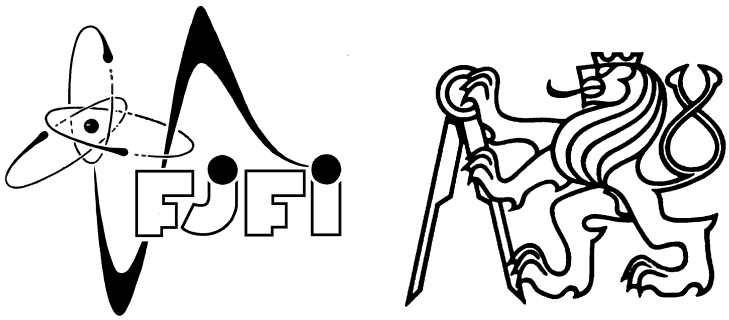
České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

katedra matematiky

obor: Inženýrská informatika

zaměření: Softwarové inženýrství a matematická informatika



Abelovská komplexita nekonečných slov

Abelian complexity of infinite words

BAKALÁŘSKÁ PRÁCE

autor práce: **Karel Břinda**

vedoucí práce: **prof. Ing. Edita Pelantová, CSc.**

konzultant: **Ing. Ondřej Turek, Ph.D.**

akademický rok: **2010/11**

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze dne 15. července 2011 Karel Břinda

Acknowledgments

I thank all members of TIGR (Theoretical Informatics Group) at FNSPE for bringing me into the amazing world of combinatorics on words. The person I want to express my gratitude – for all the time dedicated to me, ideas, inspiration and patience – is my supervisor, Edita Pelantová. Then I thank my adviser, Ondřej Turek, for ideas concerning my topic. I am also grateful to Ľubomíra Balková for answers to all my questions and language help.

Název práce: **Abelovská komplexita nekonečných slov**
Autor: Karel Břinda

Obor: Inženýrská informatika
Zaměření: Softwarové inženýrství a matematická informatika
Druh práce: bakalářská práce

Vedoucí práce: prof. Ing. Edita Pelantová, CSc.
katedra matematiky
Fakulta jaderná a fyzikálně inženýrská
České vysoké učení technické v Praze

Konzultant: Ing. Ondřej Turek, Ph.D.
Laboratory of Physics
Kochi University of Technology

Abstrakt:

Tato práce se zabývá abelovskou komplexitou nekonečných slov, což je funkce popisující složitost nekonečného slova pomocí množin Parikhových vektorů konečných faktorů daného slova.

Jsou shrnuty některé již známé výsledky pro nekonečná slova generovaná primitivními substitucemi (včetně vztahů mezi abelovskou komplexitou, balanční funkcí, diskrepanční funkcí a maticí primitivní substituce). Pro třídu slov příslušejících kvadratickým Parryho číslům je odvozen explicitní vzorec k jejímu výpočtu. Dále je popsána sada nástrojů TigrWords, jež umožňuje zkoumání nekonečných slov aproximovaných prefixy a která může sloužit k vytváření nových hypotéz.

Klíčová slova: kombinatorika na slovech, abelovská komplexita, balancovanost, diskrepance

Title: **Abelian complexity of infinite words**
Author: Karel Břinda

Abstract:

This thesis deals with Abelian complexity of infinite words, i.e., function describing complexity of a given infinite word using sets of Parikh vectors of finite factors of the word.

Some of already known results for words generated by primitive substitutions (including connection between Abelian complexity, balance function, discrepancy function and matrix of a primitive substitution) are summarized. Then an explicit formula for Abelian complexity of infinite words associated with quadratic Parry numbers is derived. In addition, a new toolkit TigrWords, which facilitates study of infinite words approximated by their prefixes and may serve to discover of new hypotheses, is described.

Keywords: combinatorics on words, Abelian complexity, balance property, discrepancy

Contents

List of symbols

Symbol	Description
$\mathbb{N}^+$	set of positive integer numbers, $\{1, 2, \ldots\}$
$\mathbb{N}$	set of non-negative integer numbers, $\{0, 1, 2, \ldots\}$
$\mathbb{Z}$	set of integer numbers, $\{\ldots - 2, -1, 0, 1, 2, \ldots\}$
$\mathbb{Q}$	set of rational numbers
$\mathbb{C}$	set of complex numbers
$\#M$	cardinality of the set M
$\binom{n}{k}$	binomial coefficient, $\frac{n!}{k!(n-k)!}$
$\mathcal{A}$	alphabet
$\mathcal{A}^\star$	set of all finite words over $\mathcal{A}$
$\mathcal{A}^{\mathbb{N}}$	set of all one-sided infinite words over $\mathcal{A}$
ε	empty word in $\mathcal{A}^\star$
$\mathbf{u}, \mathbf{v}$	infinite words over $\mathcal{A}$
u, v	finite words over $\mathcal{A}$
$\lvert u\rvert$	length of u
$\lvert u\rvert_a$	number of occurrences of letter a in u
$u_{[n,n+k)}$	factor $u_n u_{n+1} \cdots u_{n+k-1}$ of u
$\mathbf{u}_{[n,n+k)}$	factor $\mathbf{u}_n \mathbf{u}_{n+1} \cdots \mathbf{u}_{n+k-1}$ of $\mathbf{u}$
$\bar{u}$	mirror image $u_{n-1} u_{n-2} \cdots u_0$ of $u = u_0 \cdots u_{n-2} u_{n-1}$
$\mathcal{L}(\mathbf{u})$	set of factors of $\mathbf{u}$
$\mathcal{L}_n(\mathbf{u})$	set of factors of $\mathbf{u}$ of length n
Δ	end of theorem, proposition, lemma, observation, corollary, definition, and example
$\square$	end of proof

CHAPTER 0

Introduction

The topic of this thesis makes a part of combinatorics on words. Information about history of this mathematical branch can be found, e.g., in [3]. Our basic term is Abelian complexity of an infinite word, i.e., function describing complexity of a given infinite word using sets of Parikh vectors of finite factors of the word.

Abelian complexity of infinite words has been examined for the first time by Ethan M. Coven and Gustav A. Hedlund in [8], where they have revealed that it could serve as an alternative way of characterization of periodic words and Sturmian words. However, the notion "Abelian complexity" itself comes from the paper [19] that in a sense initiated a general study of the Abelian complexity of infinite words over finite alphabets. It is noteworthy that besides Sturmian words, Abelian complexity is still known for only few infinite words.

The following text is divided into five chapters:

1) **Preliminaries**

   Basic terms concerning combinatorics on words – especially Abelian complexity of infinite words – and numeration systems are defined and the most important general theorems are stated.

2) **Relations between Abelian complexity, balance properties, discrepancy, and incidence matrices**

   Relations between Abelian complexity and other characteristics of fixed points of primitive substitutions are summarized, in particular, relations between

   - Abelian complexity and balance properties,
   - balance properties and discrepancy,
   - discrepancy and matrices of primitive substitutions.

3) **Balance properties and Abelian complexity of words over binary alphabets**

   This chapter resumes known results about Abelian complexity in the case of binary alphabets. Then an explicit formula for Abelian complexity of words associated with simple and non-simple quadratic Parry numbers is derived – this result has been already submitted as [4].

4) **Balance properties and Abelian complexity of words over multiliteral alphabets**

   The question of Abelian complexity of the d-bonacci word is introduced and basic theorems about the Tribonacci word and words associated with cubic non-simple Parry numbers are stated.

5) **TigrWords**

   The toolkit TigrWords – to be found on the enclosed CD – is described.

CHAPTER 1

Preliminaries

In this chapter, we recall basic definitions and statements concerning combinatorics on words and numeration systems.

1.1 Combinatorics on words

1.1.1 Basic terms

Alphabet is a finite non-empty set $\mathcal{A} = \{a_0, \ldots, a_{d-1}\}$. Its elements are called **letters**. In this text, we will usually use the alphabet $\mathcal{A} = \{0, \ldots, d-1\}$. **Finite word** u is a finite sequence over $\mathcal{A}$. The number of letters in the word u is called **length** of u and we denote it by $|u|$. A special case is a unique word of zero length called **empty word** and denoted by ε. Let us denote the number of occurrences of a letter $a \in \mathcal{A}$ in the word u by $|u|_a$.

$\mathcal{A}^*$ is the set of all finite words over $\mathcal{A}$ and $\mathcal{A}^+$ is the set of all finite words over $\mathcal{A}$ except ε. $\mathcal{A}^n$ is the subset of $\mathcal{A}^*$ containing all words of length n. $\mathcal{A}^{\mathbb{N}}$ is the set of all **infinite words**, i.e., one-sided infinite sequences over $\mathcal{A}$.

We can equip the sets $\mathcal{A}^*$, $\mathcal{A}^+$ with the binary operation called **concatenation**

$$(v_1 v_2 \cdots v_m) \cdot (w_1 w_2 \cdots w_n) = v_1 v_2 \cdots v_m w_1 w_2 \cdots w_n$$

in order to get a monoid[1] $(\mathcal{A}^*, \cdot)$ with the neutral element ε and a semigroup[2] $(\mathcal{A}^+, \cdot)$ respectively.

The word u is **factor** of a word w if $w = vuv'$ for some $v \in \mathcal{A}^*, v' \in \mathcal{A}^* \cup \mathcal{A}^{\mathbb{N}^+}$. In the case $v = \varepsilon$, we say that u is **prefix** of w. Similarly, if $v' = \varepsilon$, then v is **suffix** of w. To make the formulae in this text shorter, we denote $u_n u_{n+1} \cdots u_{n+k-1}$ by $u_{[n,n+k)}$.

The set of all factors of an infinite word $\mathbf{u}$ is called **language** and we denote it by $\mathcal{L}(\mathbf{u})$. Its subset containing just all factors of $\mathbf{u}$ of length n is denoted by $\mathcal{L}_n(\mathbf{u})$.

Let $\mathbf{u}$ be an infinite word and let w be its non-empty factor. The set of all letters $a \in \mathcal{A}$ such that $wa \in \mathcal{L}(\mathbf{u})$ is called **right extension** of w in $\mathbf{u}$. It is denoted by $Rext(w)$. If $\#Rext(w) > 1$, the word w is said to be **right special factor**. Similarly, we define **left extension** and **left special factor**. A factor which is left special and right special is called **bispecial factor**.

[1] Monoid is a structure with an associative binary operation and a neutral element.

[2] Semigroup is a structure with an associative binary operation.

If every factor of an infinite word $\mathbf{u}$ occurs at least twice (which implies that it occurs infinitely many times), $\mathbf{u}$ is said to be **recurrent**. A word v is **return word** of $w \in \mathcal{L}(\mathbf{u})$ if the following conditions are satisfied:

- $vw \in \mathcal{L}(\mathbf{u})$;
- w is a prefix of vw;
- w occurs in vw exactly twice.

The factor vw is said to be **complete return word** of w. Let us denote the set of all return words of w by $Ret(w)$.

A recurrent word $\mathbf{u}$ satisfying that for all w in $\mathcal{L}(\mathbf{u})$ it holds $\#Ret(w) < +\infty$ is called **uniformly recurrent**. If there exists $C > 0$ such that for all w in $\mathcal{L}(\mathbf{u})$ and all v in $Ret(w)$, it is satisfied $|v| < C|w|$, we call the word $\mathbf{u}$ **linearly recurrent**.

Let $u = u_0u_1 \cdots u_{n-1}$ be a finite word. We define its **mirror image** as $\bar{u} = u_{n-1} \cdots u_1u_0$. Words satisfying $u = \bar{u}$ are called **palindromes**.

An infinite word $\mathbf{u}$ is **eventually periodic** if $\mathbf{u} = vw^\omega$ for some $v, w \in \mathcal{A}^*$, where w^ω means repetition of w infinitely many times. In the special case $v = \varepsilon$, $\mathbf{u}$ is said to be **periodic**. Words which are not eventually periodic are **aperiodic**.

An infinite word $\mathbf{u}$ has **vector of frequencies** $\mu = (\mu_{a_0}, \ldots, \mu_{a_{d-1}})$ if there exist limits

$$\mu_a = \lim_{N \to +\infty} \frac{|\mathbf{u}_{[0,N)}|_a}{N}$$

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

The set $\mathcal{A}^{\mathbb{N}^+}$ can be equipped with a metric defined as

$$\rho(\mathbf{u}, \mathbf{v}) = \begin{cases} 2^{-\min\{m \in \mathbb{N} \mid \mathbf{u}_m \neq \mathbf{v}_m\}} & \text{for } \mathbf{u} \neq \mathbf{v}; \\ 0 & \text{for } \mathbf{u} = \mathbf{v}. \end{cases} \tag{1.1}$$

The metric ρ induces **product topology**. The set $\mathcal{A}^{\mathbb{N}^+}$ is a compact topological space.

1.1.2 Substitution

Among several ways of how to create an infinite word, one of the most common ones is to generate it by a special kind of morphisms on $\mathcal{A}^*$ – using the so-called substitution.

Definition 1.1. **Substitution** is a morphism[3] $\varphi : \mathcal{A}^* \to \mathcal{A}^*$ satisfying:

- there exists a letter $a \in \mathcal{A}$ such that a is a prefix of $\varphi(a)$;
- for all letters $b \in \mathcal{A}$, it holds $\lim_{n \to +\infty} |\varphi^n(b)| = +\infty$.

The action of the morphism φ can be naturally extended to an infinite word $\mathbf{u}$ by $\varphi(\mathbf{u}_0\mathbf{u}_1\mathbf{u}_2 \cdots) = \varphi(\mathbf{u}_0)\varphi(\mathbf{u}_1)\varphi(\mathbf{u}_2) \cdots$. An infinite word $\mathbf{u}$ is a fixed point of morphism φ if $\mathbf{u} = \varphi(\mathbf{u})$. Δ

We can mention a property which directly follows from the definition.

[3] Morphism is a mapping $\varphi : \mathcal{A}^* \to B^*$ satisfying $\varphi(vw) = \varphi(v)\varphi(w)$ for all v, w in $\mathcal{A}^*$. Every morphism is fully defined by images of all letters of the source alphabet.

Lemma 1.2. Every substitution has a **fixed point**

$$\mathbf{u} = \lim_{n\to+\infty} \varphi^n(a),$$

where a is from Definition 1.1. Δ

The class of all substitutions is very wide and and hard to treat in full generality. Therefore, we consider in this thesis an extensively studied subclass of primitive substitutions.

Definition 1.3. A substitution φ is **primitive** if there exists $k \in \mathbb{N}^+$ such that for all $a, b \in \mathcal{A}$, it is satisfied that $\varphi^k(a)$ contains b. Δ

Proposition 1.4. Fixed points of a primitive substitution are linearly recurrent. Δ

Now, we can introduce a matrix which contains information about images of individual letters of the alphabet. As we will see in the following chapters, many properties of fixed points of substitutions are readable from this matrix.

