since in this case, $4^{n+1} > 2 \cdot 3^{n+1}\sqrt{3}$. One can check that $h(3,2) > 0$.

We will prove the first item by considering partial derivatives, with respect to q, of $z(q,n)$. We have, for any $j \in [n-1]$,

$$\frac{\partial^j z}{\partial q}(n,q) = \left(n + \frac{1}{2}\right)\left(n + \frac{1}{2} - 1\right)\cdots\left(n + \frac{1}{2} - (j-1)\right) z(n-j,q).$$

We now prove that $z(n,q) < 0$ for any q by induction on $n \geq 0$:

- The result is true for $n = 0$: we have to prove that $z(1,q) < 0$ which is equivalent to $\sqrt{q} + 3\sqrt{q+2} < 3\sqrt{q+1} + \sqrt{q+3}$. Taking the square of both members at each step since all terms are positive, we obtain

$$\begin{aligned} &\sqrt{q} + 3\sqrt{q+2} < 3\sqrt{q+1} + \sqrt{q+3} \\ \Leftrightarrow\ &\sqrt{q}\sqrt{q+2} + 1 < \sqrt{q+1}\sqrt{q+3} \\ \Leftrightarrow\ &\sqrt{q}\sqrt{q+2} < q+1 \\ \Leftrightarrow\ &0 < 1. \end{aligned}$$

- Let us assume that $z(n,q) < 0$ for any $q \geq 2$. Then

$$\frac{\partial z}{\partial q}(n+1,q) = \left(n + \frac{3}{2}\right) z(n,q) < 0$$

so the function $q \mapsto z(n+1,q)$ is decreasing if $q \geq 2$. It remains to check that $z(n+1,2) < 0$.

Consider the function $z_2 : n \mapsto z(n,2)$. We have

$$z_2(n) = \sqrt{2}\,2^n - 3\sqrt{3}\,3^n + 3\sqrt{4}\,4^n - \sqrt{5}\,5^n.$$

Hence

$$\begin{aligned} &z_2(n) < 0 \\ \Leftrightarrow\ &\sqrt{2}\,2^n + 3\sqrt{4}\,4^n < 3\sqrt{3}\,3^n + \sqrt{5}\,5^n. \end{aligned}$$

We obviously have $\sqrt{2}\,2^n < 3\sqrt{3}\,3^n$. Moreover, $3\sqrt{4}\,4^n < \sqrt{5}\,5^n$ for every $n \geq 5$. Hence $z(n,2) < 0$ for every $n \geq 5$ and the result can be checked by hand for $n = 3$ and $n = 4$.

Recall that we assumed i to be odd in Inequality (5.13). If i is even, there exists $n \geq 4$ such that $i = 2n$. The proof is very similar in this case: define

$$h(n,q) := (q+2)^n q(q-1) - 2(q-1)(q+1)^{n+1} + q^{n+2} + q^{n+1} - 2$$

and proceed in the same way, with

$$z(n,q) := q^n - 3(q+1)^n + 3(q+2)^n - (q+3)^n.$$

There is only one minor change: prove that $z(n,q) < 0$ by induction on n, with base case $n = 3$ and not $n = 0$ anymore. □

Theorem 5.3.17. *Let $u \in \mathcal{A}^n$ be an unknown word of length $n \geq q-1$. We can determine u uniquely by asking strictly less queries than Quantity* (5.9)*, which stands for the classical reconstruction problem.*

Proof. Set, as previously, $\alpha^{\lfloor\cdot\rfloor}(x) = \lfloor \frac{16}{7}\sqrt{x} \rfloor + 5$ and $\alpha(x) = \frac{16}{7}\sqrt{x} + 5$ for any $x > 0$. First, we get by Lemma 5.3.15

$$(5.9) \geq \sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} \frac{1}{i} \left(\frac{(q+1)^{i/2} - 1}{q} \right).$$

Let us show by induction on $q \geq 2$ that, for any $n \in \mathbb{N}$ and any $u \in \{a_1, \ldots, a_q\}^n$, we have

$$\sum_{i \in [q]} |u|_{a_{\iota(i)}} (q - i + 1) < \sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} \frac{1}{i} \left(\frac{(q+1)^{\frac{i}{2}} - 1}{q} \right), \tag{5.14}$$

where permutation ι orders letters of $\{a_1, \ldots, a_q\}$ by increasing number of occurrences in u. Hence, by Theorem 5.3.14, we get the conclusion for any $n \geq q-1$.

Let us start by the base case: let q be equal to 2. We want to show that, for any $n \in \mathbb{N}$ and any $u \in \{a_1, a_2\}^n$,

$$2|u|_{a_{\iota(1)}} + |u|_{a_{\iota(2)}} < \sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} \frac{1}{i} \left(\frac{3^{\frac{i}{2}} - 1}{2} \right).$$

On the one hand, the left-hand side is less than or equal to $\frac{3n}{2}$, by definition of ι. On the other hand, we can minimize the right-hand side as follows:

$$\begin{aligned}
\sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} \frac{1}{i} \left(\frac{3^{\frac{i}{2}} - 1}{2} \right) &> \frac{1}{2\alpha^{\lfloor\cdot\rfloor}(n)} \sum_{i=1}^{\alpha^{\lfloor\cdot\rfloor}(n)} \left(\sqrt{3}^i - 1 \right) \\
&= \frac{1}{2\alpha^{\lfloor\cdot\rfloor}(n)} \frac{\sqrt{3}^{\alpha^{\lfloor\cdot\rfloor}(n)+1} - 1}{\sqrt{3} - 1} - \frac{1}{2} \\
&\geq \frac{1}{2\alpha(n)} \frac{\sqrt{3}^{\alpha(n)} - 1}{\sqrt{3} - 1} - \frac{1}{2}.
\end{aligned}$$

Define

$$h(x) = \frac{1}{2\alpha(x)} \frac{\sqrt{3}^{\alpha(x)} - 1}{\sqrt{3} - 1} - \frac{1}{2} - \frac{3x}{2}$$

and proceed as in the proof of Theorem 5.2.11 to show that this last quantity is strictly positive for every $x \geq 1$.

Let us now assume that $q \geq 3$ and that the result holds for any $q' < q$. Take $n \in \mathbb{N}$, $u \in \{a_1, \ldots, a_q\}^n$. We have

$$\begin{aligned} \sum_{i \in [q]} |u|_{a_{\iota(i)}} (q + 1 - i) &= n + \sum_{i \in [q]} |u|_{a_{\iota(i)}} (q - i) \\ &= n + \sum_{i \in [q-1]} |u|_{a_{\iota(i)}} (q - i). \end{aligned}$$

Let us consider the word u' obtained from u by removing all occurrences of letter $a_{\iota(q)}$. It is thus defined over the alphabet $\{a_{\iota(1)}, \ldots, a_{\iota(q-1)}\}$; denote by n' its length. We still have $|u'|_{a_{\iota(i)}} \leq |u'|_{a_{\iota(j)}}$ if $i < j$ so, by induction hypothesis, we get

$$\begin{aligned} \sum_{i \in [q-1]} |u'|_{a_{\iota(i)}} (q - i) &< \sum_{i=1}^{\alpha^{\lfloor \cdot \rfloor}(n')} \frac{1}{i} \left(\frac{q^{\frac{i}{2}} - 1}{q - 1} \right) \\ &\leq \sum_{i=1}^{\alpha^{\lfloor \cdot \rfloor}(n)} \frac{1}{i} \left(\frac{q^{\frac{i}{2}} - 1}{q - 1} \right), \end{aligned}$$

since $n' < n$. Hence, for the left-hand side of (5.14), we get

$$\sum_{i \in [q]} |u|_{a_{\iota(i)}} (q - i + 1) < n + \sum_{i=1}^{\alpha^{\lfloor \cdot \rfloor}(n)} \frac{1}{i} \left(\frac{q^{i/2} - 1}{q - 1} \right).$$

We have to compare this quantity with the right-hand side of (5.14), which can be rewritten as

$$\sum_{i=1}^{\alpha^{\lfloor \cdot \rfloor}(n)} \frac{1}{i} \left(\frac{(q+1)^{i/2} - 1}{q} \right) = \sum_{i=1}^{\alpha^{\lfloor \cdot \rfloor}(n)} \frac{1}{i} \left(\frac{(q-1)((q+1)^{i/2} - 1)}{q(q-1)} \right).$$

Thus the claim is proven, if the subtraction of the latter one and the previous one is greater than zero, i.e., we show that

$$\begin{aligned} \left(\sum_{i \in [\alpha^{\lfloor \cdot \rfloor}(n)]} \frac{1}{i} \frac{(q-1)((q+1)^{\frac{i}{2}} - 1) - q(q^{\frac{i}{2}} - 1)}{q(q-1)} \right) - n > 0, \text{ i.e.,} \\ \left(\sum_{i \in [\alpha^{\lfloor \cdot \rfloor}(n)]} \frac{1}{i} \frac{(q-1)(q+1)^{\frac{i}{2}} - q q^{\frac{i}{2}} + 1}{q(q-1)} \right) - n > 0. \end{aligned} \tag{5.15}$$

Set $f(i) = \frac{1}{i}\frac{(q-1)(q+1)^{\frac{i}{2}} - qq^{\frac{i}{2}}+1}{q(q-1)}$ for all $i \in [\alpha^{\lfloor\cdot\rfloor}(n)]$; the proof of (5.15) contains the following steps:

1. For all $i \geq 2$ we have $f(i) \geq 0$;
2. $f(5) + f(1) \geq 0$, and;
3. $f(\alpha^{\lfloor\cdot\rfloor}(n)) - n > 0$.

Step 1.: For $i = 2$ we have

$$f(2) = \frac{1}{2}\frac{(q-1)(q+1) - q^2 + 1}{q(q-1)} = \frac{q^2 - 1 - q^2 + 1}{2q(q-1)} = 0.$$

For $i = 3$ we have

$$f(3) = \frac{1}{3}\frac{(q-1)(q+1)\sqrt{q+1} - q^2\sqrt{q} + 1}{q(q-1)}.$$

Consider the function

$$g : \mathbb{R} \to \mathbb{R} \; : \; q \mapsto q^4 - 2q^3 - 2q^2 + q + 1$$

represented in Figure 5.4.

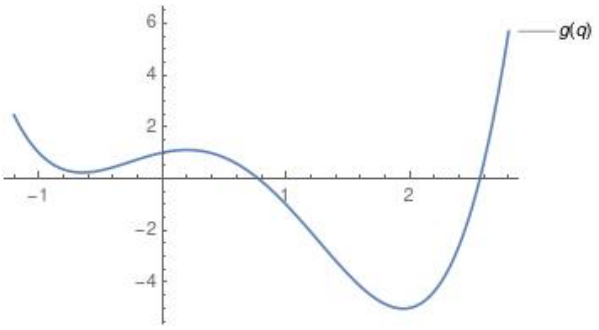


Figure 5.4: Function g on the domain $[-1.2; 1.8]$.

This function has two minima (between -0.75 and -0.5 as well as between 1.75 and 2) and one maximum (between 0.125 and 0.25). Since g has only two inflexion points and g is strictly greater than zero at the first minima, g has only two roots. The first root is between 0.7 and 0.8 and the second root is between 2.5 and 2.75. Thus for all $q \geq 2.75$ we have $g(q) > 0$. This implies $q^5 + q^4 - 2q^3 - 2q^2 + q + 1 > q^5$. Hence equivalently we get

$$(q+1)(q^4 - 2q^2 + 1) > q^5,$$

i.e., $(q+1)(q^2-1)^2 > qq^4$. This implies $\sqrt{q+1}(q^2-1) > \sqrt{q}q^2$ which proves that the numerator of $f(3)$ is positive and hence $f(3) > 0$. Before we prove the claim for $i \geq 4$, we will prove that $(q-1)(q+1)^j \geq q^{j+1}$ for $j \geq 2$. Firstly we get

$$(q-1)(q+1)^j = \left(\sum_{k \in [j]} \left(\binom{j}{k-1} - \binom{j}{k} \right) q^k \right) + q^{j+1} - 1.$$

Due to the central symmetry of each row of the Pascal triangle, for $k \leq \lfloor j/2 \rfloor$, we have

$$\binom{j}{j-k} - \binom{j}{j-k+1} = -\left(\binom{j}{k-1} - \binom{j}{k} \right) > 0$$

and thus

$$(q-1)(q+1)^j = \left(\sum_{k \in [\lfloor j/2 \rfloor]} \left(\binom{j}{k} - \binom{j}{k-1} \right) (q^{j-k+1} - q^k) \right) + q^{j+1} - 1.$$

Since $k \leq \lfloor j/2 \rfloor$, we have $j-k+1 > k$ and each term of the above sum is thus positive. This shows that $(q-1)(q+1)^j \geq q^{j+1}$. This leads to the following estimations for $f(i)$. For $i = 2j$ and $j \geq 2$ we get

$$f(i) = \frac{(q-1)(q+1)^j - qq^j + 1}{iq(q-1)} \geq \frac{q^{j+1} - q^{j+1} + 1}{iq(q-1)} > 0.$$

Finally for $i = 2j+1$ and $j \geq 2$ we get

$$f(i) = \frac{(q-1)(q+1)^j \sqrt{q+1} - qq^j \sqrt{q} + 1}{iq(q-1)} \geq \frac{q^{j+1}(\sqrt{q+1} - \sqrt{q}) + 1}{iq(q-1)} > 0.$$

Step 2.: Notice that $\alpha^{\lfloor \cdot \rfloor}(n) \geq 7$ holds and thus $f(5)$ is always a summand. For $f(5) + f(1)$ we have to prove

$$\frac{(q-1)(q+1)^2 \sqrt{q+1} - qq^2 \sqrt{q} + 1}{5q(q-1)} + \frac{(q-1)\sqrt{q+1} - q\sqrt{q} + 1}{q(q-1)} \geq 0.$$

Thus we get for the numerator

$$\begin{aligned}
&(q-1)(q+1)^2 \sqrt{q+1} - qq^2 \sqrt{q} + 1 + 5(q-1)\sqrt{q+1} - 5q\sqrt{q} + 5 \\
&= (q-1)\sqrt{q+1}((q+1)^2 + 5) - q\sqrt{q}(q^2+5) + 6 \\
&= (q-1)\sqrt{q+1}(q^2 + 2q + 6) - q\sqrt{q}(q^2+5) + 6 \\
&= q^3\sqrt{q+1} + q^2\sqrt{q+1} + 4q\sqrt{q+1} - 6\sqrt{q+1} - q^3\sqrt{q} - 5q\sqrt{q} + 6.
\end{aligned}$$

We have $q^3\sqrt{q+1} > q^3\sqrt{q}$ and, since $q \geq 3$,

$$q^2\sqrt{q+1} \geq (6+q)\sqrt{q+1}.$$

Therefore $q^2\sqrt{q+1}+4q\sqrt{q+1} \geq 6\sqrt{q+1}+5q\sqrt{q}$ and the numerator is positive.

