Szilárd Zsolt Fazekas

REPETITIVE SUBWORDS

PhD Dissertation

Supervised by Masami Ito

Department of Romance Studies

UNIVERSITAT ROVIRA I VIRGILI

· Tarragona, 2010 ·

Supervisor

Professor **Masami Ito**
Institute of Computer Sciences
Faculty of Science
Kyoto Sangyo University
603-8555 Kyoto-City
Japan

Tutor

Dr. Gemma Bel Enguix
Research Group on Mathematical Linguistics
Rovira i Virgili University
Avinguda Catalunya, 35
43002 Tarragona
Spain

Abstract

The central notion of this thesis is repetitions in words. We study problems related to contiguous repetitions. More specifically we will consider repeating scattered subwords of non-primitive words, i.e. words which are complete repetitions of other words. We will present inequalities concerning these occurrences as well as giving a partial solution to an open problem posed by Salomaa et al. We will characterize languages, which are closed under the operation of duplication, that is repeating any factor of a word. We also give new bounds on the number of occurrences of certain types of repetitions of words. We give a solution to an open problem posed by Calbrix and Nivat concerning regular languages consisting of non-primitive words. We also present some results regarding the duplication closure of languages, among which a new proof to a problem of Bovet and Varricchio.

... there was no telling what people might find out once they felt free to ask whatever questions they wanted to.

Joseph Heller

Acknowledgments

This thesis is the result of four years of work in the Research Group in Mathematical Linguistics at the Department of Romance Filology of Rovira i Virgili University. Many people have contributed, one way or another, to this effort and I would like to express my gratitude to all of them.
First of all, I am most grateful to Professor Masami Ito for granting me the honour of doing a PhD under his supervision. His professional guidance as well as his help during research visits and his friendly attitude towards me meant very much to me. I had the opportunity to learn a lot from him not only about science but also about Japanese culture. The time spent in Kyoto visiting him influenced me beyond the limits of my chosen topic of research.
Special thanks are due to Professor Carlos Martín-Vide for accepting me into the 3rd International PhD School in Formal Languages and Applications and for all the hard work he has done to provide a wonderful setting for study and research. He made all this possible by building a group where it was a pleasure to work and by helping to obtain generous funding for the time of my studies.
It is my pleasure to thank Professor Pál Dömösi for his guidance and advices he offered me ever since I first started working under his supervision towards my master's thesis. I greatly appreciate the efforts he made to help me anytime I asked.
I would like to thank Professor Sheng Yu and Professor Arto Salomaa for their support and suggestions during my visit to London, Ontario.
I have learned a lot from Professor Maxime Crochemore and Professor Costas Iliopoulos and I would like to thank both for the possibility to work with them.
I would also like to thank Tamás Gaál for giving me the opportunity to take a peek into industry related research. I very much enjoyed the time spent at Xerox Research Center Europe and many thanks go to him for it.
I am grateful to Professor Zoltán Ésik and Professor Victor Mitrana for their advice on scientific matters.
I would like to say thanks to my coauthors with whom it was always exciting putting the results together and who have a great merit in the realisation of this thesis. I would like to thank all my friends and colleagues from the Research Group in Mathematical Linguistics from Rovira i Virgili both for the interesting scientific discussions and for turning the last few years spent in Tarragona into a fantastic life experience.
Finally, I would like to thank the programme Formación de Profesorado Universitario run by the Spanish Ministry of Education for the funding they made available to me during my PhD.

Contents

1 Introduction **3**

2 Background **9**

2.1 Combinatorics on words 9

2.2 Formal languages and automata theory 18

3 Bounds on powers **25**

3.1 A bound on repeats with exponent e, with $1 < e < 2$ 27

3.2 A bound on primitively rooted cubes 30

3.3 Conclusion 38

4 Counting scattered subwords **39**

4.1 Parikh matrices and subword histories 40

4.2 Primitivity in the Context of Multisets 42

4.3 Counting scattered subwords 45

4.4 Subwords of the Thue-Morse word 51

4.5 Inferring primitivity from partial information 54

4.6 Subword inequalities 56

5 Powers of regular languages **65**

5.1 Roots and Powers of Regular Languages 66

5.2 The power of a regular language 69

5.3 Conclusion 75

6 Duplication closure of binary languages **77**

Bibliography **85**

List of Figures

2.1 Finite automaton accepting $(0^*10^*10^*)^*$ 21

2.2 Some closure properties of Chomsky language classes. 22

3.1 Structure of word, w 28

3.2 Repetitions of exponent 1.5 in a^9ba^{17} 29

3.3 Cubes of word a^{4k+3} 30

3.4 Cubes of word $a^kba^kba^kba^k$ 31

3.5 Cubes u^3 and v^3 beginning at the same position i. 37

6.1 Automaton accepting $(a^3b^2a^4c^3)^{\heartsuit\leq 2}$ 79

6.2 Automaton accepting $(abc)^{\heartsuit\leq 3}$ 80

Chapter 1

Introduction

The concept of repetition plays a major role in natural sciences, in general, and in mathematics and computer science, in particular. This is due to the fact that in most of the cases repetition is what makes a structure easier to describe and analyse. From repeating genes in DNA strands to regularly occurring events in astronomy and economy, the study of repetitive phenomena is generally the foundation on which theories are laid out. Repetitions can be of several types. In everyday language repetition may suggest similarity ranging from identical matches of several objects observed at little or no distance (be that space or time, etc.) from each other to approximate reproductions of an event happening possibly with a large time difference.

In mathematics repetitions can occur in a wide variety of contexts. Binary idempotencies are an example of this phenomenon and we will study a particular idempotence relation in Chapter 6. In computer science, perhaps the most evident and earliest example is the appearance of subroutines. Writing the same piece of code repeatedly even if in a slightly different form is far from efficient. This was recognised early on by computer scientists and they came up with reusable program parts to save storage space and working hours. This takes us to the idea of compactness. Not to waste valuable storage space it is often required to compress the stored data. This is most often achieved by, roughly speaking, replacing repeatedly occurring data blocks with some shorter code that identifies this block and thus it is only required that the duplicated block is stored once.

A great number of papers and books on Combinatorics on Words start by revoking the work of Axel Thue on square free sequences. Thus it turns out that the theoretical study of repetitions in strings marks the beginning of Combinatorics on Words. Avoiding repetitions and other patterns became a central topic in combinatorics since then.

Theory in this domain has very close links to practical applications. The software industry boom experienced in the last few years happened to a large extent due to emerging companies which turned searching and structuring information into a lucrative business. The algorithms used in text-search and information retrieval are in many cases direct results of theoretical work in the field of Combinatorics on Words. A great number of these algorithms used for searching and producing compact representation of information are based on finding and processing repetitive substrings.

A basic notion in string combinatorics connected to repetitions is primitivity. Primitive words are simply words that are not formed by concatenating several copies of another word. Primitive words have received strong interest both in Combinatorics on Words and in Formal Language Theory. The latter field has grown out of the efforts of Church, Kleene, Post Turing and Chomsky. Although basically part of Theoretical Computer Science, this field always had tight relations with linguistics. The "holy grail" of formal language theorists is finding a suitable generative or accepting device to model natural language grammars. Here, suitable is subject to interpretation; however, a common ground is probably that context-free grammars are not quite enough and context-sensitive languages are too complex for efficient parsing. Hence,a class of languages in between should be found, one that retains the beneficial properties of both classes, namely it is powerful enough to describe natural languages but still parsable in deterministic polynomial time. One of the arguments brought in favour of a class wider than that of the context-free languages is that in some languages repetitive constructions like $a^m b^n a^m b^n$ or ww may appear and context free grammars are not capable of generating these languages. We can see that the topic of repetitions is fundamental in formal languages as well. Besides the questions with obvious practical importance, there have been a number of purely theoretical problems proposed in relation with primitive words.

A well known example is the question of the language of all primitive words being a context-free language. Dömösi et al. posed this problem in [13]. Despite the numerous attempts to prove the intuitive negative answer, the problem is still open after more than 15 years. In connection with this problem, regular and context-free languages consisting solely of primitive or non-primitive words have been investigated. We will reflect on this topic in Chapter 5.

Another problem omnipresent in science is the inference of an entire object from partial information. An instance of this problem in computer science is the inference of strings from information about its subsequences. The chapter on subwords in [41] deals in detail with this. What is really interesting here is the case of words having repeating subsequences or being entirely repetitive themselves, since for words which are highly non-repetitive, we often require an amount of partial information that is close to the information content of the entire string.

Finally we mention a field which receives a lot of attention lately and where repetitions play a central role. Ever since the beginning of the Human Genome Project the need to process large data sets defined the research directions in Bioinformatics. We already mentioned how the algorithms used in this field relate in many ways to repetitions in strings. There is another aspect, though, which pushes repetitions into the spotlight. One of the operations by which the genome evolves is the so called gene repeating duplication. Basically, this means the reoccurrence of a previously existing nucleotide sequence at another spot on a DNA strand. Analysis of these gene duplications can lead to results of public interest as in some cases it holds clues to the function or the origins of specific genes. This gave rise to the study of duplications in computer science, which we will detail in Chapter 6.

Repetition is such a general notion that we believe that it exceeds the scope of a PhD thesis to cover all aspects of it even when restricted to computer science. We will try to give an overview of the topics mentioned above and present results to some problems from these topics.

Chapter 2 surveys briefly the main concepts and notations needed in the other chapters. Very basic results on words, languages and automata theory are given.

Chapter 3 considers the bounds for the maximum number of integer and fractional powers occurring in strings. The $\Theta(n \log n)$ bound for square occurrences in strings of length n is known since [6]. The upper bound for squares implies the same for higher integer powers and we show how to reach the same lower bound for these powers. We also give a $\Theta(n^2)$ bound for both the maximal number of distinct powers and the maximal number of occurrences of powers with exponent between 1 and 2. The number of distinct squares in a word is conjectured to be at most n in a word of length n. We cannot settle this question but we prove that this bound holds for higher powers, in fact it decreases as the exponent grows.

In Chapter 4 we investigate conditions given in terms of scattered subword multiplicities for words to be non-primitive. First we take a look at the Parikh image of languages consisting only of primitive words and then we move on to state a characteristic inequality which is a necessary condition for a word to be non-primitive. We also analyze an open problem posed by Mateescu et al. [46] concerning the fundamental question of decidability of scattered subword inequalities. We provide partial results but the general problem remains open.

In Chapter 5 we discuss some problems related to repetitions occurring in formal language theory. After briefly surveying the state of research on the open problem of Dömösi et al. mentioned previously, we go on to solve a decidability question posed by Calbrix and Nivat [5] regarding the so called power of a regular language. Previous attempts by Cachat [4] and Horváth et al. [27] resulted in partial solutions. Rather than extending their results, we choose an alternative path and give a positive answer to the decidability problem.

In Chapter 6 we describe the mathematical model of gene repeating duplication, introduced for this purpose by Dassow et al. [11], although it was studied before with different terminology (see copying systems [15]). We present a reproof of a theorem by Bovet and Varricchio [3] which states that over a binary alphabet, the duplication closure of any recursively enumerable language is regular. The original proof that duplication over a binary alphabet is a well quasi order, is done through demonstrating by contradiction that no anti-chain exists with respect to binary duplication. Our proof, on the other hand, reduces this order to the scattered subword order. It is shorter and simpler than the one by Bovet and Varricchio and perhaps more illustrative of the relation itself.

The material of the thesis comes mostly from papers [7], [19], [20], [16], [17] and [18] authored or co-authored by me. However, some new results are introduced and some of the proofs are improved.