Definition 1.5. Let φ be a morphism over the alphabet $\mathcal{A} = \{0, 1, \ldots, d-1\}$. The matrix

$$\mathcal{M}_\varphi = \begin{pmatrix} |\varphi(0)|_0 & \cdots & |\varphi(d-1)|_0 \\ \vdots & \ddots & \vdots \\ |\varphi(0)|_{d-1} & \cdots & |\varphi(d-1)|_{d-1} \end{pmatrix}$$

is said to be **incidence matrix**[4] of φ. Δ

It follows immediately from the definition of M_φ that for any word u over $\mathcal{A} = \{0, 1, \ldots, d-1\}$

$$\begin{pmatrix} |\varphi(u)|_0 \\ |\varphi(u)|_1 \\ \vdots \\ |\varphi(u)|_{d-1} \end{pmatrix} = M_\varphi \begin{pmatrix} |u|_0 \\ |u|_1 \\ \vdots \\ |u|_{d-1} \end{pmatrix}. \tag{1.2}$$

It is important to take into consideration that an incidence matrix does not determine uniquely the substitution.

Observation 1.6. The incidence matrix of a composed morphism satisfies

$$\mathcal{M}_{\varphi\circ\theta} = \mathcal{M}_\varphi \mathcal{M}_\theta.$$

Δ

Proof. Take arbitrary $i, j \in \{1, 2, \ldots, d\}$, then

$$[M_\varphi M_\theta]_{ij} = \sum_{k=1}^{d} [M_\varphi]_{ik}[M_\theta]_{kj} = \sum_{k=1}^{d} |\varphi(k-1)|_{i-1}|\theta(j-1)|_{k-1} = |\varphi \circ \theta(j-1)|_{i-1} = [M_{\varphi\circ\theta}]_{ij}.$$

□

[4] Some authors use a different definition of incidence matrix which is, compared to ours, transposed.

Let us mention an important theorem. Its first statement is the famous Perron[5]-Frobenius[6] theorem (to be found, e.g., in [12]). The second statement has been shown by Martine Queffélec in [18].

Theorem 1.7 (Perron-Frobenius theorem and its corollary)**.** Let M_φ be the incidence matrix of a primitive substitution φ. Then:

i) The matrix M_φ has a positive eigenvalue Λ such that:
   - Λ is strictly greater than the modulus of any other eigenvalue and greater than one;
   - Λ is algebraically simple, i.e., it is a single root of the characteristic polynomial;
   - to Λ corresponds an eigenvector with positive entries, while no other eigenvalue has an eigenvector with positive entries.

ii) The entries of the eigenvector μ corresponding to Λ normalized so that $\sum_{i=0}^{d-1} \mu_i = 1$ are equal to the letter frequencies of the fixed point of φ.

Δ

Example 1.8 (Fibonacci[7] substitution and Fibonacci word)**.** *One of the simplest examples is the famous Fibonacci substitution*

$$\varphi : \begin{array}{l} 0 \to 01; \\ 1 \to 0. \end{array} \tag{1.3}$$

Let us generate the Fibonacci word $\mathbf{f}$*:*

$$\begin{aligned} \varphi^0(0) &= 0 \\ \varphi^1(0) &= 01 \\ \varphi^2(0) &= 010 \\ \varphi^3(0) &= 01001 \\ \varphi^4(0) &= 01001010 \\ &\vdots \\ \mathbf{f} = \lim_{n\to+\infty} \varphi^n(0) &= 01001010010010100101001001010010010100101001010\cdots \end{aligned}$$

Since the substitution φ is primitive, we can use Theorem 1.7 to find the vector of frequencies from its incidence matrix

$$\mathcal{M}_\varphi = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

After finding the greatest eigenvalue $\Lambda = \frac{1}{2}\left(1+\sqrt{5}\right)$ of the matrix M_φ, we obtain the vector of letter frequencies $\mu_\mathbf{f}$ of the Fibonacci word $\mathbf{f}$:

$$\mu_\mathbf{f} = \left(\frac{1}{2}\left(-1+\sqrt{5}\right), \frac{1}{2}\left(3-\sqrt{5}\right)\right).$$

Δ

[5] Oskar Perron (1880–1975) – German mathematician, known for his contributions concerning differential equations.

[6] Ferdinand Georg Frobenius (1849—1917) – German mathematician, best-known for his contributions to theory of differential equations and to group theory.

[7] Leonardo Pisano Bigollo (also known as Leonardo Fibonacci) (12^{th}–13^{th} ct.) – Italian mathematician, he popularized Hindu-Arabic numerals in Europe.

In Table 1.1, and Table 1.2, we present some frequently used substitutions.

Table 1.1: Basic substitutions over binary alphabet

Name	**Substitution** φ
Fibonacci	$0 \to 01$ $1 \to 0$
Period-doubling	$0 \to 01$ $1 \to 00$
Quadratic non-simple Parry[8]	$0 \to 0^p1$ $1 \to 0^q1$ $\quad p > q \geq 1$
Quadratic simple Parry	$0 \to 0^p1$ $1 \to 0^q$ $\quad p \geq q$
Thue[9]-Morse[10]	$0 \to 01$ $1 \to 10$

1.1.3 Complexity functions

For a given infinite word, our aim is to describe its complexity. For this purpose, there are defined various complexity functions which measure different kinds of complexities, for instance factor complexity, Abelian complexity, arithmetical complexity, permutation complexity, palindromic complexity, θ-palindromic complexity, ...

In this thesis, we deal with Abelian complexity. Nevertheless, we recall factor complexity at first.

Definition 1.9. Let $\mathbf{u}$ be an infinite word. The map $\mathcal{C}_{\mathbf{u}} : \mathbb{N}^+ \to \mathbb{N}^+$ defined as

$$\mathcal{C}_{\mathbf{u}}(n) = \#\mathcal{L}_n(\mathbf{u})$$

is called **factor complexity** (or subword complexity). Δ

Example 1.10. *Let us consider the Fibonacci word* $\mathbf{f}$ *from Example 1.8. It is readily seen that*

$$\begin{aligned} \mathcal{L}_1(\mathbf{f}) &= \{0, 1\} \\ \mathcal{L}_2(\mathbf{f}) &= \{00, 01, 10\} \\ \mathcal{L}_3(\mathbf{f}) &= \{001, 010, 100, 101\} \\ \mathcal{L}_4(\mathbf{f}) &= \{1001, 0100, 0010, 1010, 0101\} \\ &\vdots \end{aligned}$$

indeed, it may be proven that the factor complexity of the Fibonacci word is $\mathcal{C}_{\mathbf{f}}(n) = n + 1$. Δ

[8] William Parry (1934 – 2006) – English mathematician, best-known for his work on dynamical systems.

[9] Axel Thue (1863–1922) – Norwegian mathematician, known for work on diophantine approximation and combinatorics.

[10] Marston Morse (1892–1977) – American mathematician, known for his work on calculus of variations.

[11] All letters of images are $\mod d$.

[12] Walter Rudin (1921–2010) – American mathematician, known for his books on mathematical analysis.

[13] Harold Seymour Shapiro (1928) – American mathematician, known for his research on approximation theory, complex analysis, functional analysis, and partial differential equations.

Table 1.2: Basic substitutions over multiliteral alphabets

Name	**Substitution** φ
Generalized Thue-Morse	$0 \to \overbrace{01\cdots(b-1)}^{b \text{ letters}} \mod d$ [11] $1 \to 12\cdots b \mod d$ $\vdots$ $(d-1) \to (d-1)\cdots(b+d-2) \mod d$ where $d > 1, b \geq 2$
d-bonacci	$0 \to 01$ $1 \to 02$ $\vdots$ $(d-2) \to 0(d-1)$ $(d-1) \to 0$ where $d > 1$
Non-simple Parry	$0 \to 0^{t_1}1$ $\vdots$ $m-1 \to 0^{t_m}m$ $m \to 0^{t_{m+1}}(m+1)$ $\vdots$ $m+p-2 \to 0^{t_{m+p-1}}(m+p-1)$ $m+p-1 \to 0^{t_{m+p}}m$ where every proper suffix of $t_1t_2\cdots t_m(t_{m+1}t_{m+2}\cdots t_{m+p})^\omega$ is lexicographically smaller than the sequence itself
Rudin[12]-Shapiro[13]	$0 \to 01$ $1 \to 02$ $2 \to 31$ $3 \to 32$
Simple Parry	$0 \to 0^{t_1}1$ $\vdots$ $d-2 \to 0^{t_{d-1}}(d-1)$ $d-1 \to 0^{t_d}$ where $t_it_{i+1}\cdots t_d0^\omega$ is lexicographically smaller then $t_1t_2\cdots t_d0^\omega$ for all $i > 1$

Basic properties of factor complexity are summarized in the following lemma.

Lemma 1.11. The factor complexity of an infinite word $\mathbf{u}$ is an increasing function satisfying for all $n \in \mathbb{N}^+$

- $1 \leq \mathcal{C}_{\mathbf{u}}(n) \leq (\#\mathcal{A})^n$;
- $\Delta\mathcal{C}_{\mathbf{u}}(n) = \mathcal{C}_{\mathbf{u}}(n+1) - \mathcal{C}_{\mathbf{u}}(n) = \sum_{w \in \mathcal{L}_n(\mathbf{u})} (\#Rext(w) - 1)$.

Δ

Now, we can classify words with respect to their factor complexity. The most famous class of aperiodic words is the class of Sturmian[14] words, which are defined as aperiodic words with the minimal factor complexity $\mathcal{C}(n) = n + 1$. It is easily seen that their alphabet is binary. More information about the Sturmian words is in [5].

Abelian complexity is usually defined in similar way as factor complexity. Instead of different factors of a given length, we count appropriate Parikh vectors.

Definition 1.12. Let $\mathbf{u}$ be an infinite word over the alphabet $\mathcal{A} = \{0, 1, \ldots, d-1\}$. Then **Abelian complexity** is the map $\mathcal{AC}_{\mathbf{u}} : \mathbb{N}^+ \to \mathbb{N}^+$ defined as

$$\mathcal{AC}_{\mathbf{u}}(n) = \#\{\Psi(w) \mid w \in \mathcal{L}_n(\mathbf{u})\},$$

where $\Psi(w) = (|w|_0, \ldots, |w|_{d-1})$ is **Parikh[15] vector** of a word w. Δ

Observation 1.13. The incidence matrix M_φ from Definition 1.5 can be rewritten as

$$\mathcal{M}_\varphi = \begin{pmatrix} \left(\Psi\left(\varphi(0)\right)\right)^\intercal & \ldots & \left(\Psi\left(\varphi(d-1)\right)\right)^\intercal \end{pmatrix}.$$

Δ

Example 1.14. *Let us illustrate the Abelian complexity on the Fibonacci word* $\mathbf{f}$ *from Example 1.8. Let us apply the mapping* Ψ *on its factors (distinct Parikh vectors are in bold).*

$$\begin{array}{lrrrr} \mathcal{L}_1(\mathbf{f}): & \boldsymbol{\Psi(0) = (1,0)}, & \boldsymbol{\Psi(1) = (0,1)}; & & \\ \mathcal{L}_2(\mathbf{f}): & \boldsymbol{\Psi(00) = (2,0)}, & \boldsymbol{\Psi(01) = (1,1)}, & \Psi(10) = (1,1); & \\ \mathcal{L}_3(\mathbf{f}): & \boldsymbol{\Psi(001) = (2,1)}, & \Psi(010) = (2,1), & \Psi(100) = (2,1), & \boldsymbol{\Psi(101) = (1,2)}; \\ \mathcal{L}_4(\mathbf{f}): & \boldsymbol{\Psi(1001) = (2,2)}, & \boldsymbol{\Psi(0100) = (3,1)}, & \Psi(0010) = (3,1), & \Psi(1010) = (2,2) \\ & \Psi(0101) = (2,2) & & & \\ & \vdots & & & \end{array}$$

Indeed, it can be proven that the Abelian complexity of the Fibonacci word is constantly equal to 2. Δ

Proposition 1.15. Let $\mathbf{u}$ be an infinite word over the alphabet $\mathcal{A} = \{0, 1, \ldots, d-1\}$. Then for all $n \in \mathbb{N}^+$, it is satisfied

$$1 \leq \mathcal{AC}_{\mathbf{u}}(n) \leq \binom{n+d-1}{d-1}.$$

Δ

[14] Jacques-Charles-François Sturm (1803 - 1855) – French-Swiss mathematician.
[15] Rohit Jivanlal Parikh (1936) – Indian mathematician, logician, and philosopher.

As a consequence of Proposition 1.15, we get

$$\mathcal{AC}_{\mathbf{u}}(n) \in \mathcal{O}(n^{d-1}).^{16}.$$

Proposition 1.16. An infinite word $\mathbf{u}$ is periodic of period p if and only if $\mathcal{AC}_{\mathbf{u}}(p) = 1$. Δ

Proof. A word $\mathbf{u}$ is periodic of period $p \Leftrightarrow \mathbf{u}_i = \mathbf{u}_{i+p}$ for all $i \in \mathbb{N} \Leftrightarrow \Psi(\mathbf{u}_{[i,i+p)}) = \Psi(\mathbf{u}_{[i+1,i+p+1)})$ for all $i \in \mathbb{N} \Leftrightarrow \mathcal{AC}_{\mathbf{u}}(p) = 1$. □

1.1.4 Balance properties and discrepancy

Definition 1.17. Let $\mathbf{u}$ be an infinite word over the alphabet $\mathcal{A}$. Then **balance function** is the map $B_{\mathbf{u}} : \mathbb{N}^+ \to \mathbb{N}$ defined as

$$B_{\mathbf{u}}(n) = \max_{a\in\mathcal{A}} \max_{v,w\in\mathcal{L}_n(\mathbf{u})} (|v|_a - |w|_a).$$

If $B_{\mathbf{u}}(n)$ is bounded by some C, then $\mathbf{u}$ is said to be **C-balanced**. In the case $C = 1$, we say that $\mathbf{u}$ is **balanced**. Δ

Definition 1.18. Let $\mathbf{u}$ be a fixed point of a primitive substitution. We define **discrepancy function** as

$$D_{\mathbf{u}}(n) = \max_{a\in\mathcal{A}} \big||\mathbf{u}_{[0,n)}|_a - n\mu_a\big|,$$

where μ_a denotes frequency of the letter a. Δ

1.2 Numeration systems

In the end of this chapter, we recall some basic notations concerning numeration systems, which will play an important role in Chapter 3. More detailed information about this topic can be found in [16].

Definition 1.19. Let $(U_n)_{n=0}^{+\infty}$ be a strictly increasing sequence of integers, where $U_0 = 1$, and let N be a non-negative integer. Then we say that a finite sequence of integers $(d_i)_{i=0}^{l}$ satisfying $N = \sum_{i=0}^{l} d_i U_i$ is **U-representation** of N and we write it as $(d_l, \cdots, d_0)$. Δ

Among many U-representations, the most important is the following one.

Definition 1.20. Let $(U_n)_{n=0}^{+\infty}$ be a strictly increasing sequence of integers, where $U_0 = 1$, and let N be a non-negative integer. A representation obtained by Algorithm 1 is said to be **normal U-representation** (sometimes called greedy representation).
We denote it as $\langle N\rangle_U = (d_{i_{max}}, \ldots, d_1, d_0)$. Δ

A normal U-representation is not determined uniquely. It can begin with an arbitrary number of zeros.

Let us mention a proposition which describes the relation between U-representations and normal U-representation.