Step 3.: For fixed $i \geq 5$, the value of $f(i)$ increases when q increases, by Lemma 5.3.16. This implies

$$f(\alpha^{\lfloor\cdot\rfloor}(n)) \geq \frac{2\cdot 4^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}} - 3\cdot 3^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}} + 1}{6\alpha^{\lfloor\cdot\rfloor}(n)} = \frac{2^{\alpha^{\lfloor\cdot\rfloor}(n)+1} - 3^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}+1} + 1}{6\alpha^{\lfloor\cdot\rfloor}(n)}.$$

We are going to prove that

$$2^{\alpha^{\lfloor\cdot\rfloor}(n)+1} - 3^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}+1} + 1 > 6\alpha^{\lfloor\cdot\rfloor}(n)n. \tag{5.16}$$

First, we have

$$2^{\alpha^{\lfloor\cdot\rfloor}(n)+1} - 3^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}+1} > 2^{\alpha^{\lfloor\cdot\rfloor}(n)-1} - 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}.$$

Indeed, this inequality is equivalent to

$$\begin{aligned}
&2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}\left(3\cdot 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}-1}+1\right) > 3^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}+1}\\
\Leftrightarrow\ &2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}-1}+\frac{1}{3} > \left(\frac{3}{2}\right)^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}\\
\Leftrightarrow\ &\ln\left(2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}\left(\frac{1}{2}+\frac{1}{3\cdot 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}}\right)\right) > \frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}\ln\left(\frac{3}{2}\right)\\
\Leftrightarrow\ &\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}\ln(2)+\ln\left(\frac{1}{2}+\frac{1}{3\cdot 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}}\right) > \frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}\ln\left(\frac{3}{2}\right)\\
\Leftrightarrow\ &\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}\ln\left(\frac{4}{3}\right)+\ln\left(\frac{1}{2}+\frac{1}{3\cdot 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}}\right) > 0,
\end{aligned}$$

but $\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2} > \frac{5}{2}$ thus it is sufficient to have

$$\ln\left(\left(\frac{4}{3}\right)^{\frac{5}{2}}\right)+\ln\left(\frac{1}{2}\right) > 0,$$

which is true.

Therefore, it is sufficient for (5.16) to show that

$$2^{\alpha^{\lfloor\cdot\rfloor}(n)-1} - 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}} = 2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}}(2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}-1}-1) > 6\alpha^{\lfloor\cdot\rfloor}(n)n.$$

Note that $2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}} > n$ (indeed, $\lfloor\frac{16}{7}\sqrt{n}\rfloor+5 \geq \lfloor 2\sqrt{n}\rfloor+5$, thus $2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}} \geq 2^{\frac{5}{2}}\cdot 2^{\frac{\lfloor 2\sqrt{n}\rfloor}{2}}$ and, once again, taking the logarithms, one can check that $2^{\frac{5}{2}}\cdot 2^{\frac{\lfloor 2\sqrt{n}\rfloor}{2}} > n$ holds).

To verify (5.16), it remains to show that $2^{\frac{\alpha^{\lfloor\cdot\rfloor}(n)}{2}-1} - 1 \geq 6\alpha^{\lfloor\cdot\rfloor}(n)$. By classical tools of analysis, we show that the function $y \mapsto 2^{\frac{y}{2}-1} - 6y - 1$ is strictly positive for every $y \geq 16$. Hence Equation (5.16) is true for every $n \geq 24$. We can check its validity for $n \in [23]$ by directly computing its values.

By Steps 1., 2., and 3. Equation (5.15) is proven and this proves the claim. □

5.3.3 Complexity of the reconstruction of u from its binary projections

Let us first present a depth-first search algorithm of a directed graph $G = (V, E)$, coming from [28]. This will be used to present an $\mathcal{O}(V + E)$-time algorithm computing a topological sort of G. Let us consider a graph G for which we know its set of vertices V, its set of edges E and, for any vertex $u \in V$, the set of successors of u (i.e., vertices $v \in V$ such that $(u, v) \in E$) denoted $G.Adj(u)$.

`DFS`(G):

```
1: for each vertex u ∈ V do
2:     u.color = WHITE
3:     u.π = NIL
4: end for
5: global time = 0
6: for each vertex u ∈ V do
7:     if u.color == WHITE then
8:         DFS-VISIT(G, u)
9:     end if
10: end for
```

`DFS-VISIT`(G, u):

```
1: global time
2: time = time + 1
3: u.d = time
4: u.color = GRAY
5: for each v ∈ G.Adj(u) do
6:     if v.color == WHITE then
7:         u.π += {v}
8:         DFS-VISIT(G, v)
9:     end if
10: end for
11: u.color = BLACK
12: time = time + 1
13: u.f = time
```

Figure 5.5: Algorithm `DFS`.

Let us give some explanation about the algorithm given in Figure 5.5. Every vertex $u \in V$ has four attributes for which the algorithm attributes a value: $u.d$, $u.f$ that are integers, $u.color$ that has a color and $u.\pi$ containing a list of vertices. In the first one is encoded the time at which u was discovered by the algorithm; in the second the time when we finished considering u and all its descendants[2]. The color of

[2]Since G is a directed graph, we call the *decendants* of u all vertices v such that there exists a path from u to v in G.

every vertex is initially white, it turns gray during the time where it is considered by the algorithm and is finally set to black when the vertex itself and all its descendants are treated. At the end of the algorithm, $u.\pi$ contains all successors of u that were not yet considered when u is.

From the DFS algorithm we can easily deduce a topological sort of G: let us consider an empty list ℓ. Every time that we update $u.f$ for any u in DFS-VISIT, add the vertex u in front of list ℓ. Without any supplementary time cost, list ℓ contains, at the end of the algorithm, a list of vertices of G given by decreasing finishing time. This gives a topological sort of G.

Example 5.3.18. Let us consider an example coming from [28]: graph G given in Figure 5.6 is such that its vertices are clothes and an edge goes from vertex u to vertex v if you need, when putting on your clothes, to wear garment u before garment v.

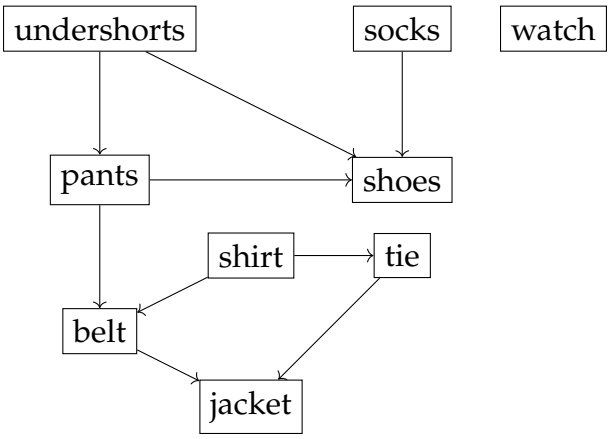


Figure 5.6: A graph indicating in which order you should put your clothes on.

We consider the set of vertices

$$G = \{\text{shirt}, \text{jacket}, \text{belt}, \text{watch}, \text{undershorts}, \text{tie}, \text{shoes}, \text{socks}, \text{pants}\}$$

given in this order, and, for every $u \in G$, the set $G.Adj(u)$ is given in the same order. The values we get for attributes d, f and π after applying DFS on G are collected in Table 5.2.

The list ℓ encoding vertices by increasing finishing time is thus

$$\{\text{socks}, \text{undershorts}, \text{pants}, \text{shoes}, \text{watch}, \text{shirt}, \text{tie}, \text{belt}, \text{jacket}\}.$$

Following this order you can put your clothes on in a convenient way.

Remark 5.3.19. The running time of the algorithm DFS is in $\Theta(V + E)$. Indeed, the loop on lines 1–3 and 6–9 in DFS take time $\Theta(V)$, exclusive of the time to execute

u	shirt	jacket	belt	watch	undershorts	tie	shoes	socks	pants
$u.d$	1	3	2	9	11	6	12	17	14
$u.f$	8	4	5	10	16	7	13	18	15
$u.\pi$	{belt, tie}	$\emptyset$	{jacket}	$\emptyset$	{shoes, pants}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Table 5.2: Applying DFS on G.

calls to DFS-VISIT. During an execution of DFS-VISIT(G, v), the loop on lines 5–10 is executed $\#|G.Adj(v)|$ times. Since DFS-VISIT is called exactly once on every vertex (only when it is still colored in white), the total number of executions of the loop on lines 5–10 is

$$\sum_{v \in V} \#|G.Adj(v)| = \#E.$$

Hence the complexity of the algorithm of depth-first search is as announced. In the topological sort algorithm, completing list ℓ by adding a vertex as first element each time that the finishing attribute of this vertex is set to a value requires no additional time.

Recall from Proposition 5.3.4 that an acyclic graph admits a hamiltonian path if and only if its topological sort is unique. This folklore fact and the use of the DFS algorithm are sufficient to claim that even though deciding if a graph admits a hamiltonian path is NP-complete in the general case, it can be done in linear time for a directed acyclic graph $G = (V, E)$ (see [124] for example). Indeed, it suffices to find a topological sort $v_1 < \cdots < v_{\#V}$ of G in time $\Theta(V + E)$ and then, via Proposition 5.3.4, we can check if the topological sort is unique by verifying that every pair (v_i, v_{i+1}) of the topological sort ($i \in [\#V - 1]$) is an edge of G.

Applying the algorithm to our problem

Recall that in Subsection 5.3.1 we are interested in finding a K_{min}-marking of all binary projections of an unknown word u, and a topological sort of its associated graph.

Let $u \in \{a_1, \ldots, a_q\}^n$ be an unknown word of length n and denote by $\mathcal{U}_{bin}$ the set of all its binary projections:

$$\mathcal{U}_{bin} = \{\pi_{i,j}(u) : i \neq j \text{ and } \{i, j\} \subseteq \mathcal{A}\}.$$

Fist note that

$$\sum_{u' \in \mathcal{U}_{bin}} |u'| = (q - 1)n.$$

Finding the unique K_{min}-marking ψ of words from $\mathcal{U}_{bin}$ can thus be done in time $\Theta((q-1)n)$. The associated graph G_ψ is composed of

$$\sum_{a\in\mathcal{A}} k_a = \sum_{a\in\mathcal{A}} |u|_a = n$$

vertices and a maximum of

$$\begin{aligned}&\sum_{a\in\mathcal{A}} (|u|_a - 1) + \sum_{u'\in\mathcal{U}_{bin}} (|u'| - 1)\\ =&(n-q) + (q-1)n - \frac{q(q-1)}{2}\end{aligned}$$

edges[3]. Hence finding a topological sort of G_ψ can be done in time $\mathcal{O}((q-1)n)$.