Chapter 2

Background

In this chapter we will introduce the notation used throughout the thesis and describe the context of the results presented later. First we will give a short overview of basic notions in combinatorics on words. The theorems presented in this section are at the very foundation of the field and perhaps it is not exaggerated to say that they come up in almost all works dealing with combinatorial properties of words. Among these we can find the theorem of Fine and Wilf, the Defect theorem and some other so called folklore results that are widely used, but it is not always possible to trace back their origins to a specific paper. In the second section of the chapter we give a short presentation of formal language theory. We look at the Chomsky-hierarchy of languages, the equivalent classes of accepting machines, as well as several basic results and closure properties that will come handy in our treatise.

2.1 Combinatorics on words

Let Σ be a set that we shall call an *alphabet.* Its elements will be called *letters.* In this thesis we only deal with words over a finite alphabet, so without further mentioning we suppose that Σ is finite.

A word over the alphabet Σ is a finite sequence of elements of Σ:

$$(a_1, a_2, \ldots, a_n), \qquad a_i \in \Sigma.$$

We will use non-capital letters from the beginning of the English alphabet to denote letters and non-capitals from the end to denote words. The set of all words over alphabet Σ is denoted by Σ^*. It is equipped with a binary operation obtained by concatenating two sequences.

$$(a_1, a_2, \ldots, a_n)(b_1, b_2, \ldots, b_m) = (a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m).$$

We will call this operation *concatenation*. It is associative, which allows writing a word as

$$a_1 a_2 \cdots a_n$$

instead of

$$(a_1, a_2, \ldots, a_n),$$

by identifying a letter $a \in \Sigma$ with the sequence (a).

The empty sequence, called the *empty word*, is a neutral element for the operation of concatenation. It is denoted by λ. For any word w:

$$\lambda w = w\lambda = w \ .$$

The concatenation of two sets of words A and B is denoted by AB and represents the set

$$AB = \{uv \mid u \in A \wedge v \in B\}.$$

The repeated concatenation of words or sets will be denoted as powers. That is u^2 represents uu and $A^n A$ we will write as A^{n+1}. The Kleene-closure or star closure of a (possibly empty or singleton) set of words A is

$$A^* = \{\lambda\} \cup A \cup A^2 \cup A^3 \cup ...$$

A semigroup is a set with an associative binary operation, hence $\Sigma^* \setminus \{\lambda\}$ equipped with concatenation is a semigroup. A *monoid* is a set M with a binary operation that is associative and has a neutral element denoted by 1_M, that is a semigroup completed with a neutral element. Hence, what has been defined on the set Σ^* is a monoid structure.

A *morphism* of a monoid M into a monoid N is a mapping φ of M into N that preserves the operations of M and N and translates the neutral element of M into that of N:

$$\varphi(mm') = \varphi(m)\varphi(m'), \qquad m, m' \in M,$$

and such that $\varphi(1_M) = 1_N$.

Proposition 1. *For any mapping α of Σ into a monoid M, there exists a unique morphism φ of monoids from Σ^* into M such that the following diagram is commutative:*

$$\begin{tikzcd}
\Sigma \arrow[r, "i"] \arrow[dr, "\alpha"'] & \Sigma^* \arrow[d, "\varphi"] \\
 & M
\end{tikzcd}$$

where i is the natural injection of Σ into Σ^*.

Because of this property, the set Σ^* of all words over the alphabet Σ is called the *free monoid* over the set Σ.

The set of all nonempty words over Σ will be denoted by Σ^+:

$$\Sigma^+ = \Sigma^* \setminus \{\lambda\}.$$

Σ^+ is called the *free semigroup* over Σ. It may be readily verified that Proposition 1 can be stated for Σ^+ instead of Σ^* by replacing the term "monoids" by "semigroups".

Let $u = a_1a_2...a_n$ be a word over the alphabet Σ. The *length* of the word $w = a_1a_2 \cdots a_n$, $a_i \in \Sigma$ is the number n of the letters w is a product of. It will be denoted by $|w|$:

$$|w| = n\,.$$

The length of the empty word is 0. The length of a word can be computed by the mapping $|| : \Sigma^* \to I\!N$, which is a morphism of the free monoid A^* onto the additive monoid $I\!N$ of positive integers.

For a letter a of the alphabet Σ, we denote by $|w|_a$ the number of occurrences of a in w, e.g. $|aab|_a = 2$.

Denoted by $alph(w)$ is the subset of the alphabet formed by the letters actually occurring in w. Therefore $a \in \Sigma$ belongs to $alph(w)$ iff

$$|w|_a \geq 1 \, .$$

A word $v \in \Sigma^*$ is said to be a *factor* of a word $x \in \Sigma^*$ if there exist words $u, w \in \Sigma^*$ such that

$$x = uvw \, .$$

The relation "v is factor of x" is an order on Σ^*. A factor v of $x \in \Sigma^*$ is said to be *proper* if $v \neq x$.

A word v is said to be a *prefix* of $x \in \Sigma^*$ if there exists a word $w \in \Sigma^*$ such that

$$x = vw \, ,$$

and w is said to be a *suffix* if $v \neq x$. The relation "v is a prefix of x" is again an order on Σ^*; it will be denoted by

$$v \leq x \, .$$

This order has the fundamental property that if

$$v \leq x, \qquad v' \leq x,$$

then v and v' are comparable: $v \leq v'$ or $v' \leq v$.

More precisely, if

$$vw = v'w',$$

either there exists $s \in \Sigma^*$ such that $v = v's$ (and then $sw = w'$) or there exists $t \in \Sigma^*$ such that $v' = vt$ (and then $w = tw'$).

The definition of a *suffix* is symmetrical to that of a prefix. The *reversal* of a word $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$, is the word

$$w_r = a_n \cdots a_2 a_1 \, .$$

Hence v is a prefix of x iff v_r is a right factor of x_r. For all $u, v \in \Sigma^+$,

$$(uv)_r = v_r u_r \,.$$

This last property means that reversal is a so called *anti-morphism*. A word w is palindrome if $w = w_r$.

A word $v \in \Sigma^*$ is said to be a (*scattered*) *subword* of a word $x \in \Sigma^*$ if

$$v = a_1 a_2 \cdots a_n, \qquad a_i \in \Sigma, n \geq 0,$$

and there exist $y_0, y_1, \ldots, y_n \in \Sigma^*$ such that

$$x = y_0 a_1 y_1 a_2 \cdots a_n y_n \,.$$

Therefore v is a subword of x if it is a sub-sequence of x. The term subword in the literature is often used also with the same meaning as factor, and what we call here subwords are referred to as scattered subwords. However, as we are mostly concerned with subsequences, we will use the shorter subword for them and keep factor to refer to continuous subsequences. In accordance with $|w|_a$ meaning the number of occurrences of the letter a in the word w we extend the notation to subwords, i.e. by $|w|_u$ the number of times u occurs as a subword of w with any two occurrences differing by at least one position. Looking at our previous example the word ab occurs twice in aab because we can choose to match the first letter of ab with any of the two a's in aab, therefore $|aab|_{ab} = 2$. In [41] the notation used is $\binom{aab}{ab}$, taken from the binomial coefficients. Indeed, over a unary alphabet, $|a^n|_{a^k} = \binom{n}{k}$. From here we acquire the convention that the empty word occurs once in any word, that is $|w|_\lambda = 1$ ($|\lambda| = 0$ and $\binom{n}{0} = 1$). The prefix, suffix and scattered subword orders mentioned above are all partial orders, i.e. it is possible for two words to be incomparable with respect to them. Natural extensions of the prefix order to total orderings are the *lexicographic* and the *alphabetic* orders. Suppose our alphabet Σ is totally ordered by some relation $\prec$. The lexicographic order is defined as follows: $u \prec_l v$ if

- u is a prefix of v or

- there exist words $x, y, z \in \Sigma^*$ and letters $a \prec b$ such that $u = xay$ and $v = xbz$.

The alphabetic order $\prec_a$, as opposed to the lexicographic one, takes into account the length of the two sides: $u \prec_a v$ if

- $|u| < |v|$ or
- $|u| = |v|$ and there exist words $x, y, z \in \Sigma^*$ and letters $a \prec b$ such that $u = xay$ and $v = xbz$.

Note that for words of the same length the two orders coincide. Now let us turn to another fundamental property of words that is very relevant to us, periodicity. We will introduce the notions of conjugacy, primitive words and a few famous results concerning these.

A word $x \in \Sigma^*$ is said to be *primitive* if it is not a power of another word; that is, if $x \neq \lambda$ and $x \in z^*$ for $z \in \Sigma^*$ implies $x = z$.

Two words x and y are said to be *conjugate* if there exist words $u, v \in \Sigma^*$ such that

$$x = uv, \qquad y = vu. \tag{2.1}$$

This is an equivalence relation on Σ^* since x is conjugate to y iff y can be obtained by a cyclic permutation of the letters of x. A primitive word w is a *Lyndon word* if it is minimal with respect to the lexicographic order among its conjugate words, that is

$$u = a_1a_2 \dots a_n \textit{ is a Lyndon word} \Leftrightarrow \forall i,\ 1 < i \leq n,\ u \prec_l a_i \dots a_n a_1 \dots a_{i-1}.$$

Proposition 2. *If*

$$x^n = y^m, \qquad x, y \in \Sigma^*,\ n, m \geq 0,$$

there exists a word z such that $x, y \in z^*$.

In particular, for each word $w \in \Sigma^+$, there exists a unique primitive word x such that $w \in x^*$.

The primitive word x in our previous proposition is called the *primitive root* of w and we denote it by $\sqrt{w}$. We can extend the meaning of this term to languages, i.e. arbitrary sets of words for now. For a set A of words, $\sqrt{A}$ is the collection of all the roots of the words in A, formally

$$\sqrt{A} = \{\sqrt{w} \mid w \in A\} .$$

We will call $\sqrt{A}$ the primitive root of the language A.

Proposition 3. *Two words $x, y \in \Sigma^+$ commute ($xy = yx$) iff they are powers of the same word. More precisely the set of words commuting with a word $x \in \Sigma^+$ is a monoid generated by a single primitive word.*

Proposition 4. *Let $x, y \in \Sigma^*$ and z, t be the primitive words such that $x \in z^*$, $y \in t^*$. Then x and y are conjugate iff z and t are also conjugate; in this case, there exists a unique pair $(u, v) \in \Sigma^* \times \Sigma^+$ such that $z = uv$, $t = vu$.*

Proposition 5. *Two words $x, y \in \Sigma^+$ are conjugate iff there exists a $z \in \Sigma^*$ such that*

$$xz = zy. \tag{2.2}$$

More precisely, equality (2.2) holds iff there exist $u, v \in \Sigma^*$ such that

$$x = uv, \qquad y = vu, \qquad z \in u(vu)^*. \tag{2.3}$$

Before we arrive to discuss periods, let us say a few words about the defect theorem, which is a very important result on words. It is often considered to be a folklore result due to the fact that there are many papers and results describing the same phenomenon. Without presenting the rather various formalizations of this theorem, we will give the following intuitive description: if a set of n words satisfies a non-trivial relation, then these words can be expressed simultaneously as products of at most $n - 1$ words.

It may be observed that, in accordance with the defect theorem, the equality $xz = zy$ implies $x, y, z \in \{u, v\}^*$, a submonoid with two generators.

At this point we have to define periodicity. If a word w is a prefix of some u^k, with u primitive, $|u| < |w|$ and $k > 1$ then we say that w is *periodic* and $|u|$ is its *period*. We can relax the requirements of u being primitive and say that if $w(i) = w(i + p)$ for some $p \geq 1$ and for all $1 \leq i \leq |w| - p$ then p is a period of w. Unless stated otherwise we will use period meaning the latter. Sometimes in the literature it is required that the length of w is a multiple of the length of u, however we will allow for so called *fractional powers.* We have seen earlier that uu is called a square, u^k a k-th power of u, etc. In accordance with this we say that a word $w = (uv)^k u$ for some non-empty words u and v is a power of exponent $k + \frac{|u|}{|u|+|v|}$. Generalizing the propositions above we can get periodicity enforcing conditions. These are very much sought after in the field and several of them are known. Let us see some of the most common properties ensuring periodicity:

- any non-trivial relation on $\{x, y\} \subseteq \Sigma^*$;
- any pair of non-trivial identities on $X = \{x, y, z\} \subseteq \Sigma^+$ of the form $x\alpha = y\beta$, $y\gamma = z\delta$ with $\alpha, \beta, \gamma, \delta \in X^*$;
- any condition on $X = \{x_1, x_2, \ldots, x_n\} \subseteq \Sigma^+$ if the transitive closure relation ρ defined as

  $$x\rho y \ \leftrightarrow xX^\omega \cap yX^\omega = \emptyset$$

  equals $X \times X$. Here ω is the infinite power operator.