Proposition 1.21. The normal U-representation of an integer is the greatest with respect to the radix order among all U-representations of that integer. Δ

[16] There exists $C > 0$ such that for all $n \in \mathbb{N}^+ : \mathcal{AC}_{\mathbf{u}}(n) < Cn^{d-1}$

Algorithm 1 Normal U-representation

Input: $N \geq 0$
  find $i_{max} \in \mathbb{N}$ such that $N < U_{i_{max}+1}$
  $i \leftarrow i_{max}$, $n \leftarrow N$
  while $i \geq 0$ **do**
    $d_i \leftarrow \lfloor \frac{n}{U_i} \rfloor$
    $n \leftarrow (n - d_i U_i)$
    $i \leftarrow (i - 1)$
  end while
Output: $(d_j)_{j=0}^{i_{max}}$

Example 1.22 (Fibonacci system)**.** *Let U be the Fibonacci sequence defined by*

$$U_0 = 1, U_1 = 2, U_n = U_{n-1} + U_{n-2}.$$

Let us find more representations of number 42*.*

i) $42 = 3 \cdot 13 + 3 \cdot 1 \Rightarrow$ *representation* $(3, 0, 0, 0, 0, 3)$

ii) $42 = 34 + 8 \Rightarrow$ *representation* $(1, 0, 0, 1, 0, 0, 0, 0) = \langle 42 \rangle_U$

Δ

CHAPTER 2

Relations between Abelian complexity, balance properties, discrepancy, and incidence matrices

This chapter summarizes results by Boris Adamczewski, which have been presented in [1] and [2]. He has shown how to recognize whether a fixed point of a primitive substitution has Abelian complexity and balance function bounded.

2.1 Abelian complexity vs. balance properties

First, we show that Abelian complexity and balance function increased by one coincide on binary alphabets. In the general case of multiliteral alphabets, one is bounded if and only if the other one is bounded, too.

Proposition 2.1. Let $\mathbf{u}$ be an infinite word over a binary alphabet. Then for all $n \in \mathbb{N}^+$, it is satisfied

$$\mathcal{AC}_{\mathbf{u}}(n) = B_{\mathbf{u}}(n) + 1.$$

Δ

Proof. Let m be minimal and m' maximal such that $(m, n-m)$ and $(m', n-m')$ are Parikh vectors of some factors of $\mathbf{u}$. Then for l so that $m < l < m'$, it is also satisfied that $(l, n-l)$ is a Parikh vector of a factor of $\mathbf{u}$. Thus, for such a word $\mathbf{u}$, $\mathcal{AC}_{\mathbf{u}}(n) = m' - m + 1$ and $B_{\mathbf{u}}(n) = m' - m$. □

Proposition 2.2. Let $\mathbf{u}$ be an infinite word. The function $B_{\mathbf{u}}(n)$ is bounded if and only if the function $\mathcal{AC}_{\mathbf{u}}(n)$ is bounded. Δ

Proof. $\Rightarrow$: The fact that $B_{\mathbf{u}}(n)$ is bounded by some C implies that for all n, the cardinality of the set of Parikh vectors of factors $\mathcal{L}_n(\mathbf{u})$ must be bounded because these Parikh vectors can differ from each other at most by C in every entry.

$\Leftarrow$: For a a positive integer n, let us consider two Parikh vectors $\Psi(\mathbf{u}_{[i,i+n)})$ and $\Psi(\mathbf{u}_{[i+1,i+n+1)})$. Either they are the same (if $\mathbf{u}_i = \mathbf{u}_{i+n}$) or they differ just in two entries by 1 (if $\mathbf{u}_i \neq \mathbf{u}_{i+n}$). This fact implies that if $\mathcal{AC}_{\mathbf{u}}(n)$ is bounded by some C, then all Parikh vectors of all factors

from $\mathcal{L}_n(\mathbf{u})$ can differ at most by $C-1$ in every entry. In consequence, the balance function is bounded by C. □

2.2 Balance properties vs. discrepancy

Now, we present a proposition (first proven in [1]) whose consequence is that in the case of a fixed point of a primitive substitution, its discrepancy function is bounded if and only if its balance function is bounded.

Proposition 2.3. Let $\mathbf{u}$ be an infinite word over $\mathcal{A}$. Then $B_{\mathbf{u}}(n)$ is bounded if and only if the vector of letter frequencies μ exists and $D_{\mathbf{u}}(n)$ is bounded. Δ

Proof. $\Rightarrow$: Take $a \in \mathcal{A}$ arbitrary. We construct a sequence $(w^{(i)})_{i=1}^{+\infty}$, where for all $i \in \mathbb{N}^+$, $w^{(i)} \in \mathcal{L}_i(\mathbf{u})$ and $w^{(i)}$ satisfies

$$|w^{(i)}|_a \leq |v|_a \tag{2.1}$$

for all $v \in \mathcal{L}_i(\mathbf{u})$, i.e., $(w^{(i)})_{i=1}^{+\infty}$ is a sequence of factors of $\mathbf{u}$ low in the letter a.

Let $n, l \in \mathbb{N}^+$ and let us consider $w \in \mathcal{L}_{nl}(\mathbf{u})$. From Definition 1.17 and from the last equation we know that

$$0 \leq |w|_a - l|w^{(n)}|_a \leq lC \text{ and } 0 \leq |w|_a - n|w^{(l)}|_a \leq nC, \tag{2.2}$$

where C is a bound of the balance function.

Let us subtract inequalities (2.2) and divide them by nl, then

$$-\frac{C}{l} \leq \frac{|w^{(l)}|_a}{l} - \frac{|w^{(n)}|_a}{n} \leq \frac{C}{n}. \tag{2.3}$$

The equation (2.3) also expresses that $\frac{|w^{(n)}|_a}{n}$ is a Cauchy sequence. Let l tend to $+\infty$ in order to get

$$0 \leq L - \frac{|w^{(n)}|_a}{n} \leq \frac{C}{n}, \tag{2.4}$$

where $L = \lim_{l\to+\infty} \frac{|w^{(l)}|_a}{l}$.

Using Definition 1.17, we can compare occurrences of a in $w^{(n)}$ and in a prefix of $\mathbf{u}$:

$$0 \leq |\mathbf{u}_{[0,n)}|_a - |w^{(n)}|_a \leq C. \tag{2.5}$$

After subtracting (2.5) and (2.4) multiplied by n, we finally find formula

$$-C \leq Ln - |\mathbf{u}_{[0,n)}|_a \leq C,$$

which also implies

$$L = \lim_{n\to+\infty} \frac{|\mathbf{u}_{[0,n)}|_a}{n} = \mu_a$$

and the boundedness of $D_{\mathbf{u}}(n)$ by C.

$\Leftarrow$: The discrepancy function is bounded – it means

$$\big||\mathbf{u}_{[0,n)}|_a - n\mu_a\big| \leq C \tag{2.6}$$

for some C, all $a \in \mathcal{A}$, and for all n. Considering an arbitrary factor $x = \mathbf{u}_{[n,n+k)}$, we get

$$\big||\mathbf{u}_{[0,n)}|_a + |x|_a - n\mu_a - k\mu_a\big| \leq C. \tag{2.7}$$

After applying the triangle inequality and using (2.6) and (2.7), we can see that for all $a \in \mathcal{A}$ and all factors x of length k, it is satisfied

$$\big||x|_a - k\mu_a\big| \leq 2C.$$

Therefore, for any factors x_1 and x_2 of the same length k, we obtain the desired relation

$$\big||x_1|_a - |x_2|_a\big| \leq 4C.$$

□

Corollary 2.4. i) If the balance function $B_{\mathbf{u}}(n)$ of an infinite word $\mathbf{u}$ is bounded by a constant C, then the discrepancy function $D_{\mathbf{u}}(n)$ is also bounded by number C.

ii) If the discrepancy function $D_{\mathbf{u}}(n)$ is bounded by some constant C and there exists a vectors of frequencies μ, then the balance function $B_{\mathbf{u}}(n)$ is bounded by the number $4C$.

Δ

2.3 Discrepancy vs. incidence matrices

First, let us introduce notations. Since the following proof is based on Jordan normal form, let us denote Jordan blocks in the following way:

$$J(\lambda,\alpha) = \overbrace{\begin{pmatrix} \lambda & 1 & 0 & \dots & 0 & 0 \\ 0 & \lambda & 1 & & 0 & 0 \\ 0 & 0 & \lambda & \ddots & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & \\ 0 & 0 & 0 & & \lambda & 1 \\ 0 & 0 & 0 & \dots & 0 & \lambda \end{pmatrix}}^{\alpha \text{ columns}} \in \mathbb{C}^{\alpha,\alpha}.$$

After raising $J(\lambda,\alpha)$ to the power of n, we get

$$J^n(\lambda,\alpha) = \begin{pmatrix} \lambda^n & \binom{n}{1}\lambda^{n-1} & \binom{n}{2}\lambda^{n-2} & \dots & \binom{n}{\alpha-1}\lambda^{n-\alpha+1} \\ 0 & \lambda^n & \binom{n}{1}\lambda^{n-1} & \dots & \binom{n}{\alpha-2}\lambda^{n-\alpha+2} \\ 0 & 0 & \lambda^n & & \binom{n}{\alpha-3}\lambda^{n-\alpha+3} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & & \binom{n}{1}\lambda^{n-1} \\ 0 & 0 & 0 & \dots & \lambda^n \end{pmatrix} \tag{2.8}$$

Lemma 2.5. Let $\mathcal{J} \in \mathbb{C}^{r,r}$ be a matrix in Jordan normal form. Let us denote the greatest (in absolute value) eigenvalue of $\mathcal{J}$ by λ_1 and its algebraic multiplicity by α_1. In the case of more possible choices of λ_1, we choose some λ_1 with the greatest multiplicity. Then

$$|(\mathcal{J}^n)_{j,k}| \in \mathcal{O}\left(\binom{n}{\alpha_1 - 1}|\lambda_1|^n\right)$$

for all $j,k \in \{1,2,\dots,r\}$. Δ

Proof. Matrix $\mathcal{J}$ is in Jordan normal form, if

$$\mathcal{J} = \begin{pmatrix} J(\lambda_1, \alpha_1) & 0 & \dots & 0 \\ 0 & J(\lambda_2, \alpha_2) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & J(\lambda_s, \alpha_s) \end{pmatrix}. \tag{2.9}$$

Raising $\mathcal{J}$ to the power of n is the same as raising all its diagonal Jordan blocks.

$$\mathcal{J}^n = \begin{pmatrix} J^n(\lambda_1, \alpha_1) & 0 & \dots & 0 \\ 0 & J^n(\lambda_2, \alpha_2) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & J^n(\lambda_s, \alpha_s) \end{pmatrix}.$$

From (2.8) and considering n great enough, we can make an estimation for an element of one Jordan block:

$$|(J^n(\lambda_i, \alpha_i))_{j,k}| \leq \begin{cases} |\lambda_i|^n \leq \binom{n}{\alpha_i - 1} |\lambda_i|^n & \text{if } |\lambda_i| \geq 1; \\ \binom{n}{\alpha_i - 1} |\lambda_i|^{n-\alpha_i+1} & \text{if } |\lambda_i| < 1. \end{cases} \tag{2.10}$$

Since all elements of $\mathcal{J}$ are equal to zero or can be estimated by (2.10), we can make a global estimation for the whole matrix by choosing the "worst" Jordan block and we get the statement of the lemma. □

Lemma 2.6. Let $\mathbf{u}$ be a fixed point of a primitive substitution φ. Then the following statements hold.

i) For all $n \in \mathbb{N}^+$, there exists $N_n \in \mathbb{N}$ and words $E_0, E_1, \dots, E_{N_n}$ with $E_{N_n} \neq \varepsilon$ such that

$$\mathbf{u}_{[0,n)} = \varphi^{N_n}(E_{N_n}) \varphi^{N_n - 1}(E_{N_n - 1}) \cdots \varphi(E_1) E_0,$$

where for all $i \in \{0, 1, \dots, N_n\}$, there exists a letter $a^{(i)} \in \mathcal{A}$ such that E_i is a prefix of $\varphi(a^{(i)})$ and $E_i \neq \varphi(a^{(i)})$.

ii) There exist constants $C_1, C_2 > 0$ such that N_n and n satisfy

$$C_1 \Lambda^{N_n} \leq n \leq C_2 \Lambda^{N_n},$$

where Λ is the greatest eigenvalue (in absolute value) of the incidence matrix of φ.

Δ

Proof. Let us denote $\mathbf{u}_0$ by a. Then $\varphi(a) = av$ for some non-empty v and $\mathbf{u} = \lim_{m \to +\infty} \varphi^m(a)$.

i) Let us find the smallest $m_0 \in \mathbb{N}^+$ such that $\mathbf{u}_{[0,n)}$ is a proper prefix of $\varphi(\mathbf{u}_{[0,m_0)})$. Then we can express $\mathbf{u}_{[0,n)}$ as

$$\mathbf{u}_{[0,n)} = \varphi(\mathbf{u}_{[0,m_0-1)}) E_0,$$

where E_0 is a prefix of $\varphi(a^{(0)})$ and $a^{(0)} = \mathbf{u}_{m_0-1}$.

In the same way and by the same reasoning, we can repeat the procedure in order to find m_1 such that

$$\mathbf{u}_{[0,m_0-1)} = \varphi(\mathbf{u}_{[0,m_1-1)})E_1$$

and we obtain

$$\mathbf{u}_{[0,n)} = \varphi^2(\mathbf{u}_{[0,m_1-1)})\varphi(E_1)E_0,$$

where E_1 is a prefix of $\varphi(a^{(1)})$, $a^{(1)} = \mathbf{u}_{m_1-1}$, and $\mathbf{u}_{[0,n)}$ is a proper prefix of $\varphi^2(\mathbf{u}_{[0,m_1)})$. We continue with this procedure until $m_i = 1$. Then:

$$\begin{aligned} \mathbf{u}_0 = \mathbf{u}_{[0,m_i)} = a^{(i)} &= a, \\ \mathbf{u}_{[0,m_i-1)} &= \varepsilon, \\ \mathbf{u}_{[0,m_{i-1}-1)} &= \varphi(\varepsilon)E_i. \end{aligned}$$

To sum up, after we set $N_n := i$, we get

$$\mathbf{u}_{[0,n)} = \varphi^{N_n}(E_{N_n})\varphi^{N_n-1}(E_{N_n-1})\cdots\varphi(E_1)E_0,$$

where $E_{N_n} \neq \varepsilon$.

ii) We have $|\varphi^{N_n}(a)| \leq n < |\varphi^{N_n+1}(a)|$. Without loss of generality, let us consider $a = 0$. Then we can express $|\varphi^{N_n}(a)|$ using M_φ in Jordan normal form as

$$|\varphi^{N_n}(a)| = (1,1,\ldots,1)M_\varphi^{N_n}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} = (1,1,\ldots,1)P\begin{pmatrix}\Lambda^{N_n} & 0 & \ldots & 0\\ 0 & & & \\ \vdots & & \mathcal{J}_0^{N_n} & \\ 0 & & & \end{pmatrix}P^{-1}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}, \quad (2.11)$$

where P denotes the basis transformation matrix satisfying $\mathcal{J} = P^{-1}M_\varphi P$ and

$$\mathcal{J}_0 = \begin{pmatrix} J(\lambda_2,\alpha_2) & 0 & \ldots & 0 \\ 0 & J(\lambda_3,\alpha_3) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & J(\lambda_s,\alpha_s) \end{pmatrix}.$$

It follows from

$$M_\varphi P = P\begin{pmatrix}\Lambda & 0 & \ldots & 0\\ 0 & & & \\ \vdots & & \mathcal{J}_0 & \\ 0 & & & \end{pmatrix} \quad (2.12)$$

that $P_{\bullet 1}$ is an eigenvector of M_φ because $M_\varphi P_{\bullet 1} = \Lambda P_{\bullet 1}$. Hence, without loss of generality, we can choose P such that $P_{\bullet 1} = \mu^\intercal$.