Remark 5.3.20. As far as we knew up to very soon, the bound of Krasikov and Roditty [62] was the best upper-known bound. However, a wise remark from Tero Harju informed us that a slightly better bound was known [43]: any word of length n can be reconstructed uniquely from its k-deck, with

$$k = 2\lfloor\sqrt{n\ln(2)}\rfloor + 3.$$

However, we decided not to adapt the proofs since a lot of new computations would have been needed in Theorems 5.2.11 and 5.3.17 as well as in related lemmas. We are convinced that our number of queries is still less than the number of Lyndon words up to $2\lfloor\sqrt{n\ln(2)}\rfloor + 3$, since all our maximizations were large. This conviction is witnessed by computations on `Mathematica`. Indeed, set $\alpha'(n) = 2\lfloor\sqrt{n\ln(2)}\rfloor + 3$; the best known bound is obtained by replacing $\alpha^{\lfloor\cdot\rfloor}(n) = \lfloor\frac{16}{7}\sqrt{n}\rfloor + 5$ by $\alpha'(n)$ in (5.9). Moreover we obtained in Theorem 5.3.14 that any word $u \in \mathcal{A}^n$ can be uniquely determined by asking less than qn queries. Hence it suffices to check that

$$f(n,q) := \left(\sum_{i=1}^{\alpha'(n)} \frac{1}{i}\sum_{d|i}\mu(d)\cdot q^{\frac{i}{d}}\right) - qn$$

is positive. That was done for $q \in \{2,\dots,20\}$ and $n \in \{1,\dots,100\}$ with `Mathematica`. We provide in Figure 5.7 graphs for $f(n,2)$, $f(n,3)$ and $f(n,4)$, $n \in [8]$.

[3]This quantity is an upper bound since for any $u' \in \mathcal{U}_{bin}$ and any $a \in \mathcal{A}$, if aa is a factor of u' then it implies that G_ψ contains an edge $((a)_i, (a)_{i+1})$ (for a $i \in [|u|_a - 1]$) that was already counted in the first member of the sum.

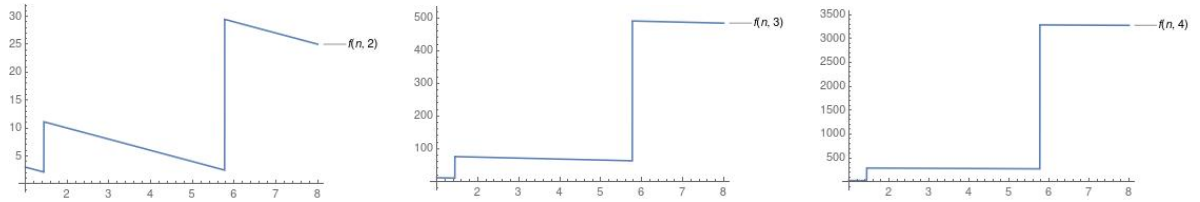


Figure 5.7: Functions $f(n,q)$ for $q \in \{2,3,4\}$ and $n \in [1;8]$.

5.4 Conclusions

Let us first summarize the different results we obtained. We first showed that any word of $\{a,b\}^n$ can be uniquely reconstructed by asking at most $\lfloor \frac{n}{2} \rfloor + 1$ queries, and that these queries are $Q(u,b), Q(u,ab), \cdots, Q(u,a^{|u|_a}b)$ if u contains less a than b, and $Q(u,a), Q(u,ba), \cdots, Q(u,b^{|u|_b}a)$ otherwise.

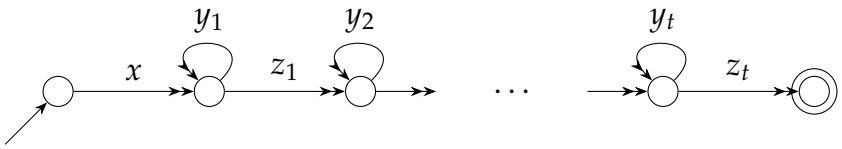
Then for the general case we proved that asking qn queries is always sufficient for reconstructing a word u of length n, where q is the size of the alphabet. Moreover, after asking $q-1$ queries for determining the 1-deck of u, it suffices to re-order the alphabet via a permutation ι such that $a_{\iota(1)}$ (resp., $a_{\iota(q)}$) is the least (resp., most) present letter in u. Then the algorithm is the following: reconstruct all binary projections of u, find their unique K_{min}-marking and then the unique topological path associated to it gives you u.

We can ask a variant of the problem. Up to know, for a fixed n, we were considering the whole set $\mathcal{A}^n$. We could take a rational language L, and consider only words of $L \cap \mathcal{A}^n$. Hence a word u would be uniquely determined in L by asking k queries $Q(u,v_1), \ldots, Q(u,v_k)$ if, for any $u' \in L$ such that $u' \neq u$, we have $\left(\binom{u}{v_1}, \ldots, \binom{u}{v_k}\right) \neq \left(\binom{u'}{v_1}, \ldots, \binom{u'}{v_k}\right)$. As we have seen in Proposition 2.1.13, a regular language has either polynomial or exponential growth. Polynomial languages are known to have interesting properties [119]: any such language L can be written as a finite union of languages of the form

$$xy_1^*z_1 \cdots y_t^*z_t \tag{5.17}$$

(we say that such a language is *t-tiered*). Moreover, if M is a DFA accepting L, then $|x|, |y_1|, \ldots, |y_t|, |z_1|, \ldots, |z_t| < M_q$, where M_q denotes the number of states of M. It is obvious that a language of the type (5.17) is accepted by an automaton whose shape is represented in Figure 5.8.

Hence let us fix a polynomial rational language L. Consider all sublanguages of the form (5.17) and denote by T the minimal natural number such that all these languages are T-tiered. Can we find a bound $C_{T,n}$ (depending on T and n) such that any word of $L \cap \mathcal{A}^n$ can be uniquely determined by asking at most $C_{T,n}$ queries? We



where for every word $u = u_1 \cdots u_n$,

Figure 5.8: An automaton accepting a language of the form $xy_1^*z_1 \cdots y_t^*z_t$.

are naturally interested for a better bound than the one we obtained in this manuscript with $L = \mathcal{A}^*$.

Appendices

A | Coding the templates for the Tribonacci word

Here one can find a `Mathematica` code computing realizable templates for Tribonacci and checking that no template from $\mathcal{T} = \{[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2] : a_1 \neq a_2\}$ is realizable. As explained in Chapter 4, it proves that two different factors of $\mathbf{T}$ are never 2-binomially equivalent. This appendix is divided into several sections, following the structure of Chapter 4. We provide with each function a detailed comment describing the inputs and the output. Some additional comments are sometimes added.

A.1 Basics: Kronecker product and Parikh matrices

We provide the code for generating a long enough prefix of $\mathbf{T}$, computing binomial coefficients, (extended) Parikh vectors and the (extended) Parikh matrix of τ.

`tau`

Input:
`u`, a finite word;

Output:
`tau[u]`, the image by τ of `u`;

Comment:
We also set the variable `tribo` equal to the prefix of $\mathbf{T}$ of length 223 317 and define a shortcut for the empty word ε.

```
> tau[u_] := Flatten[u /. {0 -> {0, 1}, 1 -> {0, 2}, 2 -> {0}}]
> tribo = Nest[tau, {0}, 20];
> ε = {};
```

`factors`

Input:
`i`, a non-negative integer;

Output:
`factors[i]`, the list of factors of **T** of length `i`.

```
> factors[i_] := DeleteDuplicates[Partition[tribo, i, 1]];
```

`coeff`

Inputs:
`u`, a finite word;
`v`, a finite word;

Output:
`coeff[u,v]`, the value of the binomial coefficient $\binom{u}{v}$.

```
> coeff[u_, v_] := coeff[u, v] = If[Length[v] == 0, 1, If[
    Length[u] < Length[v],
    0,
    coeff[Drop[u, -1], v]
      + ((Last[u] == Last[v]) /. {True -> 1, False -> 0})
        * coeff[Drop[u, -1], Drop[v, -1]]
    ]
  ]
```

`psi`

Input:
`u`, a finite word;

Output:
ψ`[u]`, the Parikh vector $\underline{\mathbf{\Psi}}(u)$ of `u`.

```
> ψ[u_] := Map[Count[u, #] &, {0, 1, 2}]
```

`phi`

Input:
`u`, a finite word;

Output:
ϕ`[u]`, the extended Parikh vector $\underline{\mathbf{\Phi}}(u)$ of `u`.

```
> ϕ[u_] := Join[ψ[u], Map[coeff[u, #] &, Tuples[{0, 1, 2}, 2]]]
```

⊗

Inputs:
`a`, a vector;
`b`, a vector;

Output:
`a` ⊗ `b`, the Kronecker product of vectors `a` and `b`;

Comment:
The Kronecker product is a built-in `Mathematica` function, we simply give another notation for the Kronecker product of two lists.

```
> a_List ⊗ b_List := Flatten[KroneckerProduct[a, b]]
```

several matrices

Comment:
We here define matrices P_3, M'_τ and M_τ, as well as their inverses.

```
> P3 = {{0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0,
  0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0,
  0, 0}, {0, 0, 1, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 1, 0, 0, 0, 0, 0}, {0, 0,
  0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0, 0},{0, 0, 0, 0, 0, 0, 1,
  0, 0}, {0, 0, 0, 0, 0, 0, 0, 1, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 1}};
> Mprim = {{1, 1, 1}, {1, 0, 0}, {0, 1, 0}};
> M = {{1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 1, 1, 1, 1, 1,
  1, 1, 1, 1}, {1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0}, {0, 1, 0, 0, 1, 0, 0,
  1, 0, 0, 1, 0}, {0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 1, 0, 0,
  0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0,
  0, 1, 1, 1, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0}, {0, 0, 0, 0,
  0, 0, 0, 1, 0, 0, 0, 0}};
> invMprim = Inverse[Mprim];
> invM = Inverse[M];
```

A.2 Templates and ancestors

We here follow the structure of Section 4.2. We first compute sets $\mathrm{PPref}(\tau^n)$ and $\mathrm{PSuff}(\tau^n)$ for any $n \in \mathbb{N}$. We then compute parents of a template.

`pref`

Input:
`n`, a positive integer;

Output:
`pref[n]`, the list of proper prefixes of $\tau^{\mathtt{n}}$ denoted as $\mathrm{PPref}(\tau^{\mathtt{n}})$ in Chapter 4;

Comment:
It is enough to consider images of 0, due to the special form of the morphism τ.

```
> pref[n_] := Reverse[Table[Drop[#, - i], {i, 1, Length[#]}]
                    &[Nest[tau, {0}, n]]];
```

`suff`

Input:
`n`, a positive integer;

Output:
`suff[n]`, the list of proper suffixes of $\tau^{\mathtt{n}}$ denoted as $\mathrm{PSuff}(\tau^{\mathtt{n}})$ in Chapter 4.

```
> suff[n_] := Sort[DeleteDuplicates[Flatten[
    Table[Table[Drop[#, i], {i, 1, Length[#]}] &[Nest[tau, {i}, n]],
      {i, 0, 2}], 1]]];
```

Definition 4.2.4 gives conditions on templates $t = [\underline{\mathbf{d}}, \underline{\mathbf{D_b}}, \underline{\mathbf{D_e}}, a_1, a_2]$ and $t' = [\underline{\mathbf{d}'}, \underline{\mathbf{D}'_\mathbf{b}}, \underline{\mathbf{D}'_\mathbf{e}}, a'_1, a'_2]$ if we want t' to be a parent by τ of t. Among them, we ask the existence of $s_u, s_v \in \mathrm{PSuff}(\tau)$ such that $a_1 s_u$ is a suffix of $\tau(a'_1)$ and $a_2 s_v$ is a suffix of $\tau(a'_2)$. Hence if $a_1 \neq 0$, the only convenient s_u is ε. In the other cases, s_u can take all the values of $\mathrm{PSuff}(\tau)$. A similar reasoning holds for s_v. The following function simply encodes this result. We then present a function computing, given suitable (p_u, s_u, p_v, s_v) and a template t, the parent of t determined by (p_u, s_u, p_v, s_v).

`findParam`

Inputs:
`a1`, a letter from $\{0, 1, 2\}$;

a2, a letter from $\{0,1,2\}$;

Output:
findParam(a1,a2), a list giving all possible 4-tuples (p_u, s_u, p_v, s_v) for which templates having a_1 and a_2 as fourth and fifth components admit a parent using (p_u, s_u, p_v, s_v).

```
> findParam[a1_, a2_] := Block[{s1, s2},
    If[a1 == 0, s1 = {ε, {1}, {2}}, s1 = {ε}];
    If[a2 == 0, s2 = {ε, {1}, {2}}, s2 = {ε}];
    Tuples[{{ε, {0}}, s1, {ε, {0}}, s2}]
  ]
```

parent

Inputs:
{d,Db,De,a1,a2}, a template;
{pu,su,pv,sv}, a 4-tuple containing proper prefixes and suffixes of τ;

Output:
parent[{d,Db,De,a1,a2}, {pu,su,pv,sv}], the parent template given by Definition 4.2.4.

```
> parent[{d_, Db_, De_, a1_, a2_}, {pu_, su_, pv_, sv_}] :=
  Block[{dprim, deprim, dbprim, aa1, aa2},
    dprim = invM.(d + ψ[Join[pu, su]] - ψ[Join[pv, sv]]
           + P3.(ϕ[pv] ⊗ d[[1 ;; 3]] + d[[1 ;; 3]] ⊗ ϕ[sv])
           - P3.((Db + ϕ[pu] - ϕ[pv]) ⊗ ϕ[Join[pu, su]]
           + ϕ[Join[pu, su]] ⊗ (De + ϕ[su] - ϕ[sv])));
    dbprim = invMprim.(Db + ϕ[pu] - ϕ[pv]);
    deprim = invMprim.(De + ϕ[su] - ϕ[sv]);
    aa1 = Which[
      su == ε, Which[a1 == 0, 2, a1 == 1, 0, a1 == 2, 1],
      su == {1}, 0,
      su == {2}, 1];
    aa2 = Which[
      sv == ε, Which[a2 == 0, 2, a2 == 1, 0, a2 == 2, 1],
      sv == {1}, 0,
      sv == {2}, 1];
    Return[{dprim, dbprim, deprim, aa1, aa2}]
  ]
```

`listParents`

Inputs:
`template`, a template;

Output:
`listParents[template]`, a list of all parents by τ of `template`;

Comment:
As we have seen in Chapter 4, a template can have several parents, depending on the values of $p_u, p_v \in \mathrm{PPref}(\tau)$ and $s_u, s_v \in \mathrm{PSuff}(\tau)$. Hence we take here all parents, ranging on suitable values for p_u, p_v, s_u, s_v given by the function `findParam`.

```
> listParents[template_] :=
  Map[parent[template, #] &, findParam[template[[4]], template[[5]]]]
```

A.3 Bounding realizable templates

Following the results of Section 4.3, we implement constants c_1, c_2 and function f of Lemma 4.3.6, as well as constant $C(\underline{\mathbf{r}})$ from Propositions 4.3.8 and 4.3.9. First of all, let us compute the exact values of eigenvalues of M_τ.