Another classical result enforcing periodicity is the Lyndon-Schützenberger theorem:

Proposition 6. *If the equation*

$$u^m v^n = w^k$$

for some non-empty words u, v, w and exponents $m, n, k \geq 2$ then all three words u, v and w are powers of the same word.

Next we are going to present a refinement of Proposition 2 known as the theorem of Fine and Wilf. Intuitively it tells us how far two periodic events (strings) have to match in order to guarantee a common period, that is to guarantee that the two sequences are ultimately the same.

Proposition 7. *Let $x, y \in \Sigma^*$, $n = |x|$, $m = |y|$, $d = gcd(n, m)$. If two powers x^p and y^q of x and y have a common left factor of length at least equal to $n + m - d$, then x and y are powers of the same word.*

Example 1. *Consider the sequence of words on $\Sigma = \{a, b\}$ defined as follows: $f_1 = b$, $f_2 = a$ and*

$$f_{n+1} = f_n f_{n-1}\,, \quad n \geq 2.$$

The sequence of the lengths $\mid f_n \mid$ is the Fibonacci sequence. Two consecutive elements $|f_n|$ and $|f_{n+1}|$ for $n \geq 3$ are relatively prime. Let g_n be the left factor of f_n of length $|f_{n-2}|$ for $n \geq 3$. Then

$$g_{n+1} = f_{n-1}^2 g_{n-2}$$

for $n \geq 5$, as it may be verified by induction. We then have simultaneously

$$f_{n+1} \leq f_n^2, \quad g_{n+1} \leq f_{n-1}^3\,.$$

Therefore, for each $n \geq 5$, f_n^2 and f_{n-1}^3 have a common left factor of length $|f_n| + |f_{n-1}| - 2$. This shows that the bound given by Proposition 7 is optimal. For instance,

$$g_7 = \rlap{$\overbrace{}^{f_6}$}\underbrace{abaab}_{f_5}\underbrace{abaab}_{f_5}a$$

The periodicity theorem of Fine and Wilf can be restated in various forms, among them as follows, which are sometimes more comfortable to use.

Corollary 1. *If a word $w \in \Sigma^+$ has periods p and q, and $|w| \geq p + q - gcd(p, q)$ then it also has a period $gcd(p, q)$.*

As we have seen in Example 1, the bound is optimal in general. In fact it is possible to find many more examples certifying the bound.

Proposition 8. *For each pair (p, q) of coprimes there exists a unique binary word (up to a renaming) w of length $p + q - 2$ such that both p and q are periods of w. More generally for coprimes $p > q$ and some k with $2 \leq k \leq q$, there exists a unique word w_k (up to a renaming) such that*

- $|w_k| = p + q - k$,
- $|alph(w)| = k$ *and*

- *both p and q are periods of w_k.*

With this we conclude the introductory section on combinatorics on words. There will be several other notions and results of the topic used later, but as they are not of general nature to the whole thesis, they will be presented in due course at the beginning of the respective chapters.

2.2 Formal languages and automata theory

Since the 1950's, the subject of formal language theory, hand in hand with automata theory, has been developed by computer scientists, linguists and mathematicians. Formal languages (or simply languages) are sets of words over finite sets of symbols, called alphabets. There are many ways to describe such languages starting from set theoretical presentations which specify certain properties that hold for some words and including regular expressions (which "generate" languages), finite automata (which "accept" languages), grammars (which "generate" languages) and Turing machines (which "accept" languages). Many natural examples of formally presentable languages can be found even outside computer science. However, the interest in the theory was mainly brought about by the need of a formal tool to tackle some problems occurring early in computing. Examples of languages among the ones first studied are the set of identifiers of a given programming language, which can be described by a regular expression or a finite automaton, and the set of all strings of tokens that are generated by the grammar of a programming language. There are many categorizations of languages based on the device generating (accepting) them, based on the time or space complexity of their membership problem, the number of words in the language at given lengths or even based on some properties of their morphic images. In this section we will present the grammar categories known as the Chomsky-hierarchy and the abstract machines accepting the languages generated by these grammars. A particular emphasis is laid on regular languages, an exciting language class, where several different ways of describing a language come to common ground.

A grammar is a construct $G = \langle N, \Sigma, S, H \rangle$, where N, Σ are the non-terminal and terminal alphabets, with $N \cap \Sigma = \emptyset$; they are finite sets. $S \in N$ is a special symbol, called the start symbol. H is a finite set of pairs, where a pair is written in the form $v \to w$ with $v \in \{N \cup \Sigma\}^* N \{N \cup \Sigma\}^*$ and $w \in \{N \cup \Sigma\}^*$. H is the set of derivation rules. A sequence of letters $v \in \{N \cup \Sigma\}^*$ is called a sentential form. As in the previous section, we refer to elements of Σ^* as words and we use λ to denote the empty word.

Let G be a grammar and $v, w \in \{N \cup \Sigma\}^*$. Then $v \Rightarrow w$ is a direct derivation if and only if there exist $v_1, v_2, v', w' \in \{N \cup \Sigma\}^*$ such that $v = v_1 v' v_2$, $w = v_1 w' v_2$ and $v' \to w' \in H$. A derivation $v \Rightarrow^* u$ holds if and only if either $v = u$ or there is a finite sequence of sequential forms which connects them, i.e. $v = v_0, v_1, ...v_m = u$ in which $v_i \Rightarrow v_{i+1}$ is a direct derivation for each $0 \leq i < m$.

The language generated by a grammar G is the set of (terminal) words that can be derived from the start symbol: $L(G) = \{w | S \Rightarrow^* w \wedge w \in \Sigma^*\}$.

The Chomsky hierarchy of languages is based on the following classification of generative grammars. • type 1, or context-sensitive grammars: all derivation rules are in the form $v_1 A v_2 \to v_1 w v_2$, with $v_1, v_2 \in \{N \cup \Sigma\}^*$, $A \in N$ and $w \in (N \cup \Sigma)^* \setminus \{\lambda\}$ (except possibly for the rule $S \to \lambda$, in which case S does not occur on any right hand side of a rule).

• type 2, or context-free grammars: for every rule the next scheme holds: $A \to v$ with $A \in N$ and $v \in \{N \cup \Sigma\}^*$.

• type 3, or right-linear grammars: each derivation rule is one of the following forms: $A \to w$, $A \to wB$; where $A, B \in N$ and $w \in \Sigma^*$.

As languages are simply set of words, the usual set operations are valid for languages as well. That is for languages $L_1, L_2 \subseteq \Sigma^*$, their union $L_1 \cup L_2$, intersection $L_1 \cap L_2$, concatenation $L_1 L_2$ and difference $L_1 \setminus L_2$ are languages too defined the same way it goes for arbitrary sets. If we start out with the simplest building blocks and apply some of these operations and the Kleene-closure defined in the previous section we get so called *regular expressions*. For a regular expression e we denote by $L(e)$ the language described by e. Let us see the inductive definition for regular expressions and the languages generated by them.

- $\emptyset$ is a regular expression and $L(\emptyset) = \emptyset$;
- λ is a regular expression and $L(\lambda) = \{\lambda\}$;

- for all $a \in \Sigma$, a is a regular expression and $L(a) = \{a\}$;
- for arbitrary regular expressions e_1 and e_2:

  $(e_1)^*$,

  (e_1e_2) and

  $(e_1 \cup e_2)$ are all regular expressions with the languages generated by them being $L(e_1)^*$, $L(e_1)L(e_2)$ and $L(e_1) \cup L(e_2)$, respectively.

We call a language L regular if there exists a regular expression e such that $L = L(e)$. Now we will define finite automata, a class of abstract machines that happen to accept the languages generated by regular expression or by right linear grammars.

A *finite automaton* is a quintuple $\mathcal{A} = \langle \Sigma, Q, q_0, F, \sigma\}\rangle$ with the following components

- Σ is the *input alphabet*,
- Q is a finite set called the *set of states*,
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the *set of final states* and
- $\sigma : Q \times \Sigma \to 2^Q$ is the *transition function.*

The language accepted by the finite automaton $\mathcal{A}$ is

$$L(\mathcal{A}) = \{w = a_1a_2 \ldots a_n \mid \exists q_1, q_2, \ldots, q_{n-1} \in Q, q_n \in F \ s.t. \forall 1 \leq i \leq n : \ q_i \in \sigma(q_{i-1}, a_i)\}.$$

Intuitively, a finite automaton can be thought of as a directed graph where the vertices are labeled with the states and the edges are labeled with the transitions. An word is accepted by the automaton if there is a path from the vertex labeled with the initial state to a vertex labeled with a final state composed of edges labeled with the letters of the word following each other in the correct order.

Example 2. *Take the automaton $\mathcal{A} = \langle \{0,1\}, \{q_0, q_1\}, q_0, \{q_0\}, \sigma\rangle$, where the transition function σ is given by the transition table in Figure 2.*

σ	0	1
q_0	q_0	q_1
q_1	q_1	q_0

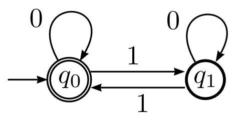

Figure 2.1: Finite automaton accepting $(0^*10^*10^*)^*$

The language accepted by automaton $\mathcal{A}$ is

$$L(\mathcal{A}) = \{w \in \Sigma^* \;|\exists k \geq 0 : \; |w|_1 = 2k\},$$

that is all binary words having an even number of 1's. For any occurrence of a 1, the automaton switches states, whereas upon reading a 0 it stays in the same state. If it switched an even number of times by the time of reaching the end of the input word, it accepts, otherwise rejects.

The definition we gave for finite automata is in fact the definition for the general case of non-deterministic automata. Non-determinism in automata has to do with the possibility to transition from a state to more than one others by reading the same input letter. Therefore if we modify our transition function σ to map from $Q \times \Sigma$ to Q instead of 2^Q we get the definition of deterministic finite automata. As it turns out, the class of languages accepted by non-deterministic and deterministic automata are the same. This is a fact widely applied in automata theoretic proofs. Our example automaton is actually a deterministic one. According to the famous theorem by Kleene, in the free semigroup Σ^* the language classes accepted by finite automata, described by regular (rational) expressions and generated by right-linear grammars are the same. This is a very important theorem of language theory and provides the foundations for most results concerning regular languages, since proofs are generally significantly easier or even only doable by reasoning with automata rather than regular expressions or grammars. Given two finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$ there are algorithms available to construct the automata accepting the languages $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, $\overline{L(\mathcal{A}_1)} = \Sigma^* \setminus L(\mathcal{A}_1)$ and therefore $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$. Moreover, we have

	REG	CF	CS
L_1^*	●	●	●
$\overline{L_1}$	●		●
$L_1 \cup L_2$	●	●	●
$L_1 \cap L_2$	●		●

Figure 2.2: Some closure properties of Chomsky language classes.

algorithms for determinising and minimising automata, which makes it possible to decide the equality of the accepted languages. From Kleene's theorem we have the same properties for languages described by regular expressions. The closure properties of the Chomsky language classes are summarised in the table in Figure 2.2.