From (2.11), we can factor out Λ^{N_n}. Let us denote

$$L_{N_n} = (1,1,\ldots,1)P\begin{pmatrix}1 & 0 & \ldots & 0\\ 0 & & & \\ \vdots & & \frac{1}{\Lambda^{N_n}}\mathcal{J}_0^{N_n} & \\ 0 & & & \end{pmatrix}P^{-1}\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}.$$

Since Λ is in absolute value greater than any other eigenvalue, we have

$$\lim_{N_n\to+\infty} L_{N_n} = (1,1,\dots,1)\begin{pmatrix} & 0 & \dots & 0 \\ P_{\bullet 1} & \vdots & & \vdots \\ & 0 & \dots & 0 \end{pmatrix}(P^{-1})_{\bullet 1} = (P^{-1})_{1,1}.$$

Let us prove that $(P^{-1})_{1,1} > 0$. From (2.12), we obtain

$$M_\varphi^\intercal \left(P^{-1}\right)^\intercal = \left(P^{-1}\right)^\intercal \begin{pmatrix} \Lambda & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \mathcal{J}_0^\intercal & \\ 0 & & & \end{pmatrix}.$$

Hence, $\left(P^{-1}\right)_{\bullet 1}^\intercal$ is the eigenvector of $M_\varphi^\intercal$ associated with Λ. Then $(P^{-1})_{1,1} \neq 0$ follows from Theorem 1.7 and since $\lim_{n\to+\infty} \frac{|\varphi^{N_n}(a)|}{\Lambda^{N_n}} = (P^{-1})_{1,1}$ and $\Lambda > 1$ by Theorem 1.7, we have $(P^{-1})_{1,1} > 0$.

As $(P^{-1})_{1,1} > 0$, for large enough n, it holds

$$\frac{1}{2}(P^{-1})_{1,1}\Lambda^{N_n} < |\varphi^{N_n}(a)| < \frac{3}{2}(P^{-1})_{1,1}\Lambda^{N_n}.$$

Consequently, for large enough n, we have

$$\underbrace{\frac{1}{2}(P^{-1})_{1,1}}_{C_1}\Lambda^{N_n} < |\varphi^{N_n}(a)| \leq n < |\varphi^{N_n+1}(a)| < \underbrace{\frac{3}{2}(P^{-1})_{1,1}\max_{b\in\mathcal{A}}|\varphi(b)|}_{C_2}\Lambda^{N_n}.$$

Therefore, there exist constants $C_1, C_2 > 0$ such that N_n and n satisfy

$$C_1\Lambda^{N_n} \leq n \leq C_2\Lambda^{N_n}.$$

□

Proposition 2.7. Let φ be a primitive substitution with a fixed point $\mathbf{u}$ and the incidence matrix M_φ. Let us denote the second greatest (in absolute value) eigenvalue of M_φ by λ_2. If $|\lambda_2| < 1$, then the discrepancy function $D_{\mathbf{u}}(n)$ is bounded. Δ

Proof. For all letters $a \in \mathcal{A} = \{0,1,\dots,d-1\}$, we define vectors $f_0, f_1, \dots, f_{d-1}$ as:

$$\begin{aligned} f_0 &= \left(1-\mu_0, \quad -\mu_0, \quad \dots, \quad -\mu_0\right), \\ f_1 &= \left(-\mu_1, \quad 1-\mu_1, \quad \dots, \quad -\mu_1\right), \\ &\vdots \\ f_{d-1} &= \left(-\mu_{d-1}, \quad -\mu_{d-1}, \quad \dots, \quad 1-\mu_{d-1}\right). \end{aligned}$$

As a consequence of the definition, we obtain for all letters $a \in \mathcal{A}$

$$f_a\mu^\intercal = 0, \tag{2.13}$$

and

$$f_a(\Psi(\mathbf{u}_{[0,n)}))^\intercal = |\mathbf{u}_{[0,n)}|_a - n\mu_a =: D_{\mathbf{u},a}(n).$$

Now, the discrepancy can be expressed as $D_{\mathbf{u}}(n) = \max_{a\in\mathcal{A}}\{|D_{\mathbf{u},a}(n)|\}$ and we will estimate $D_{\mathbf{u},a}(n)$ for all $a \in \mathcal{A}$.

Let E be a prefix of $\varphi(a)$ for some $a \in \mathcal{A}$. Then

$$f_a(\Psi(\varphi^k(E)))^\intercal = f_a M_\varphi^k(\Psi(E))^\intercal = \underbrace{f_a P}_{=(0,v) \text{ due to (2.13)}} \begin{pmatrix} \Lambda^k & 0 \\ 0 & \mathcal{J}_0^k \end{pmatrix} P^{-1}(\Psi(E))^\intercal = (0, v\mathcal{J}_0^k)P^{-1}(\Psi(E))^\intercal, \tag{2.14}$$

where $\mathcal{J}_0$ and P are the same as in the proof of Lemma 2.6, $P_{\bullet 1} = \mu^\intercal$, and v is a vector.

In (2.14), the only expression depending on k is $\mathcal{J}_0^k$. Nevertheless, any element of $\mathcal{J}_0^k$ can be estimated using Lemma 2.5, therefore,

$$|f_a(\Psi(\varphi^k(E)))^\intercal| \le const \cdot \binom{k}{\alpha_2 - 1} |\lambda_2|^k. \tag{2.15}$$

Now, we use (2.15) in order to estimate $D_{\mathbf{u},a}(n)$.

$$D_{\mathbf{u},a}(n) = f_a(\Psi(\mathbf{u}_{[0,n)}))^\intercal = f_a \sum_{k=0}^{N_n} (\Psi(\varphi^k(E_k)))^\intercal = \sum_{k=0}^{N_n} f_a(\Psi(\varphi^k(E_k)))^\intercal. \tag{2.16}$$

If we compose (2.15) and (2.16), we get

$$|D_{\mathbf{u},a}(n)| \le \sum_{k=0}^{N_n} |f_a(\Psi(\varphi^k(E_k)))^\intercal| \le \sum_{k=0}^{N_n} const \cdot \underbrace{\binom{k}{\alpha_2 - 1} |\lambda_2|^k}_{b_k}$$

We use the limit ratio test

$$L = \lim_{k\to+\infty} \frac{b_{k+1}}{b_k} = \lim_{k\to+\infty} \frac{|\lambda_2|(1+k)}{2+k-\alpha_2} = |\lambda_2|,$$

which states that if $|\lambda_2| < 1$, then $\sum_{k=0}^{+\infty} b_k$ converges. Hence, $D_{\mathbf{u},a}(n)$ is bounded under this condition.

□

Now, we have all propositions necessary to prove the most important theorem of this chapter.

Theorem 2.8. Let φ be a primitive substitution with a fixed point $\mathbf{u}$ and λ_2 be the second greatest (in absolute value) eigenvalue of its incidence matrix. If $|\lambda_2| < 1$, then the balance function $B_{\mathbf{u}}(n)$ and the Abelian complexity $\mathcal{AC}_{\mathbf{u}}(n)$ are bounded. Δ

Proof. The theorem is a consequence of Proposition 2.2, Proposition 2.3, and Proposition 2.7. □

CHAPTER 3

Balance properties and Abelian complexity of words over binary alphabets

In this chapter, we deal with binary words. As shown in Proposition 2.1, Abelian complexity is identically equal to the balance function increased by one.

At first, we summarize known results for Sturmian words and the Thue-Morse word. Then, as a new result, we derive an explicit formula for Abelian complexity of infinite words associated with quadratic Parry numbers.

3.1 Sturmian words

As it was introduced in Chapter 1, Sturmian words are aperiodic words with the lowest possible factor complexity. Essential properties of Sturmian words have been summarized in [5]. As an example of a Sturmian word, we can mention the Fibonacci word which is the fixed point of the substitution

$$\varphi : \begin{array}{l} 0 \to 01; \\ 1 \to 0. \end{array}$$

For the first time, Abelian complexity of Sturmian words has been described in [8].

Theorem 3.1. Let $\mathbf{u}$ be an aperiodic infinite word. Then $\mathbf{u}$ is Sturmian if and only if $\mathbf{u}$ satisfies

$$\mathcal{AC}_{\mathbf{u}}(n) = 2$$

for all $n \in \mathbb{N}^+$. Δ

It is natural to ask whether there exist aperiodic infinite words having their Abelian complexity constant, i.e., identically equal to higher integers. We answer the question in Section 4.3.

3.2 Thue-Morse word

The Thue-Morse word is defined as the fixed point $\mathbf{TM}_0 = \lim_{n\to+\infty} \varphi^n(0)$ of the Thue-Morse substitution

$$\varphi : \begin{array}{l} 0 \to 01; \\ 1 \to 10. \end{array}$$

Description of Abelian complexity is quite easy. We take advantage of the fact that both letter images are of the same length 2 and have the same Parikh vectors. Hence, there can be only two Parikh vectors for an odd length and three for an even length. This result is summarized in the next theorem that has been first published in [19].

Theorem 3.2. Let $\mathbf{TM}_0$ be the Thue-Morse word. Then

$$\mathcal{AC}_{\mathbf{TM}_0}(n) = \begin{cases} 2 & \text{for } n \text{ odd;} \\ 3 & \text{for } n \text{ even.} \end{cases}$$

Δ

3.3 Words associated with quadratic Parry numbers

This section has been already submitted as [4]. Here, we provide the full article with only small changes.

Let us determine Abelian complexity of infinite words associated with quadratic Parry numbers. These words are usually defined via the so-called β-integers corresponding to Parry numbers β, for details see [6]. However, here we skip the procedure for the sake of brevity and introduce them directly using the notion of substitution.

It has been shown in [11] that infinite words $\mathbf{u}_\beta$ associated with quadratic Parry numbers β are in fact fixed points of certain substitutions. These substitutions are of two types:

- The class of substitutions providing infinite words associated with **quadratic simple Parry numbers** is given by

$$\varphi : \begin{array}{l} 0 \to 0^p1; \\ 1 \to 0^q; \end{array}$$

for $p, q \in \mathbb{N}^+$, $p \geq q$.

The corresponding incidence matrix is

$$M_\varphi = \begin{pmatrix} p & q \\ 1 & 0 \end{pmatrix}.$$

The infinite word $\mathbf{u}_\beta$ is the only fixed point of this substitution, i.e., $\mathbf{u}_\beta = \varphi(\mathbf{u}_\beta)$, and it can be obtained when iterating φ on the letter 0 infinitely many times, i.e., $\mathbf{u}_\beta = \lim_{n\to\infty} \varphi^n(0)$.

- The class of substitutions providing infinite words associated with **quadratic non-simple Parry numbers** is given by

$$\varphi : \begin{array}{l} 0 \to 0^p1; \\ 1 \to 0^q1; \end{array}$$

for $p, q \in \mathbb{N}^+$, $p > q$.

The associated incidence matrix is

$$M_\varphi = \begin{pmatrix} p & q \\ 1 & 1 \end{pmatrix}.$$

Again, $\mathbf{u}_\beta = \lim_{n\to\infty} \varphi^n(0)$ is the only fixed point of φ.

Throughout the whole section, the symbol φ denotes a morphism associated with a quadratic (simple or non-simple) Parry number and $\mathbf{u}_\beta$ is exclusively used for the word $\lim_{n\to\infty} \varphi^n(0)$.

For infinite words associated with quadratic Parry numbers, the optimal balance bounds are already known [22, 6]. By Proposition 2.1, the maximum of Abelian complexity is immediately determined.

Corollary 3.3. Abelian complexity of $\mathbf{u}_\beta$ satisfies

- $\max_{n\in\mathbb{N}^+}\{\mathcal{AC}_{\mathbf{u}_\beta}(n)\} = 2 + \lfloor \frac{p-1}{p+1-q} \rfloor$ in the case of simple Parry numbers;
- $\max_{n\in\mathbb{N}^+}\{\mathcal{AC}_{\mathbf{u}_\beta}(n)\} = 1 + \lceil \frac{p-1}{q} \rceil$ in the case of non-simple Parry numbers.

Δ

The target of this section is to derive an explicit formula for $\mathcal{AC}_{\mathbf{u}_\beta}(n)$ for all $n \in \mathbb{N}^+$.

Remark 3.4. It is seen from Corollary 3.3 that for a simple Parry number β with $q = 1$ or for a non-simple Parry number β with $p = q + 1$, Abelian complexity satisfies $\mathcal{AC}_{\mathbf{u}_\beta}(n) = 2$ for all $n \in \mathbb{N}^+$. This result follows also from [13], where it has been shown that for such β the infinite word $\mathbf{u}_\beta$ is Sturmian, thus its Abelian complexity is constant and equal to 2 by Theorem 3.1. Moreover, these cases are the only ones among all Parry numbers β (not only the quadratic ones) for which the infinite word $\mathbf{u}_\beta$ is Sturmian, see [14].

For calculation of Abelian complexity, the following simple observation deduced from (1.2) will be important.

Observation 3.5. For all $n \in \mathbb{N}^+$, it holds

$$\Psi\left(\varphi^n(0)\right)^\intercal = M_\varphi^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |\varphi^n(0)|_0 = (1,0) M_\varphi^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |\varphi^n(0)|_1 = (0,1) M_\varphi^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Δ

Let us define a strictly increasing sequence $U = (U_n)_{n=0}^{+\infty}$ by $U_n = |\varphi^n(0)|$, i.e.,

$$U_n = (1,1) M_\varphi^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{3.1}$$

Furthermore, let us denote by $\langle n \rangle_U$ the normal U-representation of n, see Section 1.2. Recall that the normal U-representation of $n \in \mathbb{N}^+$ is equal to

$$\langle n \rangle_U = (d_N, d_{N-1}, \ldots, d_1, d_0) \tag{3.2}$$

if $n = \sum_{j=0}^{N} d_j U_j$ and the representation is obtained by Algorithm 1.

Remark 3.6. Note that N can be chosen as any integer such that $n < U_{N+1}$, i.e., not necessarily the smallest one satisfying $U_N \leq n < U_{N+1}$. As a result, the normal U-representation of n can start with a block of zeros, and the representation $(0, 0, ..., 0, d_N, d_{N-1}, ..., d_0)$ is equivalent to $(d_N, d_{N-1}, ..., d_0)$.

Since the inequality $U_{N+1} = |(\varphi^N(0))^p \varphi^N(1)| < |(\varphi^N(0))^{p+1}| = (p+1)U_N$ holds for all $N \in \mathbb{N}$, one can see that the coefficients in normal U-representations are less than or equal to p.

The following proposition is a particular case of a general theorem concerning substitutions associated with Parry numbers provided in [11].