`Eigenvalues and eigenvectors`

Comment:
We compute the exact eigenvalues and associated eigenvectors. `Mathematica` is able to keep the exact value as it is the root of a polynomial. The variable `duosVP` contains the pairs of eigenvalues and their associated eigenvectors.

```
> vp = Eigensystem[Transpose[M]];
> duosVP = Table[{vp[[1, i]], vp[[2, i]]}, {i, 1, Length[vp[[1]]]}];
> θ = duosVP[[2, 1]] ;
> α = 1 / θ;
> β = 1/ (θ^2);
> γ = 1/ (θ^3);
```

One of the eigenvalues (the third one in duosVP) can be approximated by $-0.7718445063 + 1.1151425080\mathtt{i}$. Its multiplicity equals 2. Let us denote by $\underline{\mathbf{v_1}}$ and $\underline{\mathbf{v_2}}$ the two associated eigenvectors. To accelerate the computations, we virtually add

to the vector duosVP a third occurrence of this eigenvalue, associated with the eigenvector $\underline{\mathbf{v_1}} + \underline{\mathbf{v_2}}$. We then modify duosVP to consider numerical approximations of all these eigenvalues and eigenvectors, keeping 10 exact decimal numbers.

```
> AppendTo[duosVP, {duosVP[[3, 1]], duosVP[[3, 2]] + duosVP[[4, 2]]}];
> duosVP = N[duosVP, 10];
```

c1

Inputs:
r, a row vector in $\mathbb{C}^{12}$;
p, a proper prefix of τ^n for a fixed n;
s, a proper suffix of τ^n for the same n;

Output:
c1[r,p,s], the value of the constant $c_1(\mathtt{r},\mathtt{p},\mathtt{s})$ from Lemma 4.3.6.

```
> c1[r_, p_, s_] := Abs[r.(P3.(φ[p] ⊗ { α, β, γ} + { α, β, γ} ⊗ φ[s]))]
```

f

Inputs:
r, a row vector in $\mathbb{C}^{12}$;
p, a proper prefix of τ^n for a fixed n;
s, a proper suffix of τ^n for the same n;
$\delta 0$, a real number;
$\delta 1$, a real number;
$\delta 2$, a real number;

Output:
f[r,p,s,δ0,δ1,δ2], the value of the function $f(\mathtt{r},\mathtt{p},\mathtt{s},\delta 0,\delta 1,\delta 2)$ from Lemma 4.3.6;

Comment:
In practise, $\delta 0$, $\delta 1$ and $\delta 2$ are such that $(\delta 0, \delta 1) \in \Delta$ and $\delta 2 = -\delta 0 - \delta 1$, where Δ is the set defined at the beginning of Subsection 4.3.1.

```
> f[r_, p_, s_, δ0_, δ1_, δ2_] :=
  r.(ψ[Join[p, s]] + (P3.(φ[p] ⊗ {δ0, δ1, δ2} + {δ0, δ1, δ2} ⊗ φ[s])))
```

l

Inputs:
i, an integer from $\{0,1,2\}$;

p, a proper prefix of τ^n for a fixed n;
s, a proper suffix of τ^n for the same n;
dens, a real number;

Output:
l[i,p,s,dens], a real number;

Comment:
The value of l[i,p,s,dens] is the one needed for the computation of l_i in Lemma 4.3.6. The variable dens represents the density α_i of letter i in τ.

```
> l[i_, p_, s_, dens_] := Max[Map[
    dens * Length[Join[ #[[1]], #[[2]] ]]
      - Count[ Join [#[[1]], #[[2]] ], i] &,
    Tuples[{Table[Drop[p, j], {j, 0, Length[p]}],
    Table[Drop[s, - j], {j, 0, Length[s]}]}]
  ]] - 3 / 2
```

u

Inputs:
i, an integer from $\{0,1,2\}$;
p, a proper prefix of τ^n for a fixed n;
s, a proper suffix of τ^n for the same n;
dens, a real number;

Output:
u[i,p,s,dens], a real number;

Comment:
The value of u[i,p,s,dens] is the one needed for the computation of u_i in Lemma 4.3.6. The variable dens represents the density α_i of letter i in τ.

```
> u[i_, p_, s_, dens_] := Min[Map[
    dens * Length[Join[ #[[1]], #[[2]] ]]
      - Count[ Join [#[[1]], #[[2]] ], i] &,
    Tuples[{Table[Drop[p, j], {j, 0, Length[p]}],
    Table[Drop[s, - j], {j, 0, Length[s]}]}]
  ]] + 3 / 2
```

c2

Inputs:
r, a row vector in $\mathbb{C}^{12}$;
p, a proper prefix of τ^n for a fixed n;
s, a proper suffix of τ^n for the same n;

Output:
c2[r,p,s], the value of the constant $c_2(\mathtt{r},\mathtt{p},\mathtt{s})$ from Lemma 4.3.6;

Comment:
We used the development carried on right after the proof of Lemma 4.3.6.

```
> c2[r_, p_, s_] := Module[{l0, l1, l2, u0, u1, u2},
  l0 = l[0, p, s, α];
  l1 = l[1, p, s, β];
  l2 = l[2, p, s, γ];
  u0 = u[0, p, s, α];
  u1 = u[1, p, s, β];
  u2 = u[2, p, s, γ];
  Return[Min[
    Max[Map[Abs[f[r, p, s, #[[1]], #[[2]], - #[[1]] - #[[2]]]] &,
      Tuples[{{l0, u0}, {l1, u1}}]]],
    Max[Map[Abs[f[r, p, s, #[[1]], - #[[1]] - #[[2]], #[[2]]]] &,
      Tuples[{{l0, u0}, {l2, u2}}]]],
    Max[Map[Abs[f[r, p, s, - #[[1]] - #[[2]], #[[1]], #[[2]]]] &,
      Tuples[{{l1, u1}, {l2, u2}}]]],
    Max[Map[Abs[f[r, p, s, #[[1]], #[[2]], #[[3]]]] &,
      Tuples[{{l0, u0}, {l1, u1}, {l2, u2}}]]]
  ]]
```

Since all our computations are concerned with modulus of complex numbers and not the complex numbers themselves, we do not need to consider all the eigenvalues. Indeed, if λ was already treated, we don't have to carry on the eigenvalue equal to the conjugate value of λ. This is why we just have to look on components number $7, 9, 11, 12$ for eigenvalues of modulus less than 1, and on components number $3, 4, 13$ for eigenvalues of modulus greater than 1. The computation of $C(\underline{\mathbf{r}})$ for eigenvalues of modulus greater than 1 as done in Proposition 4.3.8 involves several intermediate computations: computing the maximum value of $|\underline{\mathbf{r}} \cdot \psi(u)|/|u|$ over all factors of $\mathbf{T}$ of length at most ℓ, computing the constant $c_3(\underline{\mathbf{r}})$, computing $\iota(\ell, n)$,...

This is done in the following functions.

`smallW`

Inputs:
`r`, a row vector in $\mathbb{C}^{12}$;
`l`, a positive integer;

Output:
`small[r,l]`, the maximum value of $|\mathtt{r} \cdot \psi(u)|/|u|$ over all factors of $\mathbf{T}$ of length at most `l`.

```
> smallW[r_, l_] := Max[Table[Max[Map[N[Abs[r.ψ[#]] / i, 10] &,
    factors[i]]], {i, 1, l} ]]
```

`ι`

Inputs:
`l`, a positive integer;
`n`, a positive integer;

Output:
`ι[l,n]`, an approximation of $\iota(\mathtt{l},\mathtt{n})$ given in the proof of Proposition 4.3.8.

```
> ι[l_, n_] := N[l / (l + θ^n * (2 + (3 / 2) / (θ - 1))), 10]
```

`lMin`

Inputs:
`n`, a positive integer;
λ, a complex number;

Output:
`lMin[n,λ]`, a real number which is the smallest ℓ satisfying Conditions (4.7) for fixed `n` and λ.

```
> lMin[n_, λ_] := Module[{l},
    l = Ceiling[θ^n * (2 + (3 / 2) / (θ - 1))];
    While[Abs[λ]^n / (ι[l, n] * θ^n) ≥ 1, l ++];
    Return[l];
  ]
```

c3

Inputs:
r, a row vector in $\mathbb{C}^{12}$;
n, a positive integer;
l, a positive integer;

Output:
c3[r,n,l], the function defined in the proof of Proposition 4.3.8 and used to compute $C(\mathtt{r})$.

```
> c3[r_, n_, l_] :=
  Max[Map[c1[r, #[[1]], #[[2]]] + 1 / l * c2[r, #[[1]], #[[2]]]&,
          Tuples[{pref[n], suff[n]}]]]
```

Finally, the following function computes the value of the bound $C(\underline{\mathbf{r}})$. It depends on the chosen values of n and ℓ, as explained in the proof of Proposition 4.3.8. This function should not be used if ℓ is less than lMin[n,λ]. We compute the values of the bounds using $n = 6$ and $\ell = 600$. For information, lMin[6,λ]$= 147$ for all λ of modulus greater than 1. The second function returns pairs of elements of the form $\{n, \text{bound}\}$. This form will be useful later on. We add 10^{-7} to the bounds, since they are just approximations (which are exact up to 7 decimal digits). Since the computations are quite long, the last command allows to stock these values in an out file.

Cgreat

Inputs:
λ, a complex number;
r, a row vector in $\mathbb{C}^{12}$;
n, a positive integer;
l, a positive integer;

Output:
Cgreat[λ,r,n,l], the value of the constant $C(\mathtt{r})$ of Proposition 4.3.8, with λ, n and l fixed;

Comment:
As specified before, this function should not be used with ℓ less than lMin[n,λ].

```
> Cgreat[λ_, r_, n_, l_] :=
```

```
  Module[{temp = ι[l, n] * θ^n, pt = smallW[r, l]},
    Max[pt, Abs[λ]^n / temp * pt + c3[r, n, l] * temp / (temp - Abs[λ]^n)]
  ]
```

CgreatList

Comment:

`CgreatList` is set to the list of pairs containing the different bounds associated to the different eigenvectors, kept together with the value of n. We export this list and give the possibility to import again.

```
> CgreatList =
  Table[{6, Cgreat[duosVP[[i, 1]], duosVP[[i, 2]], 6, 600] + 10^(-7)},
    {i, {3, 4, 13}}]
[Out]: {{6, 3.8809965}, {6, 3.11975700}, {6, 2.46598485}}
> Export["CgreatList_n6l600.m", CgreatList];
> CgreatList = Import["CgreatList_n6l600.m"];
```

We now implement the computation of bound $C(\underline{\mathbf{r}})$ when $\underline{\mathbf{r}}$ is associated to an eigenvalue λ of modulus less than 1, as treated in Proposition 4.3.9. As in the previous case, this bound depends on the value of the positive integer n that we choose. As the computations are quite fast, we chose a better bound by allowing to take different values of n for different values of λ. The second function, for a given λ and its associated eigenvector, computes the best bound for different values of n, ranging from `nMin` to `nMax`. It returns the pair $\{n, \text{bestBound}\}$. We then compute the values of the bounds, choosing the best n between 1 and 6. We add 10^{-7} to the bounds, since they are just approximations (which are exact up to 7 decimal digits). We then export them in an out file.