As it is not really relevant to our thesis, we only mention here that the class of context-free languages is the same as the class accepted by push-down automata, whereas the context-sensitive class is accepted by linear-bounded Turing machines. Finally, let us explain the notion of Parikh sets and semi-linearity.

Definition 1. *For a word w the Parikh vector $\psi(w)$ associated to a w is an n-tuple, where n is the cardinality of the alphabet $\Sigma = \{a_1, \ldots, a_n\}$, defined in the following way:*

$$\psi(w) = (w_{a_1}, \ldots, w_{a_n}).$$

This can be extended to languages, taking the set of all Parikh-vectors associated to the words of the language. This set is usually called the Parikh image (Parikh set) of the language.

A language L is linear in the Parikh sense if the Parikh-set of the language is the linear combination of some vectors V_i.

$$\psi(L) = \left\{ V_0 + \sum_i c_i V_i \right\}.$$

A language L is semi-linear (in the Parikh-sense) if the Parikh-set of the language is a finite union of some linear sets.

$$P_L = \bigcup_k \left\{ V_{0,k} + \sum_i c_{i,k} V_{i,k} \right\}$$

Parikh's theorem tells us that the Parikh image of any context-free language is a semi-linear set. This can be used to prove that a given language is not context-free. Other tools for placing languages correctly in the Chomsky hierarchy are the so called pumping lemmas. There are several of them known in the literature, but here we will only show the two most widely used, the pumping lemma for regular languages and the Bar-Hillel lemma for context-free languages.

Lemma 1. *(Pumping lemma for regular languages) For every regular language L there exist constants p and q depending only on the language such that every $w \in L$ with $|w| > p$ can be written in the form $w = uvz$ for some words $u, v, z \in \Sigma^*$, with $|uv| \leq q$, $v \neq \lambda$, and $uv^iz \in L$ for all $i \geq 0$.*

Lemma 2. *(Bar-Hillel lemma) If L is a context-free language then there exist $p, q \in I\!N$ such that every $w \in L$ with $|w| > p$ can be written in the form $w = uvxyz$, with $u, v, x, y, z \in \Sigma^*$, $|vxy| < q$, $|vy| > 0$ and $uv^ixy^iz \in L$ for all $i \geq 0$.*

With this we conclude the introductory chapter. For a more detailed introduction to the theory of formal languages one might consult the first chapters of [51].

Chapter 3

Bounds on powers

The subject of this section is the evaluation of the number of powers in strings. This is one of the most fundamental topics in combinatorics on words not only for its own combinatorial aspects considered since the beginning of last century by the precursor A. Thue [59], but also because it is related to lossless text compression, string representation, and analysis of molecular biological sequences, to quote a few applications. These applications often require fast algorithms to locate repetitions because either the amount of data to be treated is huge or their flow is to be analysed on the fly, but their design and complexity analysis depends of the type of repetitions considered and of their bounds.

As defined in Chapter 2 a repetition is a string composed of the concatenation of several copies of another string whose length is called a period. The exponent of a string is informally the number of copies and is defined as the ratio between the length of the string and its smallest period. This means that the repeated string, called the root, is primitive (it is not itself a nontrivial integer power). Here we consider two types of strings: integer powers—those having an integer exponent at least 2, and fractional powers—those having a fractional exponent between 1 and 2. For both of them we consider their maximal number in a given string as well as their maximal number of occurrences.

It is known that all occurrences of integer powers in a string of length n can be computed in time $O(n \log n)$ (see three different methods in [6], [1], and [43]). Indeed these algorithms are optimal because the number of occurrences of squares (powers of exponent 2) can be of the order of $n \log n$ [6].

The computation of occurrences of fractional powers with exponent at least 2 has been designed initially by Main [42] who restricted the question to the detection of their leftmost maximal occurrences only. Eventually the notion of runs—maximal occurrences of fractional powers with exponent at least 2—introduced by Iliopoulos et al. [30] for Fibonacci words led to a linear-time algorithm for locating all of them on a fixed-sized alphabet. The algorithm, by Kolpakov and Kucherov [33, 34], is an extension of Main's algorithm but their fundamental contribution is the linear number of runs in a string. They proved that the number of runs in a string of length n is at most cn, could not provide any value for the constant c, but conjectured that $c = 1$. After that, in a series of papers dealing with runs, this bound received a lot of attention and has been improved as follows:

- Rytter [52] proved that $c \leq 5$,
- then $c \leq 3.44$ in [53],
- Puglisi et al. [50] that $c \leq 3.48$,
- Crochemore and Ilie [8] that $c \leq 1.6$,
- and Giraud [23] that $c \leq 1.5$.

The best value computed so far is $c = 1.048$ [9] (for further details see the Web page `http://www.csd.uwo.ca/ ilie/runs.html`). Franek et al. showed a lower bound of $0.927...n$ in [22], which was improved by Matsubara et al. in [37] to $0.944565n$. These lower bounds also point in the direction of Kolpakov and Kucerov's conjecture. Runs capture all the repetitions in a string but without discriminating among them according to their exponent. For example, the number of runs is not easily related to the number of occurrences of squares. This is why we consider an orthogonal approach here. We count and bound the maximal number of repetitions having a fixed exponent, either an integer larger than 1 or a fractional number between 1 and 2. We also bound the number of occurrences of these repetitions.

After introducing the notations and basic definitions which we did not meet yet Section 3.1 deals with fractional powers with exponent between 1 and 2. It is shown that the maximum number of primitively-rooted powers with a given exponent e,

$1 < e < 2$, in a string can be quadratic as well of course as their maximum number of occurrences. In Section 3.2, we consider primitively-rooted integer powers and show that the maximum number of occurrences of powers of a given exponent k, $k \geq 2$, is $\Theta(n \log n)$. This latter result contrasts with the linear number of such powers. We also present an efficient algorithm for constructing the strings in question. Finally, we show that distinct repetitions of a fixed exponent greater than 2 have an upper bound of n in a string of length n. This is an improvement of the implied value coming from the bound for distinct squares established in $2n$ by Fraenkel and Simpson in [21] and improved by Ilie to $2n - O(\log n)$ in [29].

Recall that a k-th power is the concatenation of k copies of a non-empty word, and extending this definition we can talk about e-powers with $1 < e < 2$ as seen in the introductory chapter. A prefix of the length of a period of w is a root of w. When $w \neq \epsilon$, w^3 is called a *cube*, with *root* w. Take a primitive word uv, such that vu forms a Lyndon word and v is nonempty. In the cube $(uv)^3$, we call *central Lyndon* position the one at $|uvu|$, $uvu.vuv$. For two non-empty words u and v it is known that $uv = vu$ implies $u, v \in z^+$ for some $z \in A^*$, therefore every word has a unique Lyndon position.

If a word w can be written as $w = uv = vz$, for some words $u, v, z \in A^+$, then we say that w is *bordered* (v is a border of w). If a word w is bordered, then there exists $u \in A^+, v \in A^*$ such that $w = uvu$, that is a bordered word w always has a border of length at most half the length of w. Moreover, it is easy to see that a bordered word uvu cannot be a Lyndon word, because then either uuv (if $u < v$) or vuu (if $v < u$) is lexicographically smaller than uvu.

3.1 A bound on repeats with exponent e, with $1 < e < 2$

In this section, we show that the number of distinct repetitions with exponent e, with $1 < e < 2$ is bound by $\Theta(n^2)$. We do this by looking at the number of such repetitions that can start at a position in words of the form $a^k b a^{\frac{k}{e-1}-1}$, where k is any positive integer such that $c|k$, where $e = \frac{c+d}{d}$ and $gcd(c+d, d) = 1$.
First we consider an example with $e = \frac{3}{2}$ and $k = 9$, ie. $w = a^9 b a^{17}$ (see Fig 3.2).

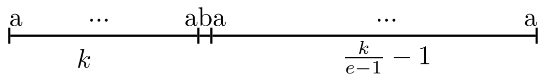

Figure 3.1: Structure of word, w

At the first position in this word, we can have 5 repetitions of exponent $\frac{3}{2}$, namely $a^9ba^5, a^9ba^8, a^9ba^{11}, a^9ba^{14}$ and a^9ba^{17}. Moving on to the second position, we will have only 4 repetitions of exponent $\frac{3}{2}$, namely $a^8ba^6, a^8ba^9, a^8ba^{12}$ and a^8ba^{15}. In the third position also, we are able to have the repetitions a^7ba^7, a^7ba^{10} and a^7ba^{13}. However, now we will have one extra repetition as we can also have a^7ba^4. It is clear that at every other position in the word, as we get closer to the b, we will have an extra repetition. The number of repetitions of exponent $\frac{3}{2}$ at each position are now $5, 4, 4, 3, 3, 2, 2, 1, 1$ (see Fig. 3.2). The total number of repetitions can now be summed up to $((5 * 6)/2) + (((5-1) * 5)/2) = 25$. We will generalise this example in the next theorem.

Theorem 1. *The maximal number of distinct repetitions of exponent e, with $1 < e < 2$, in a word of length n is $\Theta(n^2)$.*

Proof. The upper bound is trivial because no factor of the string can be counted twice as an e-th power for given e, so let us turn to proving the lower bound.
We shall count the number of repetitions starting at each position in a word. For an exponent, e, with $1 < e < 2$, we consider a word, w, formed as shown in Fig. 3.1. Here, we concatenate a repetition of exponent, e, with root a^kb and $a^{\frac{k}{e-1}-1}$, where k is any positive integer such that $c|k$, where $e = \frac{c+d}{d}$ and $gcd(c+d, d) = 1$. In this case the length of our string will be $k \cdot \frac{e}{e-1}$.

For e-powers starting at the first position, the end positions can be $(k+1)(e-1)$, $(k+1)(e-1) + (c+d)$, $(k+1)(e-1) + 2 \cdot (c+d)$,...
From here we get that the number of e-th powers starting at the first position is

$$\frac{|w| - (k+1)(e-1)}{c+d} + 1 = \frac{k \cdot \frac{e}{e-1} - (k+1)(e-1)}{c+d} + 1$$

Substituting $\frac{c+d}{d}$ for e in the formula above we get that the number of e-th powers

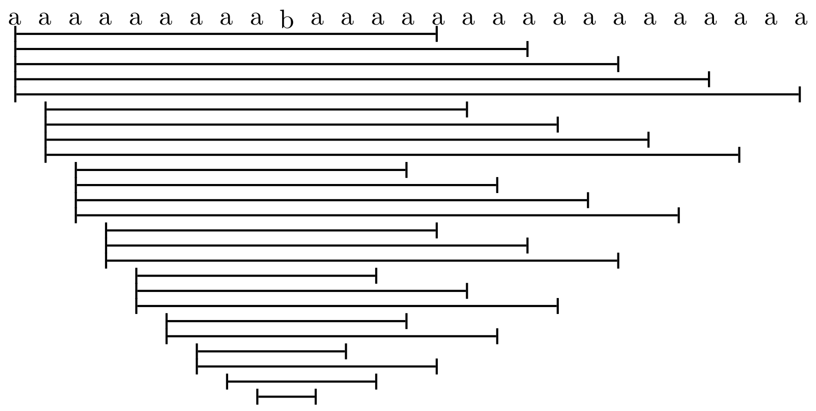

Figure 3.2: Repetitions of exponent 1.5 in a^9ba^{17}

starting at the first position is:

$$k \cdot \frac{d-c}{d \cdot c} - \frac{1}{d} + 1$$

This formula proves useful because by substituting $k-i$ for k and taking the integer part of the result (since we are talking about the number of occurrences) we get the number of e-th powers starting at position $i+1$. Now let us sum up the number of e-th power occurrences starting at any one of the first k positions:

$$\sum_{i=1}^{k} \lfloor i \cdot \frac{d-c}{d \cdot c} - \frac{1}{d} + 1 \rfloor$$

For any positive n its integer part $\lfloor n \rfloor$ is greater or equal than $n-1$. As we are trying to give a lower bound to the number of occurrences, it is alright to subtract 1 from the formula instead of taking its integer part:

$$\sum_{i=1}^{k} \left(i \cdot \frac{d-c}{d \cdot c} - \frac{1}{d} \right) = k \cdot (k+1) \cdot \frac{d-c}{2d \cdot c} - \frac{k}{d}$$

This means that the number of e-th powers in our string is quadratic in k. At the same time the length of the string, as we mentioned in the beginning, is $k \cdot \frac{e}{e-1}$, so for a given e, the number of e-th powers in a string of length n is $\Theta(n^2)$.
It is easy to see that every occurrence of an e-th power in this string is unique and this concludes the proof. □

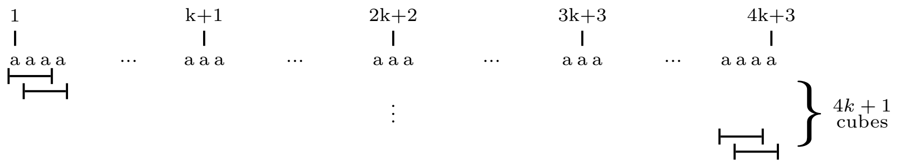

Figure 3.3: Cubes of word a^{4k+3}

3.2 A bound on primitively rooted cubes

After considering powers between 1 and 2, we shall take a look at powers greater than 2. First, we will show that it is possible to construct strings of length n, which have $\Omega(n \log n)$ occurrences of cubes. We can extend the procedure to all integer powers greater than 2, and this, together with the $O(n \log n)$ upper bound implied by the number of squares (see [6]) leads us to the $\Theta(n \log n)$ bound. Finally, we will prove that the sum of all occurrences of powers at least 2 (including non-integer exponents) is quadratic.