Proposition 3.7. Let u be the prefix of $\mathbf{u}_\beta$ of length n, $n \in \mathbb{N}^+$. If $\langle n \rangle_U = (d_N, d_{N-1}, \ldots, d_1, d_0)$, then $u = \left(\varphi^N(0)\right)^{d_N} \left(\varphi^{N-1}(0)\right)^{d_{N-1}} \cdots (\varphi(0))^{d_1} 0^{d_0}$, where the zero-power of a word w is defined as the empty word, i.e., $w^0 = \varepsilon$. Δ

Note that the above proposition finds, for $d_N \neq 0$, the words $E_0, E_1, \ldots, E_{N_n}$ from Lemma 2.6.

It turns out that for any simple or non-simple quadratic Parry number β, one can find two infinite words, let us denote them by $\mathbf{v}$ and $\mathbf{w}$, that have the following properties.

- For any finite prefix $\hat{v}$ of $\mathbf{v}$, it holds:

  i) $\hat{v}$ is a factor of $\mathbf{u}_\beta$.

  ii) For any factor u of $\mathbf{u}_\beta$,
  $$|u| = |\hat{v}| \quad \Rightarrow \quad |u|_1 \geq |\hat{v}|_1 .$$
  In other words, the number of occurrences of the letter 1 in factors of $\mathbf{u}_\beta$ of a given length n attains its *minimum* in the prefix of $\mathbf{v}$ of length n.

- For any finite prefix $\hat{w}$ of $\mathbf{w}$, it holds:

  i) $\hat{w}$ is a factor of $\mathbf{u}_\beta$.

  ii) For any factor u of $\mathbf{u}_\beta$,
  $$|u| = |\hat{w}| \quad \Rightarrow \quad |u|_1 \leq |\hat{w}|_1 .$$
  In other words, the number of occurrences of the letter 1 in factors of $\mathbf{u}_\beta$ of a given length n attains its *maximum* in the prefix of $\mathbf{w}$ of length n.

Note that $\mathbf{v}$ and $\mathbf{w}$ are unique. These words $\mathbf{v}$ and $\mathbf{w}$ play an essential role in computation of Abelian complexity.

Proposition 3.8. Let $\mathbf{v}$ and $\mathbf{w}$ be the infinite words defined above. Then Abelian complexity of $\mathbf{u}_\beta$ can be expressed for all $n \in \mathbb{N}^+$ by the formula
$$\mathcal{AC}_{\mathbf{u}_\beta}(n) = 1 + |\hat{w}|_1 - |\hat{v}|_1 , \tag{3.3}$$
where $\hat{w}$ and $\hat{v}$ are the prefixes of $\mathbf{w}$ and $\mathbf{v}$, respectively, of the same length n. Δ

Proof. Using properties of $\mathbf{v}$ and $\mathbf{w}$, it follows that

$$\max_{u,w\in\mathcal{L}_n(\mathbf{u}_\beta)}\{|v|_0-|w|_0\}=|\hat{v}|_0-|\hat{w}|_0=|\hat{w}|_1-|\hat{v}|_1=B_{\mathbf{u}_\beta}(n),$$

where $\hat{v}$, $\hat{w}$ are the prefixes of length n of $\mathbf{v}$ and $\mathbf{w}$, respectively. The statement is then a direct consequence of Proposition 2.1. □

For an application of the formula (3.3), it is necessary to find the words $\mathbf{v}$ and $\mathbf{w}$ and their structure which will be done in the rest of the section, separately for the simple and the non-simple Parry case.

3.3.1 Non-simple Parry case

At first, we will determine Abelian complexity of infinite words $\mathbf{u}_\beta$ associated with quadratic non-simple Parry numbers, which are fixed points of the substitution given by

$$\varphi(0)=0^p1\,,\quad \varphi(1)=0^q1\,,\qquad p>q\geq 1\,.$$

Let us recall that according to Corollary 3.3, the maximum of Abelian complexity of these words equals $1+\lceil\frac{p-1}{q}\rceil$. Moreover, by Remark 3.4, in the special case $p=q+1$ it holds $\mathcal{AC}_{\mathbf{u}_\beta}(n)=2$ for all $n\in\mathbb{N}^+$. The *non-simple* case – despite its name – is slightly easier to treat than the *simple* one because the words $\mathbf{v}$ and $\mathbf{w}$ have been already found in the paper [6]. Let us just adopt them from there:

$$\mathbf{v}=\mathbf{u}_\beta,\qquad \mathbf{w}=\lim_{n\to\infty}w^{(n)},\tag{3.4}$$

where $w^{(0)}=1$ and $w^{(n)}=1\varphi(w^{(n-1)})$ for $n\in\mathbb{N}^+$.

In order to determine $|\hat{v}|_1$ and $|\hat{w}|_1$ for any prefix $\hat{v}$ of $\mathbf{v}=\mathbf{u}_\beta$ and $\hat{w}$ of $\mathbf{w}$, we will use the following observation, which is a direct consequence of the substitution form.

Observation 3.9. For any $u\in\mathcal{A}^\star$, it holds $|\varphi(u)|_1=|u|$. In particular, $|\varphi^j(0)|_1=U_{j-1}$ for all $j\in\mathbb{N}^+$.

With the previous observation and Proposition 3.7 in hand, we can compute $|\hat{v}|_1$ for any prefix $\hat{v}$ of $\mathbf{v}=\mathbf{u}_\beta$.

Lemma 3.10. Let $n\in\mathbb{N}^+$ with $\langle n\rangle_U=(d_N,d_{N-1},\ldots,d_1,d_0)$ and let $\hat{v}$ be the prefix of $\mathbf{v}=\mathbf{u}_\beta$ of length n. Then

$$|\hat{v}|_1=\sum_{j=1}^{N}d_jU_{j-1}\,.$$

Δ

The following lemma will play an essential role in calculation of $|\hat{w}|_1$ for any prefix $\hat{w}$ of $\mathbf{w}$.

Lemma 3.11. For any $N\in\mathbb{N}^+{}_0$, the word $w^{(N)}$ has the following two properties:

1. $w^{(N)}$ is a palindrome,

2. $w^{(N)}$ is a proper suffix of $\varphi^{N+1}(0)$.

Δ

Proof. Let us proceed by induction on N.

1. $w^{(0)} = 1$ is a palindrome. Let $N \geq 0$ and let us suppose that $w^{(N)}$ is a palindrome. We will prove that $w^{(N+1)}$ is a palindrome, too. We have $w^{(N+1)} = 1\varphi(w^{(N)}) = 1\varphi(\overline{w^{(N)}})$. Denote $w^{(N)} = w_0 w_1 \cdots w_n$, where w_i are letters. Since for every $a \in \mathcal{A}$ it holds $1\varphi(a) = \overline{\varphi(a)}1$, we deduce $1\varphi(\overline{w^{(N)}}) = 1\varphi(w_n)\cdots\varphi(w_1)\varphi(w_0) = \overline{\varphi(w_n)}\cdots\overline{\varphi(w_1)}\,\overline{\varphi(w_0)}1 = \overline{\varphi(w^{(N)})}1 = \overline{1\varphi(w^{(N)})} = \overline{w^{(N+1)}}$.
2. $w^{(0)} = 1$ is a proper suffix of $\varphi(0)$. Let $N \geq 1$ and let us assume $\varphi^N(0) = uw^{(N-1)}$, where $u \neq \varepsilon$. Thus $\varphi^{N+1}(0) = \varphi(u)\varphi(w^{(N-1)})$. Since the last letter of $\varphi(u)$ is 1, we have proven that $w^{(N)} = 1\varphi(w^{(N-1)})$ is a proper suffix of $\varphi^{N+1}(0)$.

□

We have prepared everything to determine $|\hat{w}|_1$ for any prefix $\hat{w}$ of $\mathbf{w}$.

Lemma 3.12. Let $n \in \mathbb{N}^+$ and let $\hat{w}$ be the prefix of $\mathbf{w}$ of length n. Take k arbitrary such that $n \leq |w^{(k)}|$ and denote $\langle U_{k+1} - n\rangle_U = (e_k, e_{k-1}, \ldots, e_1, e_0)$. Then

$$|\hat{w}|_1 = U_k - \sum_{j=1}^{k} e_j U_{j-1}\,.$$

Δ

Proof. According to the choice of k, the word $\hat{w}$ is a prefix of $w^{(k)}$, and using the first statement of Lemma 3.11, $\overline{\hat{w}}$ is a suffix $w^{(k)}$. By the second statement of Lemma 3.11, we have $\varphi^{k+1}(0) = u\overline{\hat{w}}$, where u is a prefix of $\mathbf{u}_\beta$, $u \neq \varepsilon$. Therefore, $|\hat{w}|_1 = |\varphi^{k+1}(0)|_1 - |u|_1$. We obtain $|\varphi^{k+1}(0)|_1 = U_k$ from Observation 3.9, and since $\langle |u|\rangle_U = \langle U_{k+1} - n\rangle_U = (e_k, e_{k-1}, \ldots, e_1, e_0)$ by assumption, it holds $|u|_1 = \sum_{j=1}^{k} e_j U_{j-1}$ according to Lemma 3.10. □

As an immediate consequence of Proposition 3.7 and Lemmas 3.10 and 3.12, we get the main theorem.

Theorem 3.13. Let $\mathbf{u}_\beta$ be the fixed point of the substitution φ defined in (3.3). Let $(U_n)_{n=0}^{\infty}$ be the sequence defined in (3.1). Abelian complexity of the infinite word $\mathbf{u}_\beta$ is given for all $n \in \mathbb{N}^+$ by the formula

$$\mathcal{AC}_{\mathbf{u}_\beta}(n) = 1 + U_k - \sum_{j=1}^{k} (d_j + e_j) U_{j-1}\,,$$

where

- k is arbitrary such that $n \leq |w^{(k)}|$, where $w^{(k)}$ is defined in (3.4),
- $\langle n\rangle_U = (d_k, d_{k-1}, \ldots, d_1, d_0)$,
- $\langle U_{k+1} - n\rangle_U = (e_k, e_{k-1}, \ldots, e_1, e_0)$.

Δ

In the end of this section, we estimate the minimal index k from Theorem 3.13.

Lemma 3.14. Let $n \in \mathbb{N}^+$ and let the number $N \in \mathbb{N}$ satisfy $U_N \leq n < U_{N+1}$. Then

$$\left|w^{(N)}\right| \leq n < \left|w^{(N+2)}\right| .$$

Δ

Proof. Let us recall that for all $N \in \mathbb{N}$, $w^{(N)} = 1\varphi(1)\varphi^2(1)\cdots\varphi^N(1)$. In order to prove the first inequality ($\leq$), it suffices to show that $|w^{(N)}| \leq U_N$. This can be done by induction on N, using the identities $\varphi^{N+1}(0) = \left(\varphi^N(0)\right)^p \varphi^N(1)$, $\varphi^{N+1}(1) = \left(\varphi^N(0)\right)^q \varphi^N(1)$ and the inequality $p > q$. As for the second inequality ($<$), $n < U_{N+1}$ implies

$$n < |\varphi^{N+1}(0)| < |\left(\varphi^{N+1}(0)\right)^q \varphi^{N+1}(1)| = |\varphi^{N+2}(1)| < \left|1\varphi(1)\cdots\varphi^{N+2}(1)\right| = |w^{(N+2)}| .$$

□

Lemma 3.14 tells us that if $n \in \mathbb{N}^+$ satisfies $U_N \leq n < U_{N+1}$, then the minimal index k from Theorem 3.13 is either $N+1$ or $N+2$. Hence the choice $k = N+2$ works universally.

Example 3.15. *Let $p = 3$ and $q = 1$. Let us calculate $\mathcal{AC}_{\mathbf{u}_\beta}(n)$ for $n = 7$. Using (3.1), we obtain $(U_n)_{n=0}^{\infty} = (1, 4, 14, 48, 164, \ldots)$. Applying Lemma 3.14, we can put $k = 3$ in Theorem 3.13. Now, we have to get the normal U-representations:*

$$\langle 7\rangle_U = (0, 0, 1, 3) = (d_3, d_2, d_1, d_0) \text{ and } \langle U_4 - 7\rangle_U = \langle 157\rangle_U = (3, 0, 3, 1) = (e_3, e_2, e_1, e_0).$$

Finally, using Theorem 3.13, we find $\mathcal{AC}_{\mathbf{u}_\beta}(7) = 1 + U_3 - \sum_{j=1}^{3}(d_j + e_j)U_{j-1} = 3$.

Since n is small, we can illustrate the situation by observing the prefix $\varphi^3(0)$ of $\mathbf{u}_\beta$:

$$\varphi^3(0) = \underbrace{0001000}_{\hat{v}}\,1000101000100010001010001000100010\,\underbrace{1000}_{\overline{\hat{w}}}\,10\,\overbrace{1\overbrace{0\overbrace{0}^{w^{(0)}}}^{w^{(1)}}}^{w^{(2)}}$$

We see that $|\hat{w}|_1 - |\hat{v}|_1 = 2$ and therefore, indeed, $\mathcal{AC}_{\mathbf{u}_\beta}(7) = 3$. (Moreover, one can notice that in this case, it is sufficient to put $k = 2$ in Theorem 3.13.) Δ

3.3.2 Simple Parry case

In this section, we study Abelian complexity of infinite words associated with quadratic simple Parry numbers, i.e., we deal with the fixed point $\mathbf{u}_\beta$ of the substitution φ given by

$$\varphi(0) = 0^p 1\,, \quad \varphi(1) = 0^q\,, \qquad p \geq q \geq 1\,.$$

Case $q = 1$

When β is a quadratic simple Parry number, calculation of $\mathcal{AC}_{\mathbf{u}_\beta}(n)$ turns out to be more effective when the cases $q = 1$ and $q > 1$ are treated separately. Let us start with $q = 1$. As we have explained in Remark 3.4, in this case it holds $\mathcal{AC}_{\mathbf{u}_\beta}(n) = 2$ for all $n \in \mathbb{N}^+$.

Case $q > 1$

From now on, let us assume that $q > 1$. By Corollary 3.3, the maximum of Abelian complexity is $2 + \lfloor \frac{p-1}{p+1-q} \rfloor$. In order to find an explicit expression for $\mathcal{AC}_{\mathbf{u}_\beta}(n)$ for all $n \in \mathbb{N}^+$, we have to describe the infinite words $\mathbf{v}$ and $\mathbf{w}$ from Proposition 3.8. These words, as it will be proven in Proposition 3.18 below, are:

$$\mathbf{w} = \lim_{n\to\infty} w^{(n)} ,$$

where

$$\begin{aligned} w^{(0)} &= 1 \\ 0^p w^{(n)} &= \varphi^2(w^{(n-1)}) \quad \text{for all } n \in \mathbb{N}^+ , \end{aligned}$$

and

$$\mathbf{v} = \lim_{n\to\infty} v^{(n)} ,$$

where $v^{(n)} = \varphi(w^{(n)}) \quad$ for all $n \in \mathbb{N}$.