Csmall

Inputs:

λ, a complex number;
`r`, a row vector in $\mathbb{C}^{12}$;
`n`, a positive integer;

Output:

`Csmall[λ,r,n]`, the value of the constant $C(\mathtt{r})$ of Proposition 4.3.9 with λ and `n` fixed.

```
> Csmall[λ_, r_, n_] := Max[
    Map[c2[r, #[[1]], #[[2]]] &, Tuples[{pref[n], suff[n]}]]
  ] / (1 - Abs[λ]^n)
```

`bestCsmall`

Inputs:
λ, a complex number;
`r`, a row vector in $\mathbb{C}^{12}$;
`nMin`, a positive integer;
`nMax`, a positive integer strictly greater than `nMin`;

Output:
`bestCsmall[`λ`,r,nMin,nMax]`, a pair whose first element is the integer $n \in$ `[nMin,nMax]` for which the bound $C($`r`$)$ (returned as second argument) is the best.

```
> bestCsmall[λ_, r_, nMin_, nMax_] := Module[{listBounds, pos},
    listBounds = Table[Csmall[λ, r, n], {n, nMin, nMax}];
    pos = Ordering[listBounds, 1][[1]];
    Return[{pos + nMin - 1, listBounds[[pos]]}];
  ]
```

`CsmallList`

Comment:
`CsmallList` is set to the list of pairs containing the different bounds associated to the different eigenvectors, kept together with the value of n, that is chosen to be the best one between 1 and 6.

```
> CsmallList =
  Table[bestCsmall[duosVP[[i, 1]], duosVP[[i, 2]], 1, 6],
      {i, {7, 9, 11, 12}}]
> CsmallList = Map[{#[[1]], #[[2]] + 10^(-7)} &, CsmallList];
[Out]: {{6, 2.19301322}, {6, 3.805281713},
        {6, 3.18002470}, {6, 2.762848807}}
> Export["CsmallList.m", CsmallList];
> CsmallList = Import["CsmallList.m"];
```

A.4 Bounds on templates

We now follow the reasoning of Subsection 4.3.2 and compute bounds giving necessary conditions on templates to be realizable by τ. We saw that our bounds have 7 exact decimal digits. We thus check in the following that we continue keeping 7 exact decimal digits. Otherwise the program will report an error (but still continue its execution).

`fbis`

Inputs:
`r`, a row vector in $\mathbb{C}^{12}$;
`t`, a template;
$\delta 0$, a real number;
$\delta 1$, a real number;
$\delta 2$, a real number;

Output:
`fbis[r,t,`$\delta 0$`,`$\delta 1$`,`$\delta 2$`]`, the value that appears in Lemmas 4.3.10 and 4.3.11;

Comment:
This function will simplify the carried on computations.

```
> fbis[r_, t_, δ0_, δ1_, δ2_] :=
  r.(t[[1]] + (P3.(t[[2]] ⊗ {δ0, δ1, δ2} + {δ0, δ1, δ2} ⊗ t[[3]])))
```

`smallBound`

Inputs:
`t`, a template;
λ, a complex number that is an eigenvalue of M_τ;
`r`, a row vector in $\mathbb{C}^{12}$;
`n`, a positive integer;
`Csmall`, a bound verifying Proposition 4.3.9 with λ and `r`;

Output:
`smallBound[t,`λ`,r,n,Csmall]`, a boolean that is `True` if `t` is a valid (i.e., possibly realizable) template, `False` otherwise;

Comment:
This function verifies that the template `t` is valid regarding the bound $C(\mathtt{r})$ associated to the eigenvalue λ and the corresponding eigenvector `r`, according Lemma 4.3.10. We make use of the remark right after the proof of this lemma.

```
> smallBound[t_, λ_, r_, n_, Csmall_] :=Module[
    {aRe, bRe, aIm, bIm, minRe, minIm},
    {aRe, bRe, aIm, bIm} =
      Through[{Min[Re[#]] &, Max[Re[#]] &, Min[Im[#]] &, Max[Im[#]] &}[
        Map[fbis[r, t, #[[1]], #[[2]], - #[[1]] - #[[2]]] &,
```

```
        {{3 / 2, 0}, {3 / 2, - 3 / 2}, {0, 3 / 2},
         {0, - 3 / 2}, {- 3 / 2, 3 / 2}, {- 3 / 2, 0}}]];
   minRe = Which[aRe ≥ 0, aRe^2, bRe ≤ 0, bRe^2, True, 0];
   minIm = Which[aIm ≥ 0, aIm^2, bIm ≤ 0, bIm^2, True, 0];
   If[Accuracy[Sqrt[minRe + minIm]] < 7, Print["Accuracy problem"]];
   Return[Sqrt[minRe + minIm] ≤ 2 * Csmall];
 ]
```

greatBound

Inputs:
t, a template;
λ, a complex number that is an eigenvalue of M_τ;
r, a row vector in $\mathbb{C}^{12}$;
n, a positive integer;
Cgreat, a bound verifying Proposition 4.3.8 with λ and r;
L, a positive integer;

Output:
greatBound[t,λ,r,n,Cgreat,L], a boolean that is True if t is a valid (i.e., possibly realizable) template, False otherwise;

Comment:
This function verifies that the template t is valid regarding the bound $C(\mathtt{r})$ associated to the eigenvalue λ and the corresponding eigenvector r, according Lemma 4.3.11. We make use of the remark right after the proof of this lemma.

```
> greatBound[t_, λ_, r_, n_, Cgreat_, L_] := Module[{lhs, rhs},
   lhs = Abs[r.(P3.(t[[2]] ⊗ {α, β, γ} + {α, β, γ} ⊗ t[[3]]))];
   rhs = (2 * L - Total[t[[1]][[1 ;; 3]]]) / L * Cgreat
     + Max[Map[Abs[fbis[r, t, #[[1]], #[[2]], - #[[1]] - #[[2]]]] &,
       {{3 / 2, 0}, {3 / 2, - 3 / 2}, {0, 3 / 2}, {0, - 3 / 2},
        {- 3 / 2, 3 / 2}, {- 3 / 2, 0}}]] / L;
   If[Accuracy[lhs] < 7 || Accuracy[rhs] < 7, Print["Accuracy problem"]];
   Return[lhs ≤ rhs];
 ]
```

A.5 Computing the bounded ancestors of templates $[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2]$ with $a_1 \neq a_2$

This section consists in computing all the ancestors of templates $[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2]$ with $a_1 \neq a_2$ that are bounded by the constants we just computed, and to show that there are just a finite number of them.

`validTemplate`

Inputs:
`template`, a template;
`CsmallList`, a list of pairs of the type $\{n$, `bound`$\}$;
`CgreatList`, a list of pairs of the type $\{n$, `bound`$\}$;
`L`, a positive integer;

Output:
`validTemplate[template, CsmallList, CgreatList]`, a boolean that is `True` if `template` verifies all the bounds, `False` otherwise;

Comment:
`CsmallList` contains all bounds related to eigenvalues of modulus less than 1 when `CgreatList` contains bounds related to eigenvalues of modulus between 1 and θ.

```
> validTemplate[template_, CsmallList_, CgreatList_, L_] := Which[
    Not[smallBound[template, duosVP[[7, 1]], duosVP[[7, 2]],
      CsmallList[[1]][[1]], CsmallList[[1]][[2]]]], False,
    Not[smallBound[template, duosVP[[9, 1]], duosVP[[9, 2]],
      CsmallList[[2]][[1]], CsmallList[[2]][[2]]]], False,
    Not[smallBound[template, duosVP[[11, 1]], duosVP[[11, 2]],
      CsmallList[[3]][[1]], CsmallList[[3]][[2]]]], False,
    Not[smallBound[template, duosVP[[12, 1]], duosVP[[12, 2]],
      CsmallList[[4]][[1]], CsmallList[[4]][[2]]]], False,
    Not[greatBound[template, duosVP[[3, 1]], duosVP[[3, 2]],
      CgreatList[[1]][[1]], CgreatList[[1]][[2]], L]], False,
    Not[greatBound[template, duosVP[[4, 1]], duosVP[[4, 2]],
      CgreatList[[2]][[1]], CgreatList[[2]][[2]], L]], False,
    Not[greatBound[template, duosVP[[13, 1]], duosVP[[13, 2]],
      CgreatList[[3]][[1]], CgreatList[[3]][[2]], L]], False,
    True, True]
```

`valid`

Inputs:
`templatesList`, a list of templates;
`CsmallList`, a list of pairs of the type $\{n, \texttt{bound}\}$;
`CgreatList`, a list of pairs of the type $\{n, \texttt{bound}\}$;
`L`, a positive integer;

Output:
`valid[templatesList, CsmallList, CgreatList, L]`, a list of templates;

Comment:
The function takes a list of templates and keeps only the valid ones.

```
> valid[templatesList_, CsmallList_, CgreatList_, L_] :=
  valid[templatesList, CsmallList, CgreatList, L] =
  Select[templatesList, validTemplate[#, CsmallList, CgreatList, L] &]
```

Finally, this function is the one computing all valid ancestors of templates $[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2]$ with $a_1 \neq a_2$. We assume that `seenTemplates` and `toSeeTemplates` are two global list variables. The first one is initially set at $\{\,\}$ while the second one initially contains the six templates from which we start.

`computingAncestors`

Inputs:
`CsmallList`, a list of pairs of the type $\{n, \texttt{bound}\}$;
`CgreatList`, a list of pairs of the type $\{n, \texttt{bound}\}$;
`L`, a positive integer;

Output:
`computingAncestors[CsmallList, CgreatList, L]`, the list of all valid ancestors of the ones initially in the global variable `toSeeTemplates`;

Comment:
After defining the function, we compute all valid ancestors of $\mathcal{T} = \{[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2] : a_1 \neq a_2\}$ and export them in an out file.

```
> computingAncestors[CsmallList_, CgreatList_, L_] :=
  Module[{currentTemplate},
    While[Length[toSeeTemplates] > 0,
```

```
      currentTemplate = toSeeTemplates[[1]];
      toSeeTemplates = Drop[toSeeTemplates, 1];
      toSeeTemplates = Union[toSeeTemplates, Complement[valid[
        listParents[currentTemplate], CsmallList, CgreatList, L],
        seenTemplates]];
      AppendTo[seenTemplates, currentTemplate]
    ]
  ]
> toSeeTemplates = List[
    List[ConstantArray[0,12], ConstantArray[0,3], ConstantArray[0,3], 0, 1],
    List[ConstantArray[0,12], ConstantArray[0,3], ConstantArray[0,3], 0, 2],
    List[ConstantArray[0,12], ConstantArray[0,3], ConstantArray[0,3], 1, 0],
    List[ConstantArray[0,12], ConstantArray[0,3], ConstantArray[0,3], 1, 2],
    List[ConstantArray[0,12], ConstantArray[0,3], ConstantArray[0,3], 2, 0],
    List[ConstantArray[0,12], ConstantArray[0,3], ConstantArray[0,3], 2, 1]
  ];
> seenTemplates = {};
> L = 15; (* This value can be adapted *)
> computingAncestors[CsmallList, CgreatList, L];
> Export["seenTemplates.m", seenTemplates];
> Print["Number of valid ancestors : ", Length[seenTemplates]]
[Out]: Number of valid ancestors : 241544
```

A.6 Conclusion: factor complexity equals 2-binomial complexity

We just obtained a finite number of bounded ancestors. We want to conclude by Lemma 4.2.3. Then, making use of Proposition 4.2.9, if templates $[\underline{\mathbf{0}}, \underline{\mathbf{0}}, \underline{\mathbf{0}}, a_1, a_2]$ are realizable, either there are realizable by a pair (u, v) of factors of Tribonacci with $\min(|u|, |v|) \leq L$, or one of their ancestors is realizable by a pair (u, v) of factors with $L \leq \min(|u|, |v|) \leq 2L$. The first function deals with the first case. Since these templates are such that $\underline{\mathbf{d}} = \underline{\mathbf{0}}$, $\underline{\mathbf{D_b}} = \underline{\mathbf{0}}$, $\underline{\mathbf{D_e}} = \underline{\mathbf{0}}$, (u, v) realizes one of them if and only if $\underline{\mathbf{\Phi}}(u) = \underline{\mathbf{\Phi}}(v)$. We thus check that all words of length at most L are non pairwise 2-binomially equivalent.

smallWordsNonEquiv

Input:
L, a positive integer;

Output:
`smallWordsNonEquiv[L]`, a boolean that is `True` if all factors of **T** of length at most `L` are pairwise non-2-binomially equivalent, `False` otherwise.

```
> smallWordsNonEquiv[L_] := Module[{i, j, k, words},
    For[i = 1, i ≤ L, i ++,
      words = factors[i];
      For[j = 1, j ≤ 2 * i + 1, j ++,
        For[k = j + 1, k ≤ 2 * i + 1, k ++,
          If[φ[words[[j]]] == φ[words[[k]]], Return[False]]]]];
    Return[True];
    ]
```

`verif`

Inputs:
`lu`, a list of extended Parikh vectors;
`lv`, a list of extended Parikh vectors;
`t`, a template;

Output:
`verif[lu,lv,t]`, a boolean that is `True` if no pair of words (u, v) realizes t, where $\underline{\mathbf{\Phi}}(u)$ is an element of `lu` and $\underline{\mathbf{\Phi}}(v)$ is an element of `lv`, `False` otherwise.

```
> verif[lu_, lv_, t_] := Not[MemberQ[
    Map[ #[[1]] - #[[2]] ==
        t[[1]] + P3. (t[[2]] ⊗ #[[1]][[1 ;; 3]]
                + #[[1]][[1 ;; 3]] ⊗ t[[3]])&,
      Tuples[{lu, lv}]], False]
  ]
```