Lemma 3. *The maximal number of primitively rooted cubes in a word of length n is $\Omega(n \log n)$.*

Proof. Let us suppose there are two primitively rooted cubes $(uv)^3$ and $(xy)^3$ in w such that their central Lyndon positions $uvu.vuv$ and $xyx.yxy$ are the same. First let us look at the case where the cubes have to be of different length. Without loss of generality we can assume $|uv| < |xy|$. In this case vu is at the same time a prefix and suffix of yx. Hence, yx is bordered and cannot be a Lyndon word contradicting the assumption that $x.y$ is a Lyndon position. This proves that should there be more cubes which have their central Lyndon position identical, they all have to be of the same length. Naturally, the first and last position of a word cannot be central Lyndon to any cube and this gives us the bound $n-2$ if we disregard cubes of the same length which have their central Lyndon positions at the same place (see Fig. 3.3). It is easy to see, that because of the periodicity theorem the only string of length n, for which $n-2$ different positions are central Lyndon ones to some cube, is a^n.

Now take the word a^{4k+3}. According to our previous argument it has at most $4k+1$ cubes. However, if we change a's into b's at positions $k+1$,$2k+2$ and $3k+3$ we get that the number of primitively rooted cubes in this word is $4k+1-9+(k+1) =$

$5k-7$. This is because by introducing each b we lose three cubes but in the end we gain another $k+1$ cubes of the form $(a^j ba^{k-j})^3$ with $0 \leq j \leq k$ (see Fig. 3.4). Note that these latter cubes all have their central Lyndon position after the first b (assuming $a < b$).

We introduced three b's in the previous step but of course we can repeat the procedure for the four block of a's delimited by these b's and then in turn for the new, smaller blocks of a's that result and so on. In the second step, however, we need to introduce 12 b's - that is, 3 for each of the 4 blocks of a's - not to disrupt the cubes of length $3k+3$. This way we lose $12 \cdot 3 = 36$ cubes and we gain $(\lfloor (k-3)/4 \rfloor + 1) \cdot 4$ new ones. Performing the introduction of b's until the number of cubes we lose in a step becomes greater or equal to the ones we gain, gives us a string with the maximal possible number of cubes for its length. If k equals $4j$,

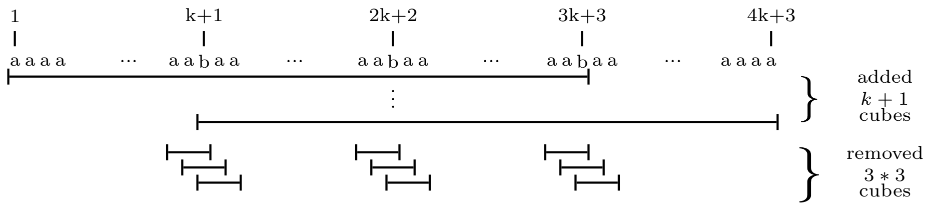
1
k+1
2k+2
3k+3
4k+3
a a a a ··· a a b a a ··· a a b a a ··· a a b a a ··· a a a a
added $k+1$ cubes
removed $3*3$ cubes

Figure 3.4: Cubes of word $a^k ba^k ba^k ba^k$

$4j+1$ or $4j+2$ for some j then according to the formula above the number of cubes we gain is $4j$. Note that if $k = 4j+3$ than the number of cubes we gain in the second step is $4j+4 = k+1$, i.e. the same as in the first step. However, together with the delimiting b's introduced before we would get a big cube which is not primitively rooted anymore, so we need to move the newly introduced b's 1, 2 and 3 positions to the left, respectively. This gives us that in this case too the number of newly formed cubes will be $4j$. The smallest length at which introducing the b's does not induce less cubes is 35 that is with $k = 8$. Summarizing the points above we get that for a string of length n the maximum increase in the number of cubes for the ith ($i > 1$) consecutive application of our procedure is:

$$\frac{(n-3)}{4} - 9 \cdot 4^{i-1}$$

To be able to sum these increases we have to know the number of steps performed.

This is given by solving for i the equation:

$$\frac{n-3}{4} = 9 \cdot 4^{i-1}$$

From here we get that the number of steps performed is $\#steps = \lfloor \log_4 \frac{n-3}{9} \rfloor$, where by $\lfloor x \rfloor$ we mean the integer part of x.

Hence the number of cubes for length $n \geq 39$ is:

$$n - 2 + 1 + \sum_{i=1}^{\#steps} \left(\frac{n-3}{4} - 9 \cdot 4^{i-1} \right)$$

$$= n - 1 + \frac{(n-3)\lfloor \log_4 \frac{(n-3)}{9} \rfloor}{4} - \frac{9(1 - 4^{\lfloor \log_4 \frac{n-3}{9} \rfloor})}{-3}$$

$$= n + 2 + \frac{(n-3)\lfloor \log_4 \frac{(n-3)}{9} \rfloor}{4} - 3 \cdot 4^{\lfloor \log_4 \frac{n-3}{9} \rfloor}$$

The plus one after $n-2$ comes from the first application of the insertion of b's where we get $(n-3)/4 + 1$ cubes instead of $(n-3)/4$. For strings shorter than 39 therefore the count is one less.

□

Since the first paragraph of the proof is valid for any integer power, we can extend the proof by giving the construction of the strings that prove the lower bound in general for a string of length n and power k (see Fig. 3.2).

Algorithm *ConstructStrings1*(n, k)
Input: $n \geq 0$, $k \geq 0$
Output: A string which proves the lower bound of the number of occurrences of integer powers.

```
1.  $\ell = n$
2.  string = $a^\ell$
3.  power$(1, \ell)$
4.  Procedure: power( start, end)
5.  $\ell$ = end - start
6.  **if** $\ell < k^3 + k^2 + k$
7.     **then** return
8.     **else** string[start + $\lfloor \ell/(k+1) \rfloor$] = $b$
9.           string[start + $2 \cdot \lfloor \ell/(k+1) \rfloor$] = $b$
10.          ...
11.          string[start + $k \cdot \lfloor ell/(k+1) \rfloor$] = $b$
12.          **for** $i \leftarrow 0$ **to** $k$
13.         power(start + $i \cdot \ell/(k+1)$, start + $(i+1) \cdot \ell/(k+1)$)
```

Algorithm *ConstructStrings2*(n, k)
Input: $n \geq 0$, $k \geq 0$
Output: A string which proves the lower bound of the number of occurrences of integer powers.

```
1.  $\ell = n$
2.  while $\ell \geq k^3 + k^2 + 3k + 2$
3.      do $\ell = \frac{\ell - k}{k+1}$
4.  $string = (a^{k^2+1} + b)^k + a^{(k+1)\cdot\ell - k^3 - k}$
5.  $delimiter = b$
6.  while $length(string) * (k+1) + k < n$
7.      do $string = (string + delimiter)^k + string$
8.         if $delimiter = b$
9.            then $delimiter = a$
10.           else $delimiter = b$
11.        (* changing the delimiter is needed to stay primitive *)
12. $string = string + a^{n - length(string)}$
```

The algorithm above produces strings which have $O(n \log n)$ occurrences of k-th powers. Note, that if we perform the procedure the other way around, we only need $O(\log n)$ cycles and we can eliminate the recursion:

Theorem 2. *Algorithm ConstructStrings2 (see Fig. 3.2) produces a string of length n that has $\Omega(n \log n)$ occurrences of primitively rooted cubes.*

Proof. Before entering the second **while** loop, the length of *string* and the number of k-th power occurrences in it are both $c = (k+1) \cdot \ell + k$. Now we will show by induction on i that after the i-th iteration of the second **while** loop the length of *string* will be $(k+1)^i \cdot (c+1) - 1$ and the number of occurrences of k-th powers will be $(k+1)^i \cdot c + i \cdot (k+1)^{i-1}(c+1)$.

Note that if the length of *string* was m and the number of k-th power occurrences was p after the previous cycle, then concatenating $k+1$ copies of *string* delimited by k copies of *delimiter* we get $(k+1) \cdot p + m + 1$ powers in the new *string*, which will have length $(k+1) \cdot m + k$. Therefore, after the first cycle the length of *string* will be

$$(k+1) \cdot c + k = (k+1) \cdot c + (k+1) - 1 = (k+1)^1 \cdot (c+1) - 1$$

At the same time the number of k-th powers will be

$$(k+1) \cdot c + c + 1 = (k+1)^1 \cdot c + 1 \cdot (k+1)^0 \cdot (c+1)$$

so our statement holds for $i = 1$. Now suppose it is true for some $i \geq 1$. From here we get that for $i + 1$ the length of *string* will be:

$$(k+1) \cdot ((k+1)^i \cdot (c+1) - 1) + k = (k+1)^{i+1} \cdot (c+1) - 1$$

whereas the number of k-th powers is:

$$(k+1) \cdot ((k+1)^i \cdot c + i \cdot (k+1)^{i-1} \cdot (c+1)) + ((k+1)^i \cdot (c+1) - 1) + 1$$

$$= (k+1)^{i+1} \cdot c + i \cdot (k+1)^i \cdot (c+1) + (k+1)^i \cdot (c+1)$$

$$= (k+1)^{i+1} \cdot c + (i+1) \cdot (k+1)^i (c+1)$$

Now let us look at the running time of the algorithm. In the first **while** loop we divide the actual length by $k+1$ and we do it until it becomes smaller than k^3+k^2+k therefore we perform $O(\log n)$ cycles. The second **while** loop has the same number of cycles, with one string concatenation performed in each cycle, hence substituting $\log n$ for i in the formula above concludes the proof.

□

Corollary 2. *In a string of length n the maximal number of primitively rooted k-th powers, for a given integer $k \geq 2$, is $\Theta(n \log n)$.*

Proof. We know from [10] that the maximal number of occurrences of primitively rooted squares in a word of length n is $O(n \log n)$. This implies that the number of primitively rooted greater integer powers also have an $O(n \log n)$ upper bound, while in Theorem 2 we showed the lower bound $\Omega(n \log n)$.