To demonstrate how $\mathbf{v}$ and $\mathbf{w}$ are constructed, here are their prefixes written explicitly:

$$\begin{aligned} w^{(n)} &= 1\varphi(0^{q-1})\varphi^3(0^{q-1})\varphi^5(0^{q-1})\cdots\varphi^{2n-3}(0^{q-1})\varphi^{2n-1}(0^{q-1}) , \\ v^{(n)} &= 0^q\varphi^2(0^{q-1})\varphi^4(0^{q-1})\varphi^6(0^{q-1})\cdots\varphi^{2(n-1)}(0^{q-1})\varphi^{2n}(0^{q-1}) . \end{aligned} \tag{3.5}$$

Both $w^{(n)}$ and $v^{(n)}$ are factors of $\mathbf{u}_\beta$: it can be easily proven by induction on n that $w^{(n)}$ is a suffix of $\varphi^{2n}(1)$, and $v^{(n)}$ is the image by φ of $w^{(n)}$.

We observe from the construction of the words $\mathbf{v}$ and $\mathbf{w}$ that they are related via φ:

Observation 3.16. The infinite words $\mathbf{v}$ and $\mathbf{w}$ satisfy the relations

$$\mathbf{v} = \varphi(\mathbf{w}) , \qquad 0^p\mathbf{w} = \varphi(\mathbf{v}) .$$

Δ

The following simple observation will be used as a tool in the proof of Proposition 3.18.

Observation 3.17.

1. If 10^k1 is a factor of $\mathbf{u}_\beta$, then $k = p$ or $k = q + p$.
2. If $u1$ is a factor of $\mathbf{u}_\beta$ such that it has the prefix 0^p1 or 0^{q+p}, then there exists a unique factor u' of $\mathbf{u}_\beta$ satisfying $\varphi(u'0) = u1$. Moreover, $|u'0| < |u1|$.

Δ

Proposition 3.18. Let u be a factor of $\mathbf{u}_\beta$, let $\hat{w}$ be the prefix of $\mathbf{w}$ of length $|u|$ and let $\hat{v}$ be the prefix of $\mathbf{v}$ of length $|u|$. Then

$$|\hat{w}|_1 \geq |u|_1 \geq |\hat{v}|_1 .$$

Δ

Proof. We will prove the statement by contradiction. Find the shortest factor u, for which the statement is not satisfied. Then either $|\hat{w}|_1 < |u|_1$ or $|\hat{v}|_1 > |u|_1$.

1. Assume $|\hat{w}|_1 < |u|_1$. By the minimality of $|u|$, $|\hat{w}|_1+1 = |u|_1$. According to the definition of $\mathbf{w}$ and since we have chosen u of minimal length, it holds, with regard to the first statement of Observation 3.17,

$$\hat{w} = 1\cdots 10^{\ell}, \quad \text{where } \ell > p, \quad \text{and} \quad u = 1\cdots 1.$$

It also follows from the first statement of Observation 3.17 that

$$\tilde{w} = 0^p\hat{w}0^{q+p-\ell}1 = 0^p1\cdots 10^q0^p1 \quad \text{and} \quad \tilde{u} = 0^p u$$

are factors of $\mathbf{u}_\beta$, where the block $\hat{w}0^{q+p-\ell}1$ is necessarily a prefix of $\mathbf{w}$. Then the second statement of Observation 3.17 implies that there exist factors $v0$ and u' of $\mathbf{u}_\beta$, the factor u' being shorter than u, such that

$$\tilde{w} = \varphi(v0) \quad \text{and} \quad \tilde{u} = \varphi(u')\,.$$

Furthermore, $v0$ is a prefix of $\mathbf{v}$ by Observation 3.16.

Since $|\tilde{w}|_1 = |\tilde{u}|_1$, it follows that $|v0|_0 = |u'|_0$, hence $|v|_0 < |u'|_0$. Since moreover $0 < |\tilde{w}| - |\tilde{u}| \leq q$, one can deduce that $|v0|$ is greater than $|u'|$ by 1, hence $|v| = |u'|$. That means that we have found a prefix v of $\mathbf{v}$ and a factor u' of $\mathbf{u}_\beta$ such that $|v| = |u'| < |u|$ and $|v|_0 < |u'|_0$, i.e., $|v|_1 > |u'|_1$. This is a contradiction with the assumption that u is the shortest possible.

2. Suppose $|\hat{v}|_1 > |u|_1$. By the minimality of $|u|$, $|\hat{v}|_1 = |u|_1 + 1$. Using the definition of $\mathbf{v}$, the minimality of $|u|$, and the first statement of Observation 3.17, we deduce that

$$\hat{v} = 0^q0^p\cdots 1 \quad \text{and} \quad u = 0^k1\cdots 0^{\ell}\,,$$

where both k and ℓ are greater than p. We apply Observation 3.17 once more to find that

$$\tilde{u} = 0^{q+p-k}u0^j1 = 0^q0^p1\cdots 0^q0^p1 \quad \text{for certain } j,\ 0 \leq j \leq q+p-\ell\,,$$

is a factor of $\mathbf{u}_\beta$, and that there exists a factor $u'0$ of $\mathbf{u}_\beta$ such that $\tilde{u} = \varphi(u'0)$. The second statement of Observation 3.17 together with Observation 3.16 imply that $\hat{v} = \varphi(\hat{w})$, where $\hat{w}$ is a prefix of $\mathbf{w}$, and that $\hat{w}$ is shorter than $\hat{v}$.

Since $|\hat{v}|_1 = |\tilde{u}|_1$, one can see that $|\hat{w}|_0 = |u'0|_0$, hence $|\hat{w}|_0 > |u'|_0$. Since moreover $\tilde{u}$ is longer than $\hat{v}$, the factor $u'0$ is also longer than $\hat{w}$. Let us cut off a suffix of $u'0$ of length $|u'0| - |\hat{w}|$, and denote the resulting factor as u''. Then u'' is a factor of $\mathbf{u}_\beta$ of the same length as $\hat{w}$, and since we have cut off at least one 0, it holds $|\hat{w}|_0 > |u''|_0$, i.e., $|\hat{w}|_1 < |u''|_1$. As the factor u'' is shorter than u (because $|u''| = |\hat{w}| < |\hat{v}| = |u|$), we have reached a contradiction with the minimality of $|u|$.

□

In order to determine $|\hat{v}|_1$ and $|\hat{w}|_1$ for any prefix $\hat{v}$ of $\mathbf{v}$ and $\hat{w}$ of $\mathbf{w}$, we will need several observations. The first observation follows from the definition of $v^{(N)}$ and $w^{(N)}$ and from Observation 3.5.

Observation 3.19. For all $N \in \mathbb{N}$, it holds

$$\left|v^{(N)}\right| = 1 + (q-1)\sum_{j=0}^{N} U_{2j}\,, \qquad \left|v^{(N)}\right|_1 = (q-1)\,(0,1)\sum_{j=0}^{N} M_\varphi^{2j}\begin{pmatrix}1\\0\end{pmatrix},$$
$$\left|w^{(N)}\right| = 1 + (q-1)\sum_{j=0}^{N-1} U_{2j+1}\,, \qquad \left|w^{(N)}\right|_1 = 1 + (q-1)\,(0,1)\sum_{j=0}^{N-1} M_\varphi^{2j+1}\begin{pmatrix}1\\0\end{pmatrix}.$$

Δ

As a consequence of the form of $v^{(N)}$ and $w^{(N)}$, we obtain another observation.

Observation 3.20. Let $\hat{v}$ be a prefix of $\mathbf{v}$ and $\hat{w}$ a prefix of $\mathbf{w}$ of length $n \in \mathbb{N}^+$.

- Find $M \in \mathbb{N}$ such that $|v^{(M)}| \le n < |v^{(M+1)}|$. Then $\hat{v} = v^{(M)}\hat{u}$, where $\hat{u}$ is a prefix of $\mathbf{u}_\beta$.
- Find $N \in \mathbb{N}$ such that $|w^{(N)}| \le n < |w^{(N+1)}|$. Then $\hat{w} = w^{(N)}\hat{u}'$, where $\hat{u}'$ is a prefix of $\mathbf{u}_\beta$.

Δ

The last observation we need to determine $|\hat{v}|_1$ and $|\hat{w}|_1$ follows from Proposition 3.7 and Observation 3.5.

Observation 3.21. Let $\langle n\rangle_U = (d_k, d_{k-1}, \dots, d_1, d_0)$. Then the prefix u of $\mathbf{u}_\beta$ of length n satisfies

$$|u|_1 = (0,1)\left(\sum_{i=0}^{k} d_i M_\varphi^i\right)\begin{pmatrix}1\\0\end{pmatrix}.$$

Δ

Observations 3.19, 3.20, and 3.21 lead to the formulae for $|\hat{v}|_1$ and $|\hat{w}|_1$.

Lemma 3.22. Let $\hat{v}$ be the prefix of $\mathbf{v}$ of length $n \in \mathbb{N}^+$. Find $M \in \mathbb{N}$ such that $|v^{(M)}| \le n < |v^{(M+1)}|$. Then

$$|\hat{v}|_1 = (0,1)\left((q-1)\sum_{i=0}^{M} M_\varphi^{2i} + \sum_{i=0}^{k} d_i M_\varphi^i\right)\begin{pmatrix}1\\0\end{pmatrix}, \tag{3.6}$$

where $\langle n - |v^{(M)}|\rangle_U = \langle n - 1 - (q-1)\sum_{i=0}^{M} U_{2i}\rangle_U = (d_k, d_{k-1}, \dots, d_1, d_0)$. Δ

Lemma 3.23. Let $\hat{w}$ be the prefix of $\mathbf{w}$ of length $n \in \mathbb{N}^+$. Find $N \in \mathbb{N}$ such that $|w^{(N)}| \le n < |w^{(N+1)}|$. Then

$$|\hat{w}|_1 = 1 + (0,1)\left((q-1)\sum_{i=0}^{N-1} M_\varphi^{2i+1} + \sum_{i=0}^{\ell} c_i M_\varphi^i\right)\begin{pmatrix}1\\0\end{pmatrix}, \tag{3.7}$$

where $\langle n - |w^{(N)}|\rangle_U = \langle n - 1 - (q-1)\sum_{i=0}^{N-1} U_{2i+1}\rangle_U = (c_\ell, c_{\ell-1}, \dots, c_1, c_0)$. Δ

At this moment, we could provide a formula for Abelian complexity using Proposition 3.8. However, we prefer to find a more elegant expression for $\mathcal{AC}_{\mathbf{u}_\beta}(n)$. For this purpose we will use the following observation, which can be easily proven by mathematical induction.

Observation 3.24. For all $j \in \mathbb{N}$, it holds

$$U_{2j} \leq |v^{(j)}| < U_{2j+1} \leq |w^{(j+1)}| < U_{2j+2}.$$

Δ

Theorem 3.25. Let $\mathbf{u}_\beta$ be the fixed point of the substitution φ defined in (3.3) with $q > 1$. Let $n \in \mathbb{N}$ and let $J \in \mathbb{N}$ satisfy $U_J \leq n < U_{J+1}$, where $(U_J)_{J=0}^{\infty}$ is the sequence defined in (3.1).

- If J is even: put $N := \frac{J}{2}$ and put $M := \frac{J}{2}$ if $|v^{(\frac{J}{2})}| \leq n$ and $M := \frac{J}{2} - 1$ otherwise.
- If J is odd: put $M := \frac{J-1}{2}$ and put $N := \frac{J+1}{2}$ if $|w^{(\frac{J+1}{2})}| \leq n$ and $N := \frac{J-1}{2}$ otherwise.

Then Abelian complexity $\mathcal{AC}_{\mathbf{u}_\beta}(n)$ of the infinite word $\mathbf{u}_\beta$ is given by the formula

$$\mathcal{AC}_{\mathbf{u}_\beta}(n) = 2+(0,1)\left((q-1)\left[(M_\varphi+I)^{-1}(M_\varphi^{2N}-I)-(M-N+1)M_\varphi^{2N}\right]+\sum_{i=0}^{J}(c_i-d_i)M_\varphi^i\right)\begin{pmatrix}1\\0\end{pmatrix},$$

where

$$\begin{array}{lllll} \langle n-|w^{(N)}|\rangle_U & = & \langle n-1-(q-1)\sum_{i=0}^{N-1}U_{2i+1}\rangle_U & = & (c_J, c_{J-1}, \ldots, c_1, c_0)\,, \\ \langle n-|v^{(M)}|\rangle_U & = & \langle n-1-(q-1)\sum_{i=0}^{M}U_{2i}\rangle_U & = & (d_J, d_{J-1}, \ldots, d_1, d_0)\,, \end{array}$$

and $w^{(j)}, v^{(j)}$ are defined in (3.5). Δ

Proof. Due to Proposition 3.8, it holds $\mathcal{AC}_{\mathbf{u}_\beta}(n) = 1 + |\hat{w}|_1 - |\hat{v}|_1$, where $\hat{v}$, $\hat{w}$ are the prefixes of $\mathbf{v}$ and $\mathbf{w}$, respectively, of length n. By Lemma 3.22, we have

$$|\hat{v}|_1 = (0,1)\left((q-1)\sum_{i=0}^{M}M_\varphi^{2i}+\sum_{i=0}^{k}d_iM_\varphi^i\right)\begin{pmatrix}1\\0\end{pmatrix}, \tag{3.8}$$

where M is given by $|v^{(M)}| \leq n < |v^{(M+1)}|$ and we can put $k := J$ because $n - |v^{(M)}| < n < U_{J+1}$. Similarly, Lemma 3.23 gives

$$|\hat{w}|_1 = 1+(0,1)\left((q-1)\sum_{i=0}^{N-1}M_\varphi^{2i+1}+\sum_{i=0}^{\ell}c_iM_\varphi^i\right)\begin{pmatrix}1\\0\end{pmatrix},$$

where N is determined by $|w^{(N)}| \leq n < |w^{(N+1)}|$ and we can set $\ell := J$ because $n - |w^{(N)}| < n < U_{J+1}$.

Now, using the assumption $U_J \leq n < U_{J+1}$ together with Observation 3.24, we find:

- If J is even, then $N = \frac{J}{2}$, and $M = \frac{J}{2} - 1$ or $M = \frac{J}{2}$;
- if J is odd, then $M = \frac{J-1}{2}$, and $N = \frac{J+1}{2}$ or $N = \frac{J-1}{2}$.