`conjecture`

Inputs:
`L`, a positive integer;
`seenTemplates`, a list of templates;

Output:
`conjecture[L,seenTemplates]`, a boolean that is `True` if no template from the list is realizable by a pair (u, v) of factors of Tribonacci such that $L \leq \min(|u|, |v|) \leq 2L$;

Comment:
To accelerate the verification, we compute the maximal length of $|u|$ and $|v|$ and we stock all possible values of $\underline{\mathbf{\Phi}}(u)$ and $\underline{\mathbf{\Phi}}(v)$.

```
> F[L_, seenTemplates_] := Module[
  {listlistpsi, j, i, fac, currentTemplate, k, lgV, lu, lv, total, Dd},

  (* Maximal length of |u| and |v| *)
  Dd = Max[Map[Abs[Total[#[[1]][[1 ;; 3]]]] &, seenTemplates]];

  (* Generation of lists containing φ[u] and φ[v]. *)
  listlistpsi = {};
  For[j = L, j ≤ 2 L + Dd, j ++,
    fac = factors[j];
    AppendTo[listlistpsi, List[
      Map[φ[#] &, Select[fac, #[[- 1]] == 0 &]],
      Map[φ[#] &, Select[fac, #[[- 1]] == 1 &]],
      Map[φ[#] &, Select[fac, #[[- 1]] == 2 &]]
    ]]
  ]

  (* Looking at every template separately *)
  For[j = 1, j ≤ Length[seenTemplates], j ++,
    currentTemplate = seenTemplates[[j]];
    total = Total[currentTemplate[[1]][[1 ;; 3]]];

    (* Two cases depending on if |u| <|v| or not. *)
    If[total ≥ 0,
      For[i = 1, i ≥ L + 1, i ++,
        If[verif[listlistpsi[[i + total, currentTemplate[[4]] + 1]],
            listlistpsi[[i, currentTemplate[[5]] + 1]],
            currentTemplate] == False,
          Return[False]
        ]
      ],
      For[i = 1, i ≤ L + 1, i ++,
        If[verif[listlistpsi[[i, currentTemplate[[4]] + 1]],
            listlistpsi[[i - total, currentTemplate[[5]] + 1]],
            currentTemplate] == False,
```

```
        Return[False]
      ]
    ]
  ]
];
Return[True];
]
```

Finally, these instructions check that Proposition 4.2.9 can be applied, as explained earlier. The value of L can change, but has to be equal to the one chosen in the verification of the greater bounds. We recall that `seenTemplates` is a global variable.

```
> L = 15;
> smallWordsNonEquiv[L]
[Out]: True
> conjecture[L, seenTemplates]
[Out]: True
```

B | k-binomial complexity of non-N-balanced Arnoux–Rauzy words

As stated in Section 4.5, there exists a sequence $(\mathbf{w_n})_{n\geq 2}$ of Arnoux–Rauzy words such that for any n, $\mathbf{w_n}$ is non-$(n-1)$-balanced. This result was seen as quite surprising since most of the known Arnoux–Rauzy words are balanced. Using the results from [21] it is possible to give an algorithm generating such a sequence. The goal of this appendix is first to give this algorithm, following [21], and then to compute the first values of $b^{(2)}_{\mathbf{w_n}}$ for the first words of this sequence.

Let us thus build a sequence of infinite words $(\mathbf{w_n})_{n\geq 2}$ such that, for any $n \geq 2$, $\mathbf{w_n}$ is an Arnoux–Rauzy word that is not $(n-1)$-balanced. It means that, for any $n \geq 2$, there exist u_n and v_n, factors of $\mathbf{w_n}$, such that

$$\exists\, a \in \mathcal{A} \,:\, ||u_n|_a - |v_n|_a| \geq n.$$

We will call u_n and v_n the *witnesses (of the non-$(n-1)$-balancedness of $\mathbf{w_n}$)*.

We define the *standard Arnoux–Rauzy morphisms*:

$$\sigma_0 : \begin{cases} 0 \mapsto 0; \\ 1 \mapsto 01; \\ 2 \mapsto 02, \end{cases} \quad \sigma_1 : \begin{cases} 0 \mapsto 10; \\ 1 \mapsto 1; \\ 2 \mapsto 12, \end{cases} \quad \sigma_2 : \begin{cases} 0 \mapsto 20; \\ 1 \mapsto 21; \\ 2 \mapsto 2. \end{cases}$$

Proposition B.1 ([21]). *For any $N \in \mathbb{N}$, there exists a primitive morphism $\sigma_{[N]}$ such that for any Arnoux–Rauzy word $\mathbf{w}$, $\sigma_{[N]}(\mathbf{w})$ is not $(N-1)$-balanced. Moreover, $\sigma_{[2]}$ can be chosen as*

$$\sigma_{[2]} = \sigma_0 \circ \sigma_1 : \begin{cases} 0 \mapsto 010; \\ 1 \mapsto 01; \\ 2 \mapsto 0102. \end{cases}$$

This proposition allows us to build $\mathbf{w_2}$. For example we can take

$$\mathbf{w_2} = \sigma_{[2]}(\mathbf{T}),$$

and we obtain as witnesses $u_2 = 101$ and $v_2 = 020$.

We will show that $\sigma_{[N+1]}$, u_{N+1} and v_{N+1} can be built from $\sigma_{[N]}$, u_N and v_N, hence we will define $\mathbf{w_n} = \sigma_{[n]}(\mathbf{T})$ for all $n \geq 2$.

Proposition B.2 ([21]). *If $\mathbf{w_N}$ is not $(N-1)$-balanced and if u_N and v_N are two witnesses of its non-$(N-1)$-balancedness, then there exists a permutation (i_N, j_N, k_N) of $(0,1,2)$ such that*

$$\begin{pmatrix} \big|\, |u_N|_{i_N} - |v_N|_{i_N} \,\big| \\ \big|\, |u_N|_{j_N} - |v_N|_{j_N} \,\big| \\ \big|\, |u_N|_{k_N} - |v_N|_{k_N} \,\big| \end{pmatrix} = \begin{pmatrix} 1 \\ n \\ n-1 \end{pmatrix}.$$

We can now get the construction of $\sigma_{[N+1]}$ from $\sigma_{[N]}$.

Proposition B.3 ([21]). *Let $\mathbf{w_N}$ be a non-$(N-1)$-balanced Arnoux–Rauzy word and u_N, v_N be witnesses. Let (i_N, j_N, k_N) be taken as in the previous proposition. Then*

$$\sigma_{[N+1]} := \sigma^N_{k_N} \circ \sigma^N_{i_N} \circ \sigma_{[N]}$$

satisfies Proposition B.1.

We assumed that (i_N, j_N, k_N) is known. We have

$$(i_2, j_2, k_2) = (0,1,2) \text{ or } (2,1,0).$$

We are now going to highlight the fact that $(i_{N+1}, j_{N+1}, k_{N+1})$ can be obtained from (i_N, j_N, k_N). Let $a \in \mathcal{A}$ and define

$$\begin{aligned} \sigma_{(a,+)} &: \mathcal{A}^* \to \mathcal{A}^* : u \mapsto \sigma_a(u)a, \\ \sigma_{(a,-)} &: \mathcal{A}^* \to \mathcal{A}^* : u \mapsto a^{-1}\sigma_a(u), \end{aligned}$$

where σ_a is one of the standard Arnoux–Rauzy morphisms.

Proposition B.4 ([21]). *Let us assume that $\mathbf{w_N}$ is a non-$(N-1)$-balanced Arnoux–Rauzy word having u_N and v_N as witnesses and that $\sigma_{[N+1]}$ is known. Set $\mathbf{w_{N+1}} = \sigma_{[N+1]}(\mathbf{T})$ and denote by u_{N+1} and v_{N+1} its witnesses. Let finally (i_N, j_N, k_N) be the permutation related to $\mathbf{w_N}$ given in Proposition B.2. Then we have*

$$u_{N+1} = \sigma^N_{(k_N,-)} \circ \sigma^N_{(i_N,+)}(v_N)$$

and

$$v_{N+1} = \sigma^N_{(k_N,+)} \circ \sigma^N_{(i_N,-)}(u_N).$$

Moreover if $(i_{N+1}, j_{N+1}, k_{N+1})$ denotes the permutation of Proposition B.2 related to $\mathbf{w_{N+1}}$, then $(i_{N+1}, j_{N+1}, k_{N+1}) = (k_N, i_N, j_N)$.

Proposition B.5. *The sequence $(\mathbf{w_n})_{n\geq 2} = (\sigma_{[n]}(\mathbf{T}))_{n\geq 2}$ is such that $\mathbf{w_n}$ is not $(n-1)$-balanced for any $n \geq 2$.*

This result is direct from Proposition B.1 and we just gave an algorithm computing this sequence: from $\sigma_{[N]}$, u_N, v_N and (i_N, j_N, k_N) you can compute $\sigma_{[N+1]}$ via Proposition B.3 and u_{N+1}, v_{N+1}, $(i_{N+1}, j_{N+1}, k_{N+1})$ via Proposition B.4.

`Mathematica` code

We are here giving a `Mathematica` code that computes prefixes of the first words $\mathbf{w_n}$, as well as the first values of their 2-binomial complexity function. As in the previous appendix, we provide with each function a detailed comment describing the inputs and the output. Some additional comments are sometimes added.

`coeff`

Inputs:
`u`, a finite word;
`v`, a finite word;

Output:
`coeff[u,v]`, the value of the binomial coefficient $\binom{u}{v}$.

Comment:
This function was already defined in Appendix A, but we recall it here for the sake of completeness.

```
> coeff[u_, v_] := coeff[u, v] = If[Length[v] == 0, 1, If[
    Length[u] < Length[v],
    0,
    coeff[Drop[u, -1], v]
      + ((Last[u] == Last[v]) /. {True -> 1, False -> 0})
        * coeff[Drop[u, -1], Drop[v, -1]]
    ]
  ]
```

`carb2`

Inputs:
`u`, a finite word;
`m`, a positive integer;

Output:
`cab2[u,m]`, a vector that encodes all binomial coefficients $\binom{u}{v}$, for $v \in \mathcal{A}^{\leq 2}$ where $\mathcal{A} = \{0, \ldots, m-1\}$;

Comment:
Two words `u` and `u'` over $\{0, \ldots, m-1\}$ are 2-binomially equivalent if and only if `carb2[u,m]` and `carb2[u',m]` return the same vector.

```
> carb2[u_, m_] := Map[coeff[u, #] &,
    Join[Partition[Range[0, m - 1], 1],
      Flatten[Table[List[i, j], {i, 0, m - 1}, {j, i + 1, m - 1}], 1 ]]]
```

`factors`

Inputs:
`w`, a finite word;
`n`, a non-negative integer;

Output:
`factors[w,n]`, a list of all factors of length `n` that are present in `w`.

Comment:
If `w` is a long enough prefix of a given infinite word, we can assume that `factors[w,n]` returns the whole set of factors of length `n` of this infinite word. Moreover, we will only work with Arnoux–Rauzy words so we know that we can stop the program if we already collected $2n+1$ factors of length `n`. It increases the rapidity of the code when we consider really long prefixes (more than three millions of letters).

```
> factors[w_, n_] := Module[{l, m, word},
    l = List[];
    m = 1;
    While [m <= Length[w] - n && Length[l] < 2*n + 1,
      word = Take[w, {m, m + n - 1}];
      If [Not[MemberQ[l, word]], AppendTo[l, word]];
      m = m + 1
    ];
    Return[l]
  ]
```

`b2`

Inputs:
`w`, a finite word;
`n`, a non-negative integer;
`m`, a positive integer;

Output:
Admitting that `w` is a long enough prefix of an infinite word $\mathbf{x}$ over $\{0,\dots,m-1\}$, `b2[w,n,m]` returns the value in `n` of the 2-binomial complexity of the infinite word $\mathbf{x}$.

```
> b2[w_, n_, m_] := Length[Union[Map[carb2[#, m] &, factors[w, n]]]]
```

p

Inputs:
`w`, a finite word;
`n`, a non-negative integer;

Output:
Admitting that `w` is a long enough prefix of an infinite word $\mathbf{x}$, `p[w,n]` returns the value in `n` of the factor complexity of the infinite word $\mathbf{x}$.