□

Remark 1. *The first part of the proof is directly applicable to runs so we have that in a string of length n the number of runs of length at least $3p - 1$, where p is the (smallest) period of the run is at most $n - 2$. Unfortunately we cannot apply the proof directly for runs shorter than that because we need the same string on both sides of the central Lyndon position.*

We have seen that the number of occurrences of k-th powers for a given k(≥ 2) in a string of length n is $\Theta(n \log n)$, but what happens if we sum up the occurrences of k-th powers for all $k \geq 2$?

Remark 2. *The upper bound of the sum of all occurrences of k-th powers with primitive root, where $k \geq 2$, in a word w with $|w| = n$ is $\frac{n\cdot(n-1)}{2}$. Moreover, the bound is sharp.*

Proof. First consider the word a^n, for some $n > 0$. Clearly, taking any substring a^k, with $2 \leq k \leq n$, we get a k-th power, so the number of powers greater or equal to two is given by the number of contiguous substrings of length at least two, that is $\frac{n\cdot(n-1)}{2}$. Now we will show that this is the upper bound. Let us suppose that any two positions i and j in the string delimit a k-th power with $k \geq 2$, just like in the example above. We need to prove that the same string cannot be considered a k_1-th power and a k_2-th power at the same time, with $k_1, k_2 \geq 2$ and $k_1 \neq k_2$. Suppose the contrary, that is there are $1 \leq m < \ell \leq \frac{j-i}{2}$ so that both m and ℓ are periods of $w[i,j]$. Since $j - i > m + \ell - gcd(m,\ell)$ the periodicity lemma tells us that $w[i,j]$ has a period p smaller than m with $p|m$ and $p|\ell$, and this, in turn, means $w[i, i+\ell]$ is not primitive. □

Let us turn now to counting distinct powers in strings. As opposed to before, now we are not concerned about how many copies a repetition has in a given string but rather about how many pairwise different repetitions can show up. The problem is has a quite different taste to it than counting all the occurrences. Here, the strings having the most distinct powers are not as repetitive in some sense as the ones having the most occurrences of powers. Also the way of proving bounds with respect to this is different in that for every repetition we usually have to designate a particular occurrence of it in the string that we will consider. In other words, what we can do here is to show some restrictions on the overlap of the last occurrences of some repetitions. The question was first considered by Fraenkel and Simpson in [21], where they considered the maximal number of distinct squares.

Theorem 3. *[21, 28] The number of distinct squares in a word of length n is at most $2n$.*

Their proof uses intricate combinatorics to achieve the desired bound and they conjectured, supported by computer verification, that a sharp upper bound would be no greater than n. Later on, Ilie presented a simpler proof for the $2n$ bound in [28]. For a while there has been no progress on the issue and then Ilie improved the bound by a logarithmic term.

Theorem 4. *[29] The maximal number of distinct squares in a word of length n is $2n - O(\log n)$.*

Although his result does not get very close to the conjectured bound, the methods he uses may prove very useful in further developments. In particular the following lemma suggests that maybe it is possible to relate the upper bounds on the number of runs in a string to the number of distinct squares.

Lemma 4. *[29] If the last occurrences of some squares u^2, v^2, with $|u| < |v|$ start at position i, and the last occurrence of another square w^2 starts at position $i+1$, then either $|w| \in \{|u|, |v|\}$ or $|w| > 2|u|$.*

Here we are not going to consider distinct squares but rather look at distinct k-th powers with k greater than two. The earlier mentioned results give an implicit upper bound for greater powers as well. On the other hand, it may be the case that we can show significantly smaller upper bounds for cubes, 4-th powers and so on. Indeed it is the case as the next theorem illustrates.

Theorem 5. *The number of distinct k-th powers, for a fixed integer $k \geq 3$, in a string of length n is at most $\frac{n}{k-2}$.*

Proof. We will show the upper bound by considering the last occurrences of every k-th power. The proof is split into two parts. We will prove the statement for cubes by considering their starting positions while for higher exponents we will look at their root positions (see below).

Let us start with the case $k = 3$. Suppose the last occurrence of two different cubes u^3 and v^3 with $|u| < |v|$ start at the same position i in the string. By a simple argument we will arrive at a contradiction by looking at the two cases shown in Figure 3.5.

First let us look at the case when $|u^3| \leq |v^2|$. In this case there is another occurrence of u^3 starting at position $i + |v|$ contradicting our assumption of the

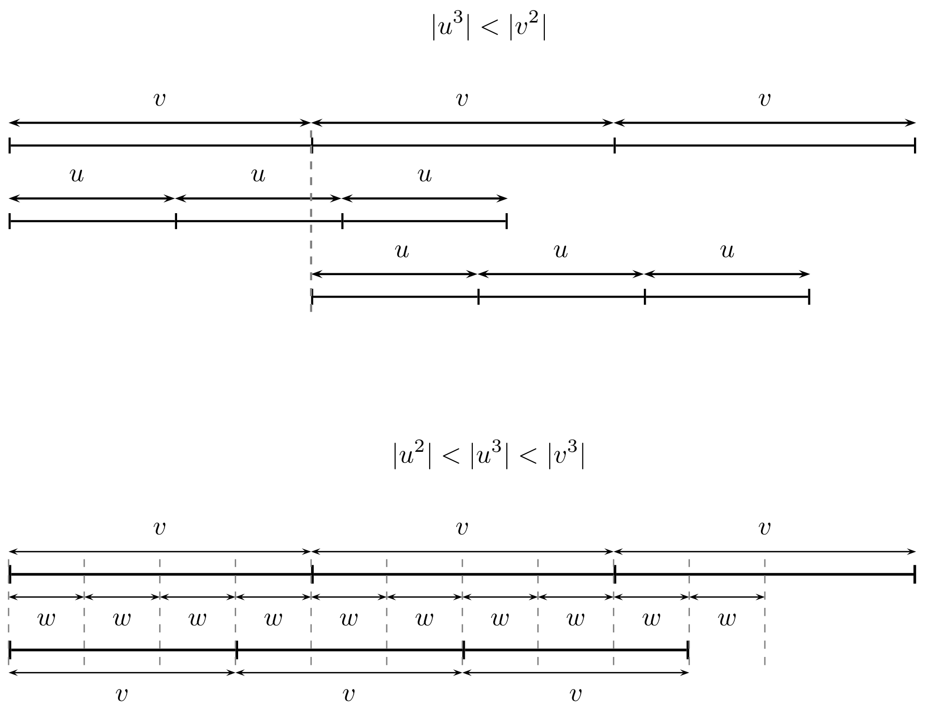

Figure 3.5: Cubes u^3 and v^3 beginning at the same position i.

previous occurrence being the last.

Now we are left to treat the case when $|v^2| < |u^3| < |v^3|$. The overlap between the two cubes in this case is at least $2 \cdot |v|$ which is greater than $|v| + |u|$ and from this, Fine and Wilf's theorem tells us it has a period of length $gcd(|u|, |v|)$. Therefore, there exists some w such that $u = w^m$ and $v = w^n$, for some integers $m < n$. It is easy to see then that in $|w^{3n}|$ the last occurrence of $|w^{3m}|$ starts at position $i + 3 \cdot (m - n) \cdot |w|$ contradicting our assumption again.

We showed that there can be no two different cubes which have their last occurrence starting at the same position. This implies the bound n for higher distinct powers as well. However, we can prove something stronger, as we claim in the theorem. To achieve that result, we will look at *r*oot positions. In a power u^k starting at position i, with the smallest period of u being p, we will call position $i + p$ the second root position, $i + 2p$ the third root position and so on. We show that for the last occurrences of two 4-th powers u^4 and v^4, with $|u| < |v|$, u^4 starting at position i and v^4 starting at position j, the following positions cannot coincide:

1. the second root position of u and the second root position of v:

if $3|u| < 2|v|$ then u^4 occurs at $i + |v|$, contradiction;

if $2|v| \leq 3|u|$ then according to Fine and Wilf's theorem v has period $k \cdot p$, for some k and then u^4 occurs at $i + p$, contradiction.

2. the second root position of u and the third root position of v:

   if $3|u| \leq |v|$ then u^4 occurs at $i + |v|$, contradiction;

   if $|v| < 3|u| \leq 2|v|$ then again Fine and Wilf's theorem gives us u^4 occurring at $i + p$, contradiction;

   if $2|v| < 3|u|$: if this is the case then similarly as before u and v are powers of the same word and hence v^4 occurs at $j + p$, contradiction;

3. the second root position of v and the third root position of u:

   if $3|u| < |v|$ then u^4 occurs at $i + |v|$, contradiction;

   if $|v| \leq 3|u|$ then u^4 occurs at $i + p$, contradiction.

4. the third root position of u and the third root position of v:

   if $2|u| \leq |v|$ then u^4 occurs at $i + |v|$, contradiction;

   if $|v| < 2|u|$ then u^4 occurs at $i + p$.

We can apply the same argument for 5-th powers looking at the second, third and fourth root positions and so on for greater powers as well, getting the desired bound. □

3.3 Conclusion

In conclusion, we have proven the following bounds on repetitions in words:

(i) The maximal number of distinct repetitions of exponent, e, with $1 < e < 2$, in a word of length n is $\Theta(n^2)$.

(ii) The maximal number of primitively rooted k-th powers in a word of length n is $\Omega(n \log n)$.

We have also described an $O(m \log n)$ algorithm which can be used to construct strings to illustrate these bounds. Here $O(m)$ is the time complexity of concatenating two strings of length n.

Chapter 4

Counting scattered subwords

A widely investigated topic in combinatorics on words and formal language theory is that of primitive words. From a formal language point of view most studies are related to languages consisting of only primitive words or just the opposite, languages made up of powers. We take a look at the language of words that stay primitive even when considering their commutative closure. In particular we give some criteria based on the number of scattered subword occurrences which are sufficient but not necessary for a word to be primitive starting from the simplest subwords and going on to formulate inequalities between subword occurrence multiplicities that need hold in order for the containing word to be non-primitive. We consider another interpretation of powers when the building blocks do not need to be exactly the same but rather need to have the same Parikh vector. We show that in the limit this property is equivalent to the classical notion of non-primitivity from the point of view of the number of scattered subword occurrences. After this we move on to investigating how much information about scattered subwords suffices to identify the word uniquely. Eventually, closing the chapter, in Section 4.6 we discuss some cases of an open problem regarding general inequalities between scattered subword multiplicities. In [46] the authors define the so called subword histories, which are formal power series representing the number of subword occurrences. They settle the question of decidability of equalities between these polynomials and ask whether there exists an algorithm to decide whether an inequality of this kind holds for all words over an alphabet. We approach the problem in an incremental way by first showing that only trivial inequalities between monomial subword histories hold and then characterising subword inequalities with the smaller side consisting of one and the greater side consisting of two terms (a $(1, 2)$ instance of subword inequalities). The latter may give rise to the decidability of a wide range of subword inequalities decomposable into simple $(1, 2)$ instances.

4.1 Parikh matrices and subword histories

A general problem regarding subword multiplicities is to reconstruct a word w if we know the values $|w|_u$ for some u. For instance over the alphabet $\{a, b\}^*$ a word w is uniquely determined by the values $|w|_a = |w|_b = 4$ and $|w|_{ab} = 15$. The word in question is a^3bab^3. On the other hand a word of length 4 is not uniquely determined by the values $|w|_u$, $|u| \leq 2$. Both *abba* and *baab* satisfy the equations. It is shown in the chapter on subwords in [41] that the equation

$$|vb|_{ua} = |v|_{ua} + \delta_{a,b}|v|_u$$

where

$$\delta_{i,j} = \begin{cases} 1 & \text{when } a = b \\ 0 & \text{when } a \neq b \end{cases}$$

together with the already mentioned

$$|w|_\lambda = 1 \text{ and } |w|_u = 0 \text{ if } |w| < |u|$$

suffice to compute all values $|w|_u$. As we saw in Chapter 2, Parikh vectors tell us how many times the letters of the alphabet occur in a word. Sometimes, however, we need more information about structure. A powerful yet simple tool is provided by an extension of the Parikh mapping. In [45] the Parikh matrix is introduced, which extends the notion of the Parikh vector. For an alphabet of n letters the Parikh matrix of a word is an $(n + 1) \times (n + 1)$ upper triangular matrix that contains information about those scattered subwords of the word that are factors of the concatenation of the alphabet.