Note that it always holds $M = N - 1$ or $M = N$, which allows us to rewrite the first sum in (3.8) using the equality

$$\sum_{i=0}^{M}M_\varphi^{2i} = \sum_{i=0}^{N-1}M_\varphi^{2i}+(M-N+1)M_\varphi^{2N}.$$

Now, it suffices to substitute the expressions for $|\hat{v}|_1$ and $|\hat{w}|_1$ into (3.3) and after a simple manipulation (note that $\sum_{i=0}^{N-1}M_\varphi^{2i+1} - \sum_{i=0}^{N-1}M_\varphi^{2i} = (M_\varphi+I)^{-1}(M_\varphi^{2N}-I)$), we obtain the sought formula for $\mathcal{AC}_{\mathbf{u}_\beta}(n)$. $\square$

Example 3.26. *Let $p = 3$ and $q = 2$. Let us calculate $\mathcal{AC}_{\mathbf{u}_\beta}(n)$ for $n = 7$. Using (3.1), we obtain $(U_n)_{n=0}^{\infty} = (1, 4, 14, \ldots)$. We can see that $U_1 = 4 \leq 7 < U_2 = 14$, i.e., $J = 1$ is odd in Theorem 3.25 and consequently, we have $M = \frac{J-1}{2} = 0$. Since $|w^{(\frac{J+1}{2})}| = |w^{(1)}| = 5 \leq 7$, we get $N = \frac{J+1}{2} = 1$. Now, we have to determine the normal U-representations:*

$$\langle 7 - |w^{(1)}| \rangle_U = \langle 2 \rangle_U = (0, 2) = (c_1, c_0) \quad \text{and} \quad \langle 7 - |v^{(0)}| \rangle_U = \langle 5 \rangle_U = (1, 1) = (d_1, d_0).$$

Finally, using Theorem 3.25, we can calculate $\mathcal{AC}_{\mathbf{u}_\beta}(7) = 2 + (0, 1)\left((M_\varphi - I) + (I - M_\varphi)\right)\begin{pmatrix} 1 \\ 0 \end{pmatrix} = 2$. *Let us check the obtained value of $\mathcal{AC}_{\mathbf{u}_\beta}(7)$ by determining $\hat{w}$ and $\hat{v}$, the prefixes of length 7 of $\mathbf{w}$ and $\mathbf{v}$, respectively. We have $\hat{w} = 1000100$ and $\hat{v} = 0000010$, consequently indeed $\mathcal{AC}_{\mathbf{u}_\beta}(7) = |\hat{w}|_1 - |\hat{v}|_1 + 1 = 2$.* Δ

CHAPTER 4

Balance properties and Abelian complexity of words over multiliteral alphabets

Abelian complexity of words over multiliteral alphabets is still not very deeply investigated. Words for which the Abelian complexity has been partially described are the Tribonacci word and a class of words associated with cubic non-simple Parry numbers. In this chapter, we summarize known results and open questions.

4.1 d-bonacci word

The d-bonacci substitution defined in Table 1.2 is a generalization of the Fibonacci substitution. Despite the relative simplicity of the substitution, the possible values of the Abelian complexity of its fixed point have been described only for the Tribonacci word in [20], in the case of $\mathbf{t}$ being the fixed point of $\varphi : 0 \to 01, 1 \to 02, 2 \to 0$.

Let us recall the most important results concerning the Tribonacci word.

Theorem 4.1. The Tribonacci word $\mathbf{t}$ is 2-balanced. Δ

Theorem 4.2. For the Tribonacci word $\mathbf{t}$, it holds $\mathcal{AC}_{\mathbf{t}}(n) \in \{3, 4, 5, 6, 7\}$. Moreover, each of these values is attained. Δ

Proposition 4.3. Let $\mathbf{t}$ be the Tribonacci word. Then $\mathcal{AC}_{\mathbf{t}}(n) = 3$ if and only if $n = 1$ or $n = \frac{1}{2}(T_m + T_{m+2} - 1)$ for some non-negative m, where $(T_i)_{i=0}^{+\infty} = 1, 2, 4, 7, \ldots$ is the sequence of Tribonacci numbers given by $T_0 = 1, T_1 = 2, T_2 = 4, T_{i+3} = T_i + T_{i+1} + T_{i+2}$. Δ

Proposition 4.4. For the Tribonacci word $\mathbf{t}$, $\mathcal{AC}_{\mathbf{t}}(n) = 7$ is attained for infinitely many n. Δ

Let us mention some open problems concerning the d-bonacci words:

i) Is $\mathcal{AC}_{\mathbf{t}}(n) = c$ attained for infinitely many n for some $c \in \{4, 5, 6\}$?

ii) Characterize all n for which $\mathcal{AC}_{\mathbf{t}}(n) = c$, where $c \in \{4, 5, 6, 7\}$.

iii) Is the d-bonacci word $(d-1)$-balanced? (Our numerical simulations support a positive answer.)

iv) Find at least some partial description of $\mathcal{AC}(n)$ for the d-bonacci word with $d > 3$.

4.2 Words associated with cubic non-simple Parry numbers

Possible values of Abelian complexity of a whole class of multiliteral infinite words have been described in [23]. The author deals with fixed points of ternary non-simple Parry substitutions (see Table 1.2) having affine factor complexity. These words are generated by the substitution

$$\begin{aligned} 0 &\to 0^r 1, \\ 1 &\to 2, \\ 2 &\to 0^{r-1} 1, \end{aligned} \tag{4.1}$$

where $r > 1$.

Theorem 4.5. For all $r > 1$, the fixed point of the substitution (4.1) is 3-balanced and this bound is optimal. Δ

Theorem 4.6. Let $\mathbf{u}$ be the fixed point of the substitution (4.1). Then the Abelian complexity satisfies $3 \leq \mathcal{AC}_{\mathbf{u}}(n) \leq 7$ for all $n \in \mathbb{N}^+$. Both bounds of $\mathcal{AC}_{\mathbf{u}}(n)$ are optimal. Moreover, for all $r > 1$, $\mathcal{AC}_{\mathbf{u}}(n) = 7$ is attained infinitely many times. Δ

Remark 4.7. Our numerical simulations show that for each value $c \in \{4, 5, 6\}$, there is an $n \in \mathbb{N}^+$ such that $\mathcal{AC}_{\mathbf{u}}(n) = c$.

4.3 Words with (eventually) constant Abelian complexity

Another task concerning the Abelian complexity is to find words with $\mathcal{AC}(n) = d$ for all $n \in \mathbb{N}^+$ (we use d because $\mathcal{AC}(1) = \#\mathcal{A} = d$). With no additional requirements, the task seems to be quite simple.

Observation 4.8. Let us consider the alphabet $\mathcal{A} = \{0, 1, \ldots, d-1\}$. Let us construct a word $\mathbf{u}$ in this way:

$$\mathbf{u} = (d-1)(d-2)\cdots 10^\omega.$$

Then the Abelian complexity is constantly $\mathcal{AC}_{\mathbf{u}}(n) = d$ for all $n \in \mathbb{N}^+$. Δ

It is natural to add some restrictions – from now on, let us require the recurrence of the infinite words. As eventually periodic words with a pre-period of non-zero length are not recurrent, the construction from Observation 4.8 is excluded.

Let us discuss the case of binary alphabets at first and seek for recurrent words with $\mathcal{AC}(n) = 2$ for all $n \in \mathbb{N}^+$. With respect to Theorem 3.1 and Proposition 1.16, Sturmian words are the only recurrent binary words satisfying our conditions.

The case of a ternary alphabet has been described in [19]. There are described two methods of construction of such words.

Theorem 4.9. Let $\mathbf{u}$ be an aperiodic balanced word over the alphabet $\{0, 1, 2\}$. Then $\mathcal{AC}_{\mathbf{u}}(n) = 3$ for all $n \in \mathbb{N}^+$. Δ

Theorem 4.10. Let $\mathbf{u}$ be an aperiodic infinite word over the alphabet $\mathcal{A} = \{0, 1\}$, let φ be the morphism $0 \to 012, 1 \to 021$. Then for the Abelian complexity, it holds $\mathcal{AC}_{\varphi(\mathbf{u})}(n) = 3$ for all $n \in \mathbb{N}^+$. Δ

It has been shown in the paper [9] that there are no such words over alphabets of higher cardinalities.

Theorem 4.11. Let $d \geq 4$ be an integer. There is no recurrent word $\mathbf{u}$ with the Abelian complexity $\mathcal{AC}_{\mathbf{u}}(n) = d$ for all $n \in \mathbb{N}^+$. Δ

Let us make our requirements even weaker. Instead of recurrent words with constant Abelian complexity, let us demand recurrent words with eventually constant Abelian complexity. The following theorem, which can be found in [21], gives us a procedure how to construct such words.

Theorem 4.12. Let $\mathbf{u}$ be a Sturmian word over the alphabet $\mathcal{A} = \{0, 1\}$. For a given $l \geq 1$, let us consider the morphism $\varphi : 0 \to 0^l, 1 \to 1^l$. Then $\mathcal{AC}_{\varphi(\mathbf{u})}(n) = l + 1$ for all $n \geq l$. Δ

Example 4.13. *Let us find a recurrent word with the Abelian complexity eventually equal to 4. Using Theorem 4.12, we set $l = 3$ and we apply substitution $\varphi : 0 \to 000, 1 \to 111$ on the Fibonacci word $\mathbf{f}$, which is Sturmian.*

$$\begin{array}{rcccccccccc} \mathbf{f} & = & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \cdots \\ \varphi(\mathbf{f}) & = & 000 & 111 & 000 & 000 & 111 & 000 & 111 & 000 & \cdots \end{array}$$

Δ

CHAPTER 5

TigrWords

As mentioned in Chapter 4, Abelian complexity of words over multiliteral alphabets is still not deeply investigated. Hence, in order to be able to describe the Abelian complexity of these words, we have created a toolkit TigrWords (to be found on the enclosed CD), which facilitates study of infinite words approximated by their prefixes and may serve to discover new hypotheses. We describe some algorithms used in TigrWords and individual layers of TigrWords which correspond to the used programming languages (C++and Python).

5.1 Algorithms used in TigrWords

5.1.1 Iterating through prefixes of words generated by a substitution

The task of iterating over prefixes of a fixed point of a given substitution seems to be essential and we must pay enough attention to both the speed and space complexity.

The easiest approach is to generate the whole desired prefix (which is kept in memory) and then to iterate through it. Nevertheless, this method is useful only for short prefixes since the space complexity is $\mathcal{O}(n)$, i.e, there exists a constant M such that we use less than Mn units of memory to iterate through the prefix of length n.

Another approach can be characterized as "expansion from left" and we demonstrate it on a simple example since the algorithm is readable from it.

Example 5.1. *Let us consider the Fibonacci substitution* $\varphi : 0 \to 01, 1 \to 0$ *and let us iterate through the word* $\varphi^3(0)$*. We rewrite it in the following way:*

1) $\varphi^3(0)$*;*

2) $\varphi^2(01)$*;*

3) $\varphi(01)\varphi^2(1)$*;*

4) $01\varphi(1)\varphi^2(1)$*;*

5) $010\varphi^2(1)$*;*

6) $010\varphi(0)$*;*

7) 01001*;*

where the colors in every step mean:

- *black – letter still not processed*
- *red – letters processed in this step*
- *green – letters processed in some of the previous steps*

Δ

First, we choose a "large enough" k, i.e., such that the prefix through which we want to iterate is itself a prefix of $\varphi^k(0)$, and then we iterate through $\varphi^k(0)$.

We work with expressions of the form

$$f_0\varphi(f_1)\cdots\varphi^k(f_k), \tag{5.1}$$

where $|f_i| \leq \max_{a\in\mathcal{A}}|\varphi(a)|$ for all $i \in \{0,1,\ldots,k\}$. In every step, we expand the most left subexpression until $f_0 \neq \varepsilon$. Then we process f_0 and continue with the expression without f_0 in the same way.

A stack seems to be the best data structure to store the expressions 5.1. The space complexity of this method is $\mathcal{O}(\log n)$. Its analysis and a generalization can be found in [17].

5.1.2 Abelian complexity computation

We also need to know through how long prefix we must iterate to reach all factors of a given length. The answer given by the following theorem is a consequence of the "window lemma" proven by David Damanik and Douglas Zare in [10].

Theorem 5.2. Let $\mathbf{u}$ be a fixed point of a primitive substitution φ. Then $\mathbf{u}$ is linearly recurrent, i.e., there exists a constant C such that every factor of $\mathbf{u}$ of length Cn contains all factors of $\mathbf{u}$ of length n. Such constant C can be found as

$$C = \frac{(C_2+2)z_2\Lambda}{z_1},$$

where

- Λ is the dominant eigenvalue of the incidence matrix of the substitution φ;
- z_1, z_2 are positive constants such that for all $K \in \mathbb{N}^+$, it holds

$$z_1\Lambda^K \leq \min_{a\in\mathcal{A}}|\varphi^K(a)| \leq \max_{a\in\mathcal{A}}|\varphi^K(a)| \leq z_2\Lambda^K;$$

- C_2 is a constant such that $\mathbf{u}_{[0,C_2)}$ contains all factors of $\mathbf{u}$ of length 2.

Δ

Hence, our method of computation of $\mathcal{AC}(n)$ of the fixed point $\mathbf{u} = \lim_{m\to+\infty}\varphi^m(a)$ of a primitive substitution φ can be structured into the following steps:

Algorithm 2 Computation of $\mathcal{AC}(n)$

Find the constant C by Theorem 5.2.
Find the constant k such that $\varphi^k(a) \geq Cn$.
Find the first Parikh vector: $\psi = \Psi(\mathbf{u}_{[0,n)})$.
Add ψ to the set of Parikh vectors
for $i = 1 \to Cn - n$ **do**
  $\psi_{\mathbf{u}_{i-1}} \leftarrow \psi_{\mathbf{u}_{i-1}} - 1$
  $\psi_{\mathbf{u}_{i+n-1}} \leftarrow \psi_{\mathbf{u}_{i+n}} + 1$
  Add ψ to the set of Parikh vectors
  $i \leftarrow i + 1$
end for
Output: The set of Parikh vectors

5.2 C++layer

The C++layer is used for iterating through prefixes of infinite words, which takes a lot of CPU time and much optimization is necessary to reach sufficient speed and to make the memory requirements as effective as possible.

Our programs are located in Microsoft Visual C++ 2010 project *TigrWords 1.0*. Nevertheless, it uses only standard features of C++, so it is compilable on other platforms and with other compilers such GCC, too. The STL (Standard Template Library) is used for output operations and for data structures as stacks, lists, and vectors.

Let us emphasize a particularity concerning C++codes – in our C++sources, we have to write all letters increased by 1 since the C++language has the character $\backslash 0$ reserved as a terminal string character. With a respect to the used data type, the cardinality of the alphabet can be at most 255.

5.2.1 List of files

- *datatypes.h* – data types used in other files
- *main.cpp* – pattern for creation of a new program for some other substitution
- *Substitution.cpp* – class *Substitution*
- *Substitution.h* – its header file
- *Word.cpp* – class *Word*
- *Word.h* – its header file

5.2.2 Classes and data types

1. Data types
   - *letter_type* – individual letter, derived from *unsigned char*
   - *word_type* – sequence of letters, derived from *std::string*
   - *pv_type* – individual Parikh vector, derived from *std::vector* of *long*s

- *pvset_type* – set of Parikh vectors, derived from *std::set* of *pv_type*s

2. Class *Substitution*
   - *Substitution(void)* – constructor
   - *void addRule(letter_type source, word_type image)* – adds a new substitution rule (source → image) to the internal list of rules
     - *source* – source letter
     - *image* – image of the source letter
   - *word_type applySubstitution(word_type *source)* – applies the substitution on *source*
     - *source* – source word
     - returns image of the source word
   - *void print()* – prints out information about the substitution
   - *unsigned int getAlphabetCardinality()*
     - returns the cardinality of an alphabet obtained from the substitution rules
3. Class Word
   - *Word(Substitution substitution, std::string initialWord, unsigned int levels)* – constructor
     - *substitution* – substitution which will be used to generate the word
     - *initialWord* – initial word on which the substitution will be applied
     - *levels* – k into 5.1
   - *Word(Substitution substitution, std::string initialWord)* – constructor
     - *substitution* – substitution which will be used to generate the word
     - *initialWord* – initial word on which the substitution will be applied
   - *void printStack()* – prints internal states of the stack
   - *letter_type getNextLetter()* – returns next letter of a prefix of the word

5.2.3 Examples

First, we have to create an instance of the class *Substitution* and to set the rules as shown in Listing 5.1.