Comment:
As we are going to work with prefixes of Arnoux–Rauzy words, computing the factor complexity of their prefixes and checking that it equals $2n+1$ ensures that the prefix we consider is long enough.

```
> p[w_, n_] := Length[factors[w, n]]
```

s2

Input:
`u`, a finite word;

Output:
`s2[u]`, the image by $\sigma_{[2]}$ (given in Proposition B.1) of `u`.

```
> s2[u_] := Flatten[u /. {0 -> {0, 1, 0}, 1 -> {0, 1}, 2 -> {0, 1, 0, 2}}]
```

stMor

Input:
`i`, an integer from $\{0,1,2\}$;

Output:
`stMor[i]`, a function which is the standard morphism $\sigma_{\mathtt{i}}$.

```
> stMor[i_] := Function[{word}, Flatten[Map[
    Function[{letter}, If[letter == i, List[i], List[i, letter]]],
     word], 1]]
```

compose

Inputs:
`m1`, a morphism;

`m2`, a morphism defined on the same alphabet as `m1`;

Output:
`compose[m1,m2]`, a morphism which is `m1` $\circ$ `m2`.

```
> compose[m1_, m2_] := Function[{mot}, m1[m2[mot]]]
```

power

Inputs:
`m`, a morphism;
`n`, a positive integer;

Output:
`power[m,n]`, a morphism which is $\mathtt{m}^{\mathtt{n}}$.

```
> power[m_, n_] := If[n == 1, m, compose[m, power[m, n - 1]]]
```

plus_s

Input:
`i`, an integer from $\{0,1,2\}$;

Output:
`plus_s[i]`, a morphism which is $\sigma_{(\mathtt{i},+)}$.

```
> plus_s[i_] := Function[{word}, Append[stMor[i][word], i]]
```

minus_s

Input:
`i`, an integer from $\{0,1,2\}$;

Output:
`minus_s[i]`, a morphism which is $\sigma_{(\mathtt{i},-)}$.

```
> minus_s[i_] := Function[{word}, Drop[stMor[i][word], 1]]
```

nextMorphism

Inputs:
`s`, a morphism over $\{0,1,2\}$;
$\{\mathtt{i},\mathtt{j},\mathtt{k}\}$, a permutation of $\{0,1,2\}$;
$\{\mathtt{u},\mathtt{v}\}$, two words over $\{0,1,2\}$;

n, an integer greater than 1;

Output:
nextMorphism[s,i,j,k,u,v,n] returns a list of three elements:
- a new morphism over $\{0,1,2\}$;
- a permutation of $\{0,1,2\}$;
- a list of two words over $\{0,1,2\}$.

Comment:
Assume that s is the morphism $\sigma_{[n]}$, $\{i,j,k\}$ is the sequence $\{i_n, j_n, k_n\}$ and $\{u,v\}$ are the witnesses $\{u_n, v_n\}$ for the non-$(n-1)$-balancedness of $\mathbf{w}_n$. Then the output contains the morphism $\sigma_{[n+1]}$, the sequence $\{i_{n+1}, j_{n+1}, k_{n+1}\}$ and the witnesses $\{u_{n+1}, v_{n+1}\}$ of the non-n-balancedness of $\mathbf{w_{n+1}}$.

```
> nextMorphism[s_, {i_, j_, k_}, {u_, v_}, n_] :=
  List[
    compose[compose[power[stMor[k], n], power[stMor[i], n]], s],
    List[k, i, j],
    List[compose[power[minus_s[k], n], power[plus_s[i], n]][v],
      compose[power[plus_s[k], n], power[minus_s[i], n]][u]]
  ]
```

tau

Input:
u, a finite word over $\{0,1,2\}$;

Output:
tau[u], the image by τ of u, where τ is the Tribonacci morphism.

```
> tau[u_] := Flatten[u /. {0 -> {0, 1}, 1 -> {0, 2}, 2 -> {0}}]
```

generate

Inputs:
s, a morphism;
t, a finite word over the same alphabet;
n, a positive integer;

Output:
generate[s,t,n] returns a prefix of length at least n of the word s(t);

Comment:
We are interested into a prefix of `s(t)` of length at least `n`, but not too long compared to `n`. For this reason if `m` $= \min_{a\in\mathcal{A}} |\sigma(a)|$, we compute the image of the prefix of length $\lceil \frac{n}{m} \rceil$.

```
> generate[s_, t_, n_] := Module[{L, m, word},
    L = {Length[s[{0}]], Length[s[{1}]], Length[s[{2}]]};
    m = Min[L];
    word = s[t[[1 ;; Ceiling[n/m]]]];
    Return[word]
  ]
```

`complexities`

Inputs:
`m`, an integer greater than 1;

Output:
None

Comment:
This function computes the prefixes of length at least 250000 of the words $\mathbf{w_2}, \ldots, \mathbf{w_m}$, and print on the screen, for every $\mathbf{w_i}$, the values of $p_{\mathbf{w_i}}(n)$ and $b^{(2)}_{\mathbf{w_i}}(n)$ for $n \in [30]_0$.

```
> complexities[m_] := Module[{si, ii, ji, ki, ui, vi, i, wi, n},
    si = s2;
    {ii, ji, ki} = {0, 1, 2};
    {ui, vi} = {{1, 0, 1}, {0, 2, 0}};
    i = 2;
    While[i <= m,
      wi = generate[si, t, 250000];
      Print[Table[p[wi, n], {n, 0, 30}]];
      Print[Table[b2[wi, n, 3], {n, 0, 30}]];
      {si, {ii, ji, ki}, {ui, vi}} =
        nextMorphism[si, {ii, ji, ki}, {ui, vi}, i];
      i = i + 1
    ]
  ]
```

```
> complexities(6)

[Out]
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
{1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,
51,53,55,57,59,61}
```

We thus note that for $i \in \{2, \ldots, 8\}$, the prefix of $\mathbf{w_i}$ we considered is long enough since all factors appear in it, and that we obtain the equality between the factor and the 2-binomial complexity. We can thus conjecture that

$$b_{\mathbf{w_i}}^{(k)} = p_{\mathbf{w_i}},$$

for all $i \geq 2$ and $k \geq 2$. Note that $\sigma_{[8]}$ is such that

$$\left(|\sigma_{[8]}(0)|, |\sigma_{[8]}(1)|, |\sigma_{[8]}(2)|\right) = (3016665, 2397137, 4360777),$$

hence with our algorithm we get $|\sigma_{[8]}(\mathbf{T}_0)| = 3016665$. The computer cannot manage the computation of $\sigma_{[9]}$.

Index

I

K

L

M

P

R

S

T

W

Bibliography

[1] A. Aberkane and J. D. Currie. A cyclic binary morphism avoiding abelian fourth powers. *Theoret. Comput. Sci.*, 410(1):44–52, 2009.

[2] A. Aberkane, J. D. Currie, and N. Rampersad. The number of ternary words avoiding abelian cubes grows exponentially. *J. Integer Seq.*, 7(2):Article 04.2.7, 2004.

[3] B. Adamczewski and Y. Bugeaud. A short proof of the transcendence of Thue-Morse continued fractions. *Amer. Math. Monthly*, 114(6):536–540, 2007.

[4] J.-P. Allouche and M. Mendès France. Euler, Pisot, Prouhet-Thue-Morse, Wallis and the duplication of sines. *Monatsh. Math.*, 155(3-4):301–315, 2008.

[5] J.-P. Allouche and J. Shallit. The ubiquitous Prouhet-Thue-Morse sequence. In *Sequences and their applications (Singapore, 1998)*, Springer Ser. Discrete Math. Theor. Comput. Sci., pages 1–16. Springer, London, 1999.

[6] J.-P. Allouche and J. Shallit. *Automatic sequences. Theory, applications, generalizations*. Cambridge University Press, Cambridge, 2003.

[7] P. Arnoux and G. Rauzy. Représentation géométrique de suites de complexité $2n + 1$. *Bull. Soc. Math. France*, 119(2):199–215, 1991.

[8] M. Aschbacher. *Finite group theory*, volume 10 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1986.

[9] J.-M. Autebert, J. Berstel, and L. Boasson. Context-free languages and pushdown automata. In *Handbook of formal languages*, volume 1, pages 111–174. Springer, Berlin, 1997.

[10] S. V. Avgustinovich, D. G. Fon-Der-Flaass, and A. E. Frid. Arithmetical complexity of infinite words. In *Words, Languages & Combinatorics, III (Kyoto, 2000)*, pages 51–62. World Sci. Publ., River Edge, NJ, 2003.

[11] J. Bell. On the values attained by a k-regular sequence. *Adv. in Appl. Math.*, 34(3):634–643, 2005.

[12] C. Berge. *Graphes et hypergraphes*. Monographies universitaires de mathématiques, 37. Dunod, Paris, 1970.

[13] J. Berstel, M. Crochemore, and J.-E. Pin. Thue-Morse sequence and p-adic topology for the free monoid. *Discrete Math.*, 76(2):89–94, 1989.

[14] V. Berthé, J. Karhumäki, D. Nowotka, and J. Shallit. Mini-workshop: Combinatorics on words. *Oberwolfach Rep.*, 7:2195–2244, 2010.

[15] V. Berthé and M. Rigo, editors. *Combinatorics, automata and number theory*, volume 135 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2010.

[16] F. Blanchet-Sadri, D. Seita, and D. Wise. Computing abelian complexity of binary uniform morphic words. *Theoret. Comput. Sci.*, 640:41–51, 2016.

[17] M. R. Bridson and R. H. Gilman. Context-free languages of sub-exponential growth. *J. Comput. System Sci.*, 64(2):308–310, 2002.

[18] S. Brlek. Enumeration of factors in the Thue-Morse word. *Discrete Appl. Math.*, 24(1-3):83–96, 1989.

[19] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire. Logic and p-recognizable sets of integers. In *Journées Montoises (Mons, 1992)*, volume 1, pages 191–238. Soc. Math. Belgique, 1994.

[20] J. R. Büchi. On a decision method in restricted second order arithmetic. In *Logic, Methodology and Philosophy of Science (Proc. 1960 Internat. Congr.)*, pages 1–11. Stanford Univ. Press, Stanford, Calif., 1962.

[21] J. Cassaigne, S. Ferenczi, and L. Q. Zamboni. Imbalances in Arnoux-Rauzy sequences. *Ann. Inst. Fourier (Grenoble)*, 50(4):1265–1276, 2000.

[22] J. Cassaigne and A. E. Frid. On the arithmetical complexity of Sturmian words. *Theoret. Comput. Sci.*, 380(3):304–316, 2007.

[23] J. Cassaigne, J. Karhumäki, S. Puzynina, and M. A. Whiteland. k-abelian equivalence and rationality. *Fund. Inform.*, 154(1-4):65–94, 2017.

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

[25] J. Chen, X. Lü, and W. Wu. On the k-abelian complexity of the Cantor sequence. *J. Combin. Theory Ser. A*, 155:287–303, 2018.

[26] J. Chen and Z.-X. Wen. On the abelian complexity of generalized Thue-Morse sequences. *Theoret. Comput. Sci.*, 780:66–73, 2019.

[27] Joshua Cooper and Aaron Dutle. Greedy galois games. *The American Mathematical Monthly*, 120, 2011.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[29] E. M. Coven and G. A. Hedlund. Sequences with minimal block growth. *Math. Systems Theory*, 7:138–153, 1973.

[30] J. D. Currie and N. Rampersad. Recurrent words with constant Abelian complexity. *Adv. in Appl. Math.*, 47(1):116–124, 2011.

[31] J. D. Currie and N. Rampersad. Fixed points avoiding Abelian k-powers. *J. Combin. Theory Ser. A*, 119(5):942–948, 2012.

[32] D. Damanik. Local symmetries in the period-doubling sequence. *Discrete Appl. Math.*, 100(1-2):115–121, 2000.

[33] A. de Luca and S. Varricchio. Some combinatorial properties of the Thue-Morse sequence and a problem in semigroups. *Theoret. Comput. Sci.*, 63(3):333–348, 1989.

[34] M. Dekking. Transcendance du nombre de Thue-Morse. *C. R. Acad. Sci. Paris Sér. A-B*, 285(4):A157–A160, 1977.

[35] A. W. M. Dress and P. L. Erdős. Reconstructing words from subwords in linear time. *Ann. Comb.*, 8(4):457–462, 2004.

[36] M. Dudík and L. J. Schulman. Reconstruction from subsequences. *J. Combin. Theory Ser. A*, 103(2):337–348, 2003.

[37] A. Ehrenfeucht, K. P. Lee, and G. Rozenberg. Subword complexities of various classes of deterministic developmental languages without interactions. *Theoret. Comput. Sci.*, 1(1):59–75, 1975.

[38] S. Eilenberg. *Automata, languages, and machines. Vol. A*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York, 1974. Pure and Applied Mathematics, Vol. 58.

[39] P. L. Erdős, P. Ligeti, P. Sziklai, and D. C. Torney. Subwords in reverse-complement order. *Ann. Comb.*, 10(4):415–430, 2006.

[40] M. Ferov. Irreducible polynomials modulo p. Bachelor's thesis. Charles University in Prague, 2008.

[41] P. Fleischmann, M. Lejeune, F. Manea, D. Nowotka, and M. Rigo. Reconstructing words from right-bounded-block words. In Nataša Jonoska and Dmytro Savchuk, editors, *Developments in Language Theory*, pages 96–109, Cham, 2020. Springer International Publishing.

[42] S. Fossé and G. Richomme. Some characterizations of Parikh matrix equivalent binary words. *Inform. Process. Lett.*, 92(2):77–82, 2004.

[43] W. Foster and I. Krasikov. An improvement of a Borwein-Erdélyi-Kós result. *Methods Appl. Anal.*, 7(4):605–614, 2000.