Definition 2. *Let $\Sigma = \{a_1, ..., a_k\}$ be an alphabet and M_{k+1} denote the monoid of* *square matrices of dimension $k+1$ with matrix multiplication. The Parikh matrix mapping, denoted Ψ_k, is the morphism: $\Psi_k : \Sigma^* \rightarrow M_{k+1}$, defined by the condition: if $\Psi_k(a_q) = (m_{i,j}), 1 \leq i,j \leq (k+1)$, then for each $1 \leq i \leq (k+1), m_{i,i} = 1, m_{q,q+1} = 1$, all other elements of the matrix $\Psi_k(a_q)$ being 0.*

It is a morphism, i.e. $\Psi_k(a_1a_2\ldots a_n) = \Psi_k(a_1) \times \Psi_k(a_2) \times \ldots \times \Psi_k(a_n)$, where $\times$ is the usual matrix multiplication. The Parikh matrix mapping is further extended in [57]:

Definition 3. *Let Σ be an alphabet and $u = b_1 ... b_{|u|}$ be a word in Σ^* ($b_i \in \Sigma$ for all $1 \leq i \leq |u|$). The Parikh matrix mapping induced by the word u over the alphabet, denoted Ψ_u, is the morphism $\Psi_u : \Sigma^* \rightarrow M_{|u|+1}$ defined by the condition: if $a \in \Sigma$ and $\Psi_u(a) = (m_{i,j})_{1\leq i,j \leq |u|+1}$, then:*

$$m_{i,j} = \begin{cases} 1 & if \quad j = i \\ \delta_{a,b_j} & if \quad j = i+1 \\ 0 & otherwise \end{cases},$$

where δ is the Kronecker symbol defined above.

The following is an important property [46, 57] of these mappings that we will use later:

Lemma 5. *Consider $u = b_1 ... b_{|u|} \in \Sigma^*$ ($b_i \in \Sigma$ for all $1 \leq i \leq |u|$) and $w \in \Sigma^*$. The matrix $\Psi_u(w) = (m_{i,j})_{1\leq i,j\leq |u|+1}$, has the following properties:*

1. *$m_{i,j} = 0$, for all $1 \leq j < i \leq |u|+1$,*
2. *$m_{i,i} = 1$, for all $1 \leq i \leq |u|+1$,*
3. *$m_{i,j+1} = |w|_{u_{i,j}}$, for all $1 \leq i \leq j \leq |u|$, where $u_{i,j}$ is the word $b_ib_{i+1}...b_j$.*

This means that for any word w and arbitrary letters $x_1, x_2, \ldots, x_n$ the generalised Parikh matrix $\Psi_{x_1x_2\ldots x_n}$ holds the following values

$$\Psi_{x_1x_2\ldots x_n}(w) = \begin{pmatrix} 1 & |w|_{x_1} & |w|_{x_1x_2} & \ldots & |w|_{x_1x_2\ldots x_{n-1}} & |w|_{x_1x_2\ldots x_n} \\ 0 & 1 & |w|_{x_2} & \ldots & |w|_{x_2\ldots x_{n-1}} & |w|_{x_2\ldots x_n} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & |w|_{x_n} \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix}$$

The value of each minor of an arbitrary Parikh matrix is a nonnegative integer [46]. From here one can obtain the so called Cauchy inequality for words and its dual.

Theorem 6. *[46, 54] The inequality*

$$|w|_y |w|_{xyz} \leq |w|_{xy} |w|_{yz}$$

holds for arbitrary words w, x, y and z.

Theorem 7. *[54] For all words u, v, z and w:*

$$|uvz|_w |v|_w \leq |uv|_w |vz|_w.$$

4.2 Primitivity in the Context of Multisets

In this section we are going to look into languages where just by looking at the Parikh vectors of the language we can decide if it is made up of primitive words or powers greater than or equal to two. As the Parikh vector does not tell us anything about the order of the letters, our investigation is limited to so called commutative languages here.

Definition 4. *The commutative closure of a language L consists of all words that can be formed by permuting the letters in the words of L:*

$$com(L) = \{u | \psi(u) = \psi(w), w \in L\}.$$

We call a language commutative if it is equal to its commutative closure.

Definition 5. *The language of primitive words over an alphabet Σ (Q_Σ) consists* *of all such words:*

$$Q_\Sigma = \{w \in \Sigma^* | w = u^p \Rightarrow p = 1\}.$$

In the following by the cyclic permutation of a word w we mean the set $cp(w)$ consisting of w's conjugates.

Definition 6. *The language of Lyndon words has the lexicographically smallest word among the cyclic permutations of* w*, for all primitive words* w *.*

$$L = \{w | w \in Q \text{ and } w \leq u, \forall u \in cp(w)\}.$$

First, we analyze the language of words for which every permutation is primitive, in other words the maximal commutative language containing only primitive words.

Definition 7. *The commutative primitive language is the commutative language which consists of all primitive words over* Σ *whose Parikh equivalent words are primitive. We will denote by* $ComPrim_\Sigma$ *the commutative primitive language over an alphabet* Σ*.*

We might omit the subscript Σ when it does not create confusion or if a statement is valid for any alphabet. We require that each word that can be formed of the letters of any word in $ComPrim$ to be primitive. We can ensure this by the following:

Proposition 9. *[13] A word is in* $ComPrim_\Sigma$ *for some alphabet* Σ *if and only if the elements of its Parikh-vector are relatively prime.*

Now let us see what happens with the commutative primitive languages, when a symbol is deleted from or added to the alphabet.

Remark 3. *Let* $\Sigma = \{a_1, .., a_k\}$ *and* $\Sigma' = \{a_1, .., a_k, a_{k+1}\}$ *be some ordered alphabets. If* $w \in ComPrim_\Sigma$*, with Parikh-vector* $\psi(w) = (n_1, .., n_k)$ *then* $w' \in ComPrim_{\Sigma'}$*, with* $\psi(w') = (n_1, .., n_k, n_{k+1})$*, for every value of* n_{k+1}*. However, the converse does not hold, i.e. if we delete a letter from* Σ'*, then the scattered version of the word which was previously in* $ComPrim_{\Sigma'}$ *will not necessarily be in* $ComPrim_\Sigma$*.*

Proof. The first part of the statement trivially holds because for any number n_{k+1} if $gcd(n_1, .., n_k) = 1$, then $gcd(n_1, .., n_k, n_{k+1}) = 1$, where gcd means the greatest common divisor. The second part can be easily handled too. Let us take w' s.t. $\psi(w') = (n_1, .., n_k, n_{k+1})$, where $gcd(n_1, .., n_k) > 1$ but $gcd(n_1, n_{k+1}) = 1$, and thus $w' \in ComPrim_{\Sigma'}$. Clearly, removing a_{k+1} from the alphabet, i.e. deleting the occurrences of this symbol from w' leads us to a word w, with $\psi(w) = (n_1, .., n_k)$, which by Lemma 9 does not belong to $ComPrim_\Sigma$, e.g. in the case of an alphabet $\{a, b, c\}$ taking the Parikh-vector $(6, 10, 15)$. □

Moreover, if we simply delete an occurrence of a letter from a word having Parikh-vector $(6, 10, 15)$, the resulting word is not in the language anymore.

From Proposition 9 it follows that the greatest common divisor of the numbers in the Parikh-vector of a non-primitive word is greater than 1. Thus it seems that both $ComPrim_\Sigma$ and its complement are non-semilinear languages. Indeed it is the case as it is shown in [13, 49]. We give here an alternative proof.

Theorem 8. *ComPrim is not semi-linear.*

Proof. We can restrict ourselves to the case of binary alphabets without loss of generality. Let us suppose that the commutative primitive language is semi-linear. It is well known (see e.g. [35]) that for every semi-linear language there is a regular one which is letter-equivalent to it. So, due to our assumption there is a regular language L such that for any relatively prime q and r there exists a word $w \in L$ for which $\psi(w) = (q, r)$. Take two different primes $p_1 > n, p_2 > (n+1)p_1 + 1$, where n is the number of states of the minimal automaton accepting L. Then there is a word $b^{x_1}ab^{x_2}a...ab^{x_{p_1+1}} \in L$ with $x_1, ..., x_{p_1+1} \geq 0, \sum_{i=1}^{p_1+1} x_i = p_2$. Then there is at least one x_i that is bigger than n. Now this means that there is a sequence of length $l \leq n$ consisting only of b's that can be pumped arbitrarily many times so that the resulting word will be in L. But since p_1 is prime and $n < p_1$ any such l will be a generator of the group of the numbers modulo p_1 with addition, hence pumping enough times will result in a word with Parikh vector $(p_1, kl + p_2)$ for some k so that $gcd(p_1, kl + p_2) > 1$, so we arrived to a contradiction.

□

It is well-known and it also follows directly from Proposition 9, that all prime-length words are primitive, except the words that are powers of a single letter. The commutative language containing the non-primitive words contains all words of the form a^x where $x \geq 2$, $a \in \Sigma$. The commutative closure of the language of primitive words is the same as the commutative closure of the language of Lyndon words ($com(Q_\Sigma) = com(Lyn)$), and it is exactly the regular language that is the complement of the commutative language containing only non-primitive words:

$$\Sigma^* \setminus \{a^m | m \geq 2, a \in \Sigma\}$$

Note, that the word a^2b^2 is in $com(Q_\Sigma)$, but not in $ComPrim_\Sigma$.

Hence the commutative language containing only non-primitive words and the commutative closure of the language of primitive words are semi-linear languages in the Parikh sense.

Now, let us look at the complements of the languages mentioned before. Let $\overline{Q_\Sigma}$ denote the complement of Q_Σ (for $|\Sigma| \geq 2$), i.e., the language of non-primitive words over an alphabet containing at least two letters. The Parikh image of $\overline{Q_\Sigma}$ is the same as the complement of the Parikh image of the commutative primitive language over Σ, so it is the set of all vectors whose elements are not relatively prime. So as a consequence of Theorem 8

Proposition 10. *The language $\overline{Q_\Sigma}$ ($|\Sigma| \geq 2$) is not semi-linear.*

Proof. We get this result, because the class of semi-linear languages is closed under multiset complementation ([35]), and as we proved earlier in Theorem 8, the commutative primitive language is not semi-linear. □

As it is well-known that the family of context-free languages are semi-linear, Proposition 10 provides yet another proof for the language of non-primitive words to be outside the context-free class.

Proposition 11. *[13] The language $\overline{Q_\Sigma}$ ($|\Sigma| > 2$) is not context-free.*

4.3 Counting scattered subwords

In [46] the authors describe a nice extension of the Parikh mapping to one that carries more information about the words. They start out by the simple observation that $|w|_a|w|_b = |w|_{ab} + |w|_{ba}$ and they further develop it into subword histories. Starting from the equality above we can state some simply verifiable requirements for words to be primitive.