Listing 5.1: The 5-bonacci substitution

```
/* variable sub for class Substitution */
Substitution sub;

/* adding rules */
sub.addRule('\1', "\1\2");     // 0 -> 01
sub.addRule('\2', "\1\3");     // 1 -> 02
sub.addRule('\3', "\1\4");     // 2 -> 03
sub.addRule('\4', "\1\5");     // 3 -> 04
sub.addRule('\5', "\1");       // 4 -> 0
```

```
/* printing out information about the substitution */
sub.print();
```

Now, we can generate a word using the class *Word*.

Listing 5.2: A prefix of the 5-bonacci word

```
/* variable wrd for class Word */
Word wrd = Word(sub,"\1");     // inital letter 0

/* print out first 10 letters of the 5-bonacci*/
for(int i=0;i<10;i++) {
  printLetter (wrd.getNextLetter());
}
```

Now, let us demonstrate a complex program. For a fixed point $\mathbf{u}$ of a primitive substitution, it generates Parikh vectors of factors from $\mathcal{L}_n(\mathbf{u})$. It takes 2 parameters – n (which determines $\mathcal{L}_n(\mathbf{u})$) and max_n ($\mathbf{u}_{[0,max\_n)}$ is the prefix which is iterated through).

Listing 5.3: Complete example – listing of Parikh vectors of the 5-bonacci word

```
1  #include <cstdlib>
2  #include <iostream>
3  #include <string>
4  #include <map>
5  #include <new>
6  #include <stack>
7  #include <vector>
8  #include "Substitution.h"
9  #include "Word.h"
10
11 #define COMMENT_PREFIX "#␣"
12
13 int main(int argc, char *argv[]) {
14   /* check of the count of arguments */
15   if (argc == 3)
16   {
17     /* variable sub for class Substitution */
18     Substitution sub;
19
20     /* adding rules */
21     sub.addRule('\1', "\1\2");     // 0 -> 01
22     sub.addRule('\2', "\1\3");     // 1 -> 02
23     sub.addRule('\3', "\1\4");     // 2 -> 03
24     sub.addRule('\4', "\1\5");     // 3 -> 04
25     sub.addRule('\5', "\1");       // 4 -> 0
26
27     /* initial letter */
28     word_type initialWord = "\1";
29
30     /* arguments from the commandline */
31     unsigned long n = atoi(argv[1]);
```

```
32      unsigned long max_n = atoi(argv[2]);
33
34      /* variable for a set of Parikh vectors, a variable for the current↩
            Parikh vector  */
35      pvset_type pvektory;
36      pv_type current_pv(sub.getAlphabetCardinality()+1,0);
37
38      /* variables for two words (for iterating over the same word at ↩
           different positions) */
39      Word wrd1 = Word(sub,initialWord);
40      Word wrd2 = Word(sub,initialWord);
41
42      /* adding the initial Parikh vector */
43      for (unsigned i=0;i<n;i++) {
44        letter_type current_letter = wrd1.getNextLetter();
45        current_pv[current_letter]+=1;
46      }
47
48      pvektory.insert(current_pv);
49
50      /* adding the other Parikh vectors */
51      for(unsigned i=0;i<max_n-n;i++) {
52        letter_type current_letter1 = wrd1.getNextLetter();
53        letter_type current_letter2 = wrd2.getNextLetter();
54        current_pv[current_letter1]+=1;
55        current_pv[current_letter2]-=1;
56        pvektory.insert(current_pv);
57      }
58
59      /* listing of the found Parikh vectors */
60      pv_type *pvvv = new pv_type;
61      pvset_type::iterator it;
62
63      for ( it=pvektory.begin() ; it != pvektory.end(); it++ ) {
64        *pvvv = (*it);
65        for (unsigned i=0;i<sub.getAlphabetCardinality();i++){
66          std::cout << "␣" << (*pvvv)[i+1];
67        }
68        std::cout << std::endl;
69      }
70
71        }
72
73    /* wrong count of arguments */
74    else
75    {
76      std::cout << COMMENT_PREFIX << "Bad␣count␣of␣parameters␣(should␣be␣↩
            2):" << std::endl;
77      std::cout << COMMENT_PREFIX << "␣␣␣parameter␣#1:␣␣n␣for␣AC(n)" << ↩
            std::endl;
78      std::cout << COMMENT_PREFIX << "␣␣␣parameter␣#2:␣␣length␣of␣prefix␣↩
            to␣iterate␣through" << std::endl;
79      std::cout << COMMENT_PREFIX << std::endl;
```

```
80     }
81 }
```

5.3 Python layer

The programming language Python 3 is used because of speed of implementation of new algorithms. However, it is slower than compiled languages[1], e.g. C++. Therefore, we can perform some of low level operations by external programs written in C++.

5.3.1 List of files

- *libwords.py* – generic class for a word, class for a word obtained by a substitution and classes for specific substitutions: binary non-Pisot, Chacon (4 kinds), d-bonacci, non-simple Parry, period doubling, Rudin-Shapiro, simple Parry, Thue-Morse (3 kinds), generalized Thue-Morse
- *libwords_test.py* – unit tests for *libwords.py*
- *libmatrix.py* – class for matrices
- *libmatrix_test.py* – unit tests for *libmatrix.py*

5.3.2 Classes

1. Class *Word*
   - *__init__(self, prefix=(0,))* – constructor
     - *prefix* – prefix which approximates the word represented by this class
   - *getAbelianComplexity(self, n, enoughPrefixLength=0)* – returns the Abelian complexity and the Parikh vectors
     - *n* – length of factors for $\mathcal{AC}(n)$ (0 to use the whole already generated prefix)
     - *enoughPrefixLength* – length of a prefix in which all factors of length n of the infinite word are located
     - returns ($\mathcal{AC}(n)$, tuple of Parikh vectors)
   - *getAbelianComplexityC(self, n, enoughPrefixLength,program="AC")* – returns the Abelian complexity and the Parikh vectors (calls external program)
     - *n* – length of factors for $\mathcal{AC}(n)$
     - *enoughPrefixLength* – length of a prefix in which all factors of length n of the infinite word are located
     - *program* – name of the program to perform the computation
     - returns ($\mathcal{AC}(n)$, tuple of Parikh vectors)
2. Class *WordSubstitution* (descendant of *Word*)

[1] In fact, theoretically, it is possible to make Python programs as quick as C++programs or in some cases even faster, nevertheless, it is necessary to use some extensions and improvements like Cython.

- *__init__(self, substitution, prefix=[0])* – constructor, sets a substitution which will be used to generate prefixes
  - *substitution* – substitution which generates the word
  - *prefix* – initial word on which is applied the substitution
- *setSubstitution(self, substitution)* – sets a new substitution of the word
  - *substitution* – substitution which generates the word
- *useSubstitution(self, word)* – uses the substitution on the word *word*
  - *word* – word on which the substitution should be applied
  - returns $\varphi(word)$
- *generatePrefix(self, length=0)* – generates a prefix
  - *length* – desired length of the generated prefix

3. Class *WordBinaryNonPisot* (descendant of *SubstitutionWord*) – class for a specific binary non-Pisot word generated by the substitution $0 \to 01, 1 \to 000$
4. Class *WordChacon* (descendant of *SubstitutionWord*) – class for the Chacon word (variation number 1) generated by the substitution $0 \to 0012, 1 \to 12, 2 \to 012$
5. Class *WordChacon2* (descendant of *SubstitutionWord*) – class for the Chacon word (variation number 2) generated by the substitution $0 \to 0^q1, 1 \to 10^{q-1}1$
   - *__init__(self, q)* – constructor
6. Class *WordChacon3* (descendant of *SubstitutionWord*) – class for the Chacon word (variation number 3) generated by the substitution $0 \to 01, 1 \to 2^{r-1}012^{s-1}, 2 \to 2^r012^{s-1}$
   - *__init__(self, r, s)* – constructor
7. Class *WordChacon4* (descendant of *SubstitutionWord*) – class for the Chacon word (variation number 4) generated by the substitution $0 \to 0010, 1 \to 1$
8. Class *WordDBonacci* (descendant of *SubstitutionWord*) – class for the d-bonacci word as presented in Table 1.2
   - *__init__(self, d)* – constructor
9. Class *WordPeriodDoubling* (descendant of *SubstitutionWord*) – class for the period doubling word as presented in Table 1.1
10. Class *WordRudinShapiro* (descendant of *SubstitutionWord*) – class for the Rudin-Shapiro word as presented in Table 1.2
11. Class *WordThueMorse* (descendant of *SubstitutionWord*) – class for the Thue-Morse word (variation 1) as presented in Table 1.1
12. Class *WordThueMorse2* (descendant of *SubstitutionWord*) – class for the Thue-Morse word (variation 2) generated by the substitution $0 \to 12, 1 \to 102, 2 \to 0$
13. Class *WordThueMorse3* (descendant of *SubstitutionWord*) – class for the Thue-Morse word (variation 3) generated by the substitution $0 \to 12, 1 \to 13, 2 \to 20, 3 \to 21$

14. Class *WordGeneralizedThueMorse* (descendant of *SubstitutionWord*) – class for the generalized Thue-Morse word as presented in Table 1.2

    - *__init__(self,b,d)* – constructor

Now, let us demonstrate the computation of the Abelian complexity of the Tribonacci word.

Listing 5.4: The Abelian complexity of the Tribonacci word

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
import libwords

# we seek the AC(n) for n from {1,2,..,100}
ACN_MAX_N = 100

# let us suppose that k=100
k = 10

# word
wrd = libwords.WordDBonacci(3)

# Parikh vectors will be stored in this dictionary
parikh = {}

# set of all attained values of AC(n)
abelianComplexityFunctionValues = set()

for n in range(1, ACN_MAX_N + 1):
  wrd.generatePrefix(length=k*n)

  print(n, end=" ")
  acn, parikh[n] = wrd.getAbelianComplexity(n)

  abelianComplexityFunctionValues.add(acn)


for key in parikh:
  print ("AC(", key, ") = ", len (parikh[key]), "   ...Parikh vectors: ",
      [str(x) for x in parikh[key]], sep="")

print ("Values of the Abelian complexity:",
    abelianComplexityFunctionValues)
```

CHAPTER 6

Summary

6.1 Results

In this text, we have summarized definitions and known results concerning the Abelian complexity of infinite words. Our original results are

i) derivation of an explicit formula of the Abelian complexity for the case of words associated with quadratic simple and non-simple Parry numbers;

ii) creation of the toolkit TigrWords, which facilitates study of infinite words approximated by their prefixes.

6.2 Further work

In future, we want to extend the toolkit TigrWords (to add new functions and a Wolfram Mathematica layer for symbolic computations). Then we would like to describe the Abelian complexity of the d-bonacci word in more details, to prove or disprove its $(d-1)$-balance property at least in some special cases and for the Tribonacci word $\mathbf{t}$, to use TigrWords to characterize those n for which $\mathcal{AC}_{\mathbf{t}}(n) = c$ for some $c \in \{4, 5, 6\}$.

Bibliography

[1] B. Adamczewski. Balances for fixed points of primitive substitutions. *Theoretical Computer Science*, 307(1):47 – 75, 2003. 19, 20

[2] B. Adamczewski. Symbolic discrepancy and self-similar dynamics. *ANNALES DE L INSTITUT FOURIER*, 54(7):2201+, 2004. 19

[3] Ľ. Balková. Nahlédnutí pod pokličku kombinatoriky na nekonečných slovech. *Pokroky matematiky, fyziky a astronomie*, 2011. 8

[4] Ľ. Balková, K. Břinda, and O. Turek. Abelian complexity of infinite words associated with quadratic parry numbers. *Theoretical Computer Science*, (0):–, 2011. 8, 27

[5] Ľ. Balková, E. Pelantová, and t. Starosta. Sturmian jungle (or garden?) on multiliteral alphabets. *RAIRO - Theoretical Informatics and Applications*, 44(04):443–470, 2010. 16, 26

[6] Ľ. Balková, E. Pelantová, and O. Turek. Combinatorial and arithmetical properties of infinite words associated with non-simple quadratic parry numbers. *RAIRO - Theoretical Informatics and Applications*, 41(03):307–328, 2007. 27, 28, 30

[7] J. Cassaigne, G. Richomme, K. Saari, and L. Q. Zamboni. Avoiding Abelian powers in binary words with bounded Abelian complexity. *ArXiv e-prints*, May 2010.

[8] E. M. Coven and G. A. Hedlund. Sequences with minimal block growth. *Theory of Computing Systems*, 7:138–153, 1973. 10.1007/BF01762232. 8, 26

[9] J. Currie and N. Rampersad. Recurrent words with constant Abelian complexity. *ArXiv e-prints*, Nov. 2009. 40

[10] D. Damanik and D. Zare. Palindrome complexity bounds for primitive substitution sequences. *Discrete Mathematics*, 222(1):259–268, 2000. 42

[11] S. Fabre. Substitutions et [beta]-systèmes de numération. *Theoretical Computer Science*, 137(2):219 – 236, 1995. 27, 29

[12] M. Fiedler. *Speciální matice a jejich použití v numerické matematice*. SNTL, 1981. 13

[13] C. Frougny, J.-P. Gazeau, and R. Krejcar. Additive and multiplicative properties of point sets based on beta-integers. *Theor. Comput. Sci.*, 303:491–516, July 2003. 28

[14] C. Frougny, Z. Masáková, and E. Pelantová. Infinite special branches in words associated with beta-expansions. *J. Automata, Languages and Combinatorics*, 2005. 28

[15] D. Knuth. *The Art of Computer Programming: Fundamental algorithms*. The Art of Computer Programming. Addison-Wesley, 2008.

[16] M. Lothaire. Algebraic Combinatorics on Words, volume 90 of Encyclopedia of Mathematics and its Applications, 2002. 17

[17] J. Patera. Generating the fibonacci chain in o (logn) space and o (n) time. *PHYSICS OF PARTICLES AND NUCLEI C/C OF FIZIKA ELEMENTARNYKH CHASTITS I ATOMNOGO IADRA*, 33(SUPP/1):118–122, 2002. 42

[18] M. Queffélec. Substitution dynamical systems—spectral analysis (lecture notes in mathematics vol 1294), 1987. 13

[19] G. Richomme, K. Saari, and L. Q. Zamboni. Abelian complexity of minimal subshifts. *Journal of the London Mathematical Society*, 2010. 8, 27, 39

[20] G. Richomme, K. Saari, and L. Q. Zamboni. Balance and abelian complexity of the tribonacci word. *Advances in Applied Mathematics*, 45(2):212 – 231, 2010. 38

[21] A. Saarela. Ultimately constant abelian complexity of infinite words. *J. Autom. Lang. Comb.*, 14:255–258, June 2009. 40

[22] O. Turek. Balance properties of the fixed point of the substitution associated to quadratic simple pisot numbers. *RAIRO - Theoretical Informatics and Applications*, 41(02):123–135, 2007. 28

[23] O. Turek. Balances and abelian complexity of a certain class of infinite ternary words. *RAIRO - Theoretical Informatics and Applications*, 44(03):313–337, 2010. 39

Index