[44] D. Freydenberger, P. Gawrychowski, J. Karhumäki, F. Manea, and W. Rytter. *Multidisciplinary Creativity: Homage to Gheorghe Paun on his 65th Birthday*, chapter Testing k-binomial equivalence, pages 239–248. Spandugino, Bucharest, Romania, 2015.

[45] A. E. Frid. Sequences of linear arithmetical complexity. *Theoret. Comput. Sci.*, 339(1):68–87, 2005.

[46] A. E. Frid, E. Laborde, and J. Peltomäki. On prefix palindromic length of automatic words. https://arxiv.org/abs/2009.02934, 2020.

[47] G. Frobenius. Ueber Matrizen aus nicht negativen Elementen. *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, pages 456–477, 1912.

[48] P. Gawrychowski, M. Kosche, T. Koss, F. Manea, and S. Siemer. Efficiently testing Simon's congruence. In *38th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 187 of *Leibniz International Proceedings in Informatics*. LIPICS, 2021.

[49] A. Graham. *Kronecker products and matrix calculus: with applications*. Ellis Horwood Ltd., Chichester; Halsted Press [John Wiley & Sons, Inc.], New York, 1981. Ellis Horwood Series in Mathematics and its Applications.

[50] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics*. Addison-Wesley Publishing Company, Advanced Book Program, Reading, MA, 1989. A foundation for computer science.

[51] F. Greinecker. On the 2-abelian complexity of the Thue-Morse word. *Theoret. Comput. Sci.*, 593:88–105, 2015.

[52] F. Harary. On the reconstruction of a graph from a collection of subgraphs. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963)*, pages 47–52. Publ. House Czechoslovak Acad. Sci., Prague, 1964.

[53] O. H. Ibarra and B. Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *STACS 86 (Orsay, 1986)*, volume 210 of *Lecture Notes in Comput. Sci.*, pages 171–179. Springer, Berlin, 1986.

[54] R. Incitti. The growth function of context-free languages. *Theoret. Comput. Sci.*, 255(1-2):601–605, 2001.

[55] I. Kaboré and B. Kientéga. Abelian complexity of Thue-Morse word over a ternary alphabet. In *Combinatorics on words*, volume 10432 of *Lecture Notes in Comput. Sci.*, pages 132–143. Springer, Cham, 2017.

[56] L. O. Kalashnik. The reconstruction of a word from fragments. *Numerical Mathematics and Computer Technology*, pages 56–57, 1973.

[57] P. Karandikar, M. Kufleitner, and Ph. Schnoebelen. On the index of Simon's congruence for piecewise testability. *Inform. Process. Lett.*, 115(4):515–519, 2015.

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

[59] J. Karhumäki, A. Saarela, and L. Q. Zamboni. Variations of the Morse-Hedlund theorem for k-abelian equivalence. *Acta Cybernet.*, 23(1):175–189, 2017.

[60] P. J. Kelly. A congruence theorem for trees. *Pacific J. Math.*, 7:961–968, 1957.

[61] S. Knapowski. On the Möbius function. *Acta Arith.*, 4:209–216, 1958.

[62] I. Krasikov and Y. Roditty. On a reconstruction problem for sequences. *J. Combin. Theory Ser. A*, 77(2):344–348, 1997.

[63] J. Lambek and L. Moser. On some two way classifications of integers. *Canad. Math. Bull.*, 2:85–89, 1959.

[64] M. Latteux and G. Thierrin. On bounded context-free languages. *Elektron. Informationsverarb. Kybernet.*, 20(1):3–8, 1984.

[65] M. Lejeune. *Au sujet de la complexité k-binomiale*. Master's thesis, Université de Liège, 2018.

[66] M. Lejeune, J. Leroy, and M. Rigo. Computing the *k*-binomial complexity of the Thue-Morse word. In *Developments in language theory*, volume 11647 of *Lecture Notes in Comput. Sci.*, pages 278–291. Springer, Cham, 2019.

[67] M. Lejeune, J. Leroy, and M. Rigo. Computing the *k*-binomial complexity of the Thue-Morse word. *J. Combin. Theory Ser. A*, 176:105284, 44, 2020.

[68] M. Lejeune, M. Rigo, and M. Rosenfeld. Templates for the *k*-binomial complexity of the Tribonacci word. In *Combinatorics on words*, volume 11682 of *Lecture Notes in Comput. Sci.*, pages 238–250. Springer, Cham, 2019.

[69] M. Lejeune, M. Rigo, and M. Rosenfeld. The binomial equivalence classes of finite words. *Internat. J. Algebra Comput.*, 30(7):1375–1397, 2020.

[70] M. Lejeune, M. Rigo, and M. Rosenfeld. Mathematica notebook for computing the 2-binomial complexity of the Tribonacci word. http://hdl.handle.net/2268/234215, 2020.

[71] M. Lejeune, M. Rigo, and M. Rosenfeld. Templates for the *k*-binomial complexity of the Tribonacci word. *Adv. in Appl. Math.*, 112:101947, 26, 2020.

[72] J. Leroy, M. Rigo, and M. Stipulanti. Generalized Pascal triangle for binomial coefficients of words. *Adv. in Appl. Math.*, 80:24–47, 2016.

[73] V. I. Levenshteĭn. Perfect codes in the metric of deletions and insertions. *Diskret. Mat.*, 3(1):3–20, 1991.

[74] F. Liétard. *Evitabilité de puissances additives en combinatoires des mots*. PhD thesis, Université de Lorraine, 2020.

[75] M. Lothaire. *Combinatorics on words*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, corrected reprint of the 1983 original edition, 1997.

[76] M. Lothaire. *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2002.

[77] M. Lothaire. *Applied combinatorics on words*, volume 105 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2005.

[78] X. Lü, J. Chen, Z. Wen, and W. Wu. On the abelian complexity of the Rudin-Shapiro sequence. *J. Math. Anal. Appl.*, 451(2):822–838, 2017.

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

[80] B. Manvel, A. Meyerowitz, A. Schwenk, K. Smith, and P. Stockmeyer. Reconstruction of sequences. *Discrete Math.*, 94(3):209–219, 1991.

[81] B. Manvel and P. Stockmeyer. On reconstruction of matrices. *Math. Mag.*, 44:218–221, 1971.

[82] A. Mateescu and A. Salomaa. Matrix indicators for subword occurrences and ambiguity. *Internat. J. Found. Comput. Sci.*, 15(2):277–292, 2004.

[83] A. Mateescu, A. Salomaa, K. Salomaa, and S. Yu. A sharpening of the Parikh mapping. *Theor. Inform. Appl.*, 35(6):551–564 (2002), 2001. A tribute to Aldo de Luca.

[84] A. Mateescu, A. Salomaa, and S. Yu. Subword histories and Parikh matrices. *J. Comput. System Sci.*, 68(1):1–21, 2004.

[85] C. Moreau. Sur les permutations circulaires distinctes (french). *Nouv. Ann.*, 2(XI):309–314, 1872.

[86] M. Morse and G. A. Hedlund. Symbolic Dynamics. *Amer. J. Math.*, 60(4):815–866, 1938.

[87] M. Morse and G. A. Hedlund. Symbolic dynamics II. Sturmian trajectories. *Amer. J. Math.*, 62:1–42, 1940.

[88] B. Mossé. Puissances de mots et reconnaissabilité des points fixes d'une substitution. *Theoret. Comput. Sci.*, 99(2):327–334, 1992.

[89] B. Mossé. Reconnaissabilité des substitutions et complexité des suites automatiques. *Bull. Soc. Math. France*, 124(2):329–346, 1996.

[90] H. Mousavi. Automatic theorem proving in Walnut. arXiv:1603.06017.

[91] P. Ochsenschläger. Binomialkoeffizienten und Shuffle-Zahlen. *Fachbereicht Informatik*, 1981.

[92] P. V. O'Neil. Ulam's conjecture and graph reconstructions. *Amer. Math. Monthly*, 77:35–43, 1970.

[93] J.-J. Pansiot. Complexité des facteurs des mots infinis engendrés par morphismes itérés. In *Automata, languages and programming (Antwerp, 1984)*, volume 172 of *Lecture Notes in Comput. Sci.*, pages 380–389. Springer, Berlin, 1984.

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

[95] A. Parreau, M. Rigo, E. Rowland, and É. Vandomme. A new approach to the 2-regularity of the ℓ-abelian complexity of 2-automatic sequences. *Electron. J. Combin.*, 22(1):Paper 1.27, 44, 2015.

[96] O. Parshina. On arithmetic index in the generalized thue-morse word. In *Combinatorics on Words*, volume 10432 of *Lecture Notes in Computer Science*, pages 121–131. Springer International Publishing, Cham, 2017.

[97] D. Peifer. An introduction to combinatorial group theory and the word problem. *Math. Mag.*, 70(1):3–10, 1997.

[98] O. Perron. Zur Theorie der Matrices. *Math. Ann.*, 64(2):248–263, 1907.

[99] J.-É. Pin and P. V. Silva. A noncommutative extension of Mahler's theorem on interpolation series. *European J. Combin.*, 36:564–578, 2014.

[100] G. Pólya. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Math.*, 68(1):145–254, 1937.

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

[102] M. Rao, M. Rigo, and P. Salimov. Avoiding 2-binomial squares and cubes. *Theoret. Comput. Sci.*, 572:83–91, 2015.

[103] M. Rao and M. Rosenfeld. Avoiding two consecutive blocks of same size and same sum over $\mathbb{Z}^2$. *SIAM J. Discrete Math.*, 32(4):2381–2397, 2018.

[104] D. Raz. Length considerations in context-free languages. *Theoret. Comput. Sci.*, 183(1):21–32, 1997.

[105] C. Reutenauer. *Free Lie algebras*, volume 7 of *London Mathematical Society Monographs. New Series*. The Clarendon Press, Oxford University Press, New York, 1993. Oxford Science Publications.

[106] G. Richomme, K. Saari, and L. Q. Zamboni. Balance and abelian complexity of the Tribonacci word. *Adv. in Appl. Math.*, 45(2):212–231, 2010.

[107] G. Richomme, K. Saari, and L. Q. Zamboni. Abelian complexity of minimal subshifts. *J. Lond. Math. Soc. (2)*, 83(1):79–95, 2011.

[108] M. Rigo. *Formal languages, automata and numeration systems. 1*. ISTE, London; John Wiley & Sons, Inc., Hoboken, NJ, 2014. Introduction to combinatorics on words.

[109] M. Rigo and P. Salimov. Another generalization of abelian equivalence: binomial complexity of infinite words. *Theoret. Comput. Sci.*, 601:47–57, 2015.

[110] J. Sakarovitch. *Elements of automata theory*. Cambridge University Press, Cambridge, 2009. Translated from the 2003 French original by Reuben Thomas.

[111] A. Salomaa. Counting (scattered) subwords. *The Bulletin of the EATCS*, 81:165–179, 2003.

[112] A. Salomaa. Criteria for the matrix equivalence of words. *Theoret. Comput. Sci.*, 411(16-18):1818–1827, 2010.

[113] L. Schaeffer and J. Shallit. Closed, palindromic, rich, privileged, trapezoidal, and balanced words in automatic sequences. *Electron. J. Combin.*, 23(1):Article 1.25.19, 2016.

[114] J. Shallit and Y. Breitbart. Automaticity. I. Properties of a measure of descriptional complexity. *J. Comput. System Sci.*, 53(1):10–25, 1996.

[115] A. M. Shur. The structure of the set of cube-free words over a two-letter alphabet. *Izv. Math.*, 64:847–871, 2000.

[116] A. M. Shur. Combinatorial complexity of rational languages (Russian). *Diskretn. Anal. Issled. Oper. Ser. 1*, 12:78–99, 2005.

[117] A. M. Shur. Deciding context equivalence of binary overlap-free words in linear time. *Semigroup Forum*, 84:447–471, 2012.

[118] I. Simon. Piecewise testable events. In *Automata theory and formal languages (Second GI Conf., Kaiserslautern, 1975)*, volume 33 of *Lecture Notes in Comput. Sci.*, pages 214–222. Springer, 1975.

[119] A. Szilard, S. Yu, K. Zhang, and J. Shallit. Characterizing regular languages with polynomial densities. In *Mathematical foundations of computer science 1992 (Prague, 1992)*, volume 629 of *Lecture Notes in Comput. Sci.*, pages 494–503. Springer, Berlin, 1992.

[120] A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Kra. Vidensk. Selsk. Skrifter. I. Mat.-Nat. Kl.*, 10, 1912.

[121] V. I. Trofimov. Growth functions of some classes of languages. *Kibernetika (Kiev)*, 6:i, 9–12, 149, 1981.

[122] O. Turek. Abelian complexity function of the Tribonacci word. *J. Integer Seq.*, 18(3):Article 15.3.4, 29, 2015.

[123] L. Van Iersel and V. Moulton. Left-reconstructibility of phylogenetic networks. *SIAM J. Discret. Math.*, 32(3):2047–2066, 2018.

[124] O. Vernet and L. Markenzon. Solving problems for maximal reducible flow-graphs. *Discrete Applied Mathematics*, 136(2):341 – 348, 2004. The 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization.

[125] M. A. Whiteland. *On the k-abelian equivalence relation of finite words*. PhD thesis, University of Turku, 2019.