Lemma 6. *For any non-primitive word w and all words $u = a_1a_2..a_n$, with $a_i \neq a_j, \forall i \neq j, 1 \leq i < j \leq n$ and $|w|_u > 0$, there exists a set P consisting of permutations of u such that $|P| \geq 2^n - n$ and for all $p \in P$, $|w|_p > 0$.*

Proof. Since the letters of u are all different, some permutation u' of u must be present as a scattered subword in the primitive root $\sqrt{w}$ of w. As w is at least the second power of $\sqrt{w}$, we have at least two non-overlapping occurrences of u' in w. Hence by taking k letters from the first occurrence of u' in all possible ways and the remaining $n-k$ letters from its second occurrence we get $\binom{n}{k}$ permutations of u' different from each other that are scattered subwords of w. Note that by taking the first k letters from the first occurrence we get u' itself, so when summing all the possible combinations for $0 \leq k \leq n$, the identical permutation, i.e. u' is counted $n+1$ times. Hence the least (depending on the word there may be more) number of permutations of u' that are different from each other and are scattered subwords of w is:

$$\binom{n}{0} + \binom{n}{1} + ... + \binom{n}{n} - n = 2^n - n.$$

By taking the set of all these permutations to be P we get the statement above. □

Note, that in the previous proof in fact we only needed w to have a factor which is a second power of some word, thus the statement can be more generally stated for all words of the form $w = xvvy$, where v is not the empty word.

The fact that both Parikh-matrices and Lyndon words look like they require smaller letters (w.r.t. the ordering of the alphabet) to be at the beginning of the word suggests that there is some kind of a connection between the minimal words (w.r.t. the ordering of the alphabet) and the words with the maximal Parikh-matrix (maximal in the sum of the elements of the matrix). However, as we will shortly see we were not successful in establishing a general relation between them. But first let us state a condition for the maximality of the Parikh-matrix of w among its cyclic permutations in the case of binary alphabets.

Remark 4. *The Parikh-matrix of any word $w \in \{a,b\}^*$ is maximal ($a < b$) among its cyclic permutations if and only if for any decomposition $w = uv$:*

$$|u|_a|v|_b \geq |u|_b|v|_a.$$

The validity of the following examples can be easily seen with the help of Remark 4. For any word containing at most five blocks, i.e. of the forms $a^*b^*a^*b^*a^*$, $b^*a^*b^*a^*b^*$, for every word w that has the maximal $(a < b)$ Parikh-matrix among its cyclic permutations either w $(a < b)$ or its reverse $(b < a)$ is a Lyndon word. For words having more blocks than 5 the statement is not true; it is contradicted by, say a^3ba^4bab, as Lyndon words depend on the prefix blocks of the word rather than the value of the Parikh-matrix.

Although the above mentioned attempt did not lead to a direct relation between Parikh-matrices and Lyndon words, we can say the following:

Proposition 12. *For all words $w = a_1...a_n, w_r = a_na_1...a_{n-1}, w_l = a_2...a_na_1$ over a binary alphabet the minimal among the three (w.r.t. the ordering of the alphabet) has the maximal Parikh-matrix.*

For alphabets having more than two letters the statement does not hold, e.g. in the case of the word *accb* (which is lexicographically the smallest among its conjugates) the sum of the elements in its Parikh matrix is 9 whereas for its lexicographically greater conjugate *bacc* this sum is 10.

In [44] the authors consider the problem of deciding about a matrix whether or not there exists a word which has this matrix as its extended Parikh image. Besides the formal requirements of Parikh matrices - that is, they need to be upper-triangular and the main diagonal has to consist of 1's - the values are restricted by subword histories. For instance the simplest and most frequently used subword history relation is $|w|_a|w|_b = |w|_{ab} + |w|_{ba}$, for some word w and letters $a \neq b$ (the distinction is necessary since $|w|_{aa} = \binom{|w|_a}{2}$). This equality clearly restricts the values of the third diagonal in terms of the second. Above that, i.e. about the values of the fourth, fifth, etc. diagonals we cannot say much. In fact it is shown that they can be almost arbitrary positive integers. [44] also provides a polynomial time algorithm to check if there is any word the image of which is the given matrix. We will see in this section that things are easier when we have non-primitive words instead of arbitrary ones. Moreover, the results presented here apply to the extended Parikh matrices too, introduced in [57].

The main result of this section is the following theorem:

Theorem 9. *For all words $w, u \in \Sigma^+, u = a_1 a_2 ... a_k$ and $n, k > 1$:*

$$|w^n|_u \leq \left(\frac{n+1}{2n}\right)^{k-1} |w^n|_{a_1} |w^n|_{a_2} ... |w^n|_{a_k}.$$

To prove the theorem we need to see a few other statements first. First of all, counting the number of subword occurences in a non-primitive word should be possible by knowing only its root, and indeed it is.

Lemma 7. *For all words $w, u \in \Sigma^*, u = a_1 a_2 ... a_k$:*

$$|w^n|_u = \sum_{i=1}^{min(|u|,n)} \left(\binom{n}{i} \sum_{u=u_1...u_i} \prod_{j=1}^{i} |w|_{u_j} \right)$$

where all u_j are non-empty factors of u.

Proof. It is easy to see that the number of occurrences of a word u in some power of w can be computed by factoring u in all possible ways and for each factorization multiplying the number of occurrences of these factors in w. □

For Theorem 9 we need to establish a relation between the multiplicities of different permutations of the same word.

Lemma 8. *For all words $w^n \in \Sigma^+$ and $a, b \in \Sigma$, with $n > 1$ and $|w^n|_a, |w^n|_b > 0$ we have:*

$$\frac{n-1}{n+1} \leq \frac{|w^n|_{ab}}{|w^n|_{ba}} \leq \frac{n+1}{n-1}.$$

Proof. For $a = b$ the value of the fraction is 1 so the inequality holds. Hence we may assume that $a \neq b$. By Lemma 7 we have a formula for $|w^n|_{ab}$ in terms of n and $|w|_{ab}$. If we express $|w|_{ab}$ from this formula and use basic equalities from subword histories, we get:

$$\begin{aligned} |w|_{ab} &= \frac{2n|w^n|_{ab} - (n-1)|w^n|_a |w^n|_b}{2n^2} \\ &= \frac{2n|w^n|_{ab} - (n-1)(|w^n|_{ab} + |w^n|_{ba})}{2n^2} \\ &= \frac{(n+1)|w^n|_{ab} - (n-1)|w^n|_{ba}}{2n^2} \qquad (4.1) \end{aligned}$$

Since $|w|_a|w|_b = |w|_{ab} + |w|_{ba}$, we also know that

$$|w|_{ab} \leq |w|_a|w|_b = \frac{|w^n|_a}{n}\frac{|w^n|_b}{n} = \frac{|w^n|_{ab} + |w^n|_{ba}}{n^2} \tag{4.2}$$

From equations (4.1) and (4.2) it follows that:

$$(n+1)|w^n|_{ab} - (n-1)|w^n|_{ba} \leq 2(|w^n|_{ab} + |w^n|_{ba})$$

$$\frac{n-1}{n+1} \leq \frac{|w^n|_{ba}}{|w^n|_{ab}}.$$

If we follow the same procedure for $|w|_{ba}$ we get the other bound too. □

In particular taking $n = 2$ this means that for a word w if $\frac{|w|_{ab}}{|w|_{ba}}$ is less than $\frac{1}{3}$ or more than 3 then w is primitive. Note that because of Lemma 6, if the conditions $|w|_{ab}, |w|_{ba} > 0$ are not satisfied, we can state directly that w is a primitive word.

Theorem 10. *In Lemma 8:*

1. *the bounds are sharp, i.e. the equality is reached if and only if*

$$pr_{\{a,b\}}(w) \in a_1^+a_2^+, a_1, a_2 \in \{a, b\}, a_1 \neq a_2$$

2. *the ratio is equal to* 1 *if* $pr_{\{a,b\}}(w)$ *is a palindrome but the converse does not hold.*

where $pr_\Sigma(w)$ *is the projection of* w *on the alphabet* Σ.

Proof. 1. the if part is fairly easy to show. We can get the number $|w^n|_{a_1a_2}$ by summing up for all occurrences of a_1 the a_2's that follow them. So if $\psi(w) = (n_1, n_2)$ then this number will be $n_1(n_2n) + n_1(n_2(n-1)) + .. + n_1n_2 = n_1n_2\frac{n(n+1)}{2}$ whereas for $|w^n|_{a_2a_1}$ we get $n_2(n_1(n-1)) + n_2(n_1(n-2)) + .. + n_2n_1 = n_2n_1\frac{(n-1)n}{2}$. Dividing one with the other results in the bounds of Lemma 8. For the only if part we may assume that $\frac{|w^n|_{a_1a_2}}{|w^n|_{a_2a_1}} = \frac{n+1}{n-1}$. Lemma 7 tells us that

$$\frac{|w^n|_{a_1a_2}}{|w^n|_{a_2a_1}} = \frac{n|w|_{a_1a_2} + \binom{n}{2}|w|_{a_1}|w|_{a_2}}{n|w|_{a_2a_1} + \binom{n}{2}|w|_{a_1}|w|_{a_2}} = \frac{2|w|_{a_1a_2} + (n-1)(|w|_{a_1a_2} + |w|_{a_2a_1})}{2|w|_{a_2a_1} + (n-1)(|w|_{a_1a_2} + |w|_{a_2a_1})}.$$

Using the initial assumption $\frac{|w^n|_{a_1a_2}}{|w^n|_{a_2a_1}} = \frac{n+1}{n-1}$ we get that

$$(n+1)^2|w|_{a_2a_1} + (n^2-1)|w|_{a_1a_2} = (n^2-1)|w|_{a_1a_2} + (n-1)^2|w|_{a_2a_1}$$

and after short computation this leads us to $|w|_{a_2a_1} = 0$ so we can conclude the proof. For the other bound the argumentation goes along the same line.
2. w being a palindrome means w is equal to its reverse w^r. We can easily see that for all words u, v the relation $|u|_v = |u^r|_{v^r}$ holds, so $|w|_{a_1a_2} = |w^r|_{a_2a_1} = |w|_{a_2a_1}$ and this proves our claim. The converse does not hold in general; take for instance the word $w = abaaabbaa$ which is not a palindrome, but $|w^n|_{ab} = |w^n|_{ba}$ holds for any n. □

Now we are ready to prove Theorem 9.

Proof. (of Theorem 9) If u is not a subword of w^n then the inequality is trivially satisfied. From Lemma 8 we know that for all pairs a_i, a_j:

$$\frac{n-1}{n+1} \leq \frac{|w^n|_{a_ia_j}}{|w^n|_{a_ja_i}} \leq \frac{n+1}{n-1}$$

and we have from subword histories that

$$|w|_a|w|_b = |w|_{ab} + |w|_{ba}.$$

So we get

$$\frac{|w^n|_{a_ia_j}}{|w^n|_{a_i}|w^n|_{a_j}} \leq \frac{|w^n|_{a_ia_j}}{|w^n|_{a_ia_j} + |w^n|_{a_ja_i}} \leq \frac{n+1}{2n} \Rightarrow$$

$$\Rightarrow |w^n|_{a_ia_j} \leq \frac{n+1}{2n}|w^n|_{a_i}|w^n|_{a_j} \tag{4.3}$$

At the same time from the general form of Theorem 6 we have:

$$|w^n|_{a_2}|w^n|_{a_3}...|w^n|_{a_{k-1}}|w^n|_{a_1a_2...a_k} \leq |w^n|_{a_1a_2}|w^n|_{a_2a_3}...|w^n|_{a_{k-1}a_k} \tag{4.4}$$

Thus equations (4.3) and (4.4) yield:

$$|w^n|_{a_2}...|w^n|_{a_{k-1}}|w^n|_{a_1a_2...a_k} \leq \left(\frac{n+1}{2n}\right)^{k-1}|w^n|_{a_1}|w^n|^2_{a_2}...|w^n|^2_{a_{k-1}}|w^n|_{a_k}.$$

Dividing both sides by $|w^n|_{a_2}...|w^n|_{a_{k-1}}$ reduces the inequality to the desired form.

□

Note that the bounds are sharp, that is the inequality becomes an equality exactly in the cases of Theorem 10. As we have seen, the bound for the difference between the number of occurrences of a word u and some permutation of it in some non-primitive word w^n decreases as n increases. At the same time - as it is to be expected intuitively - as n grows, the ratio between the number of occurrences of some u in w^n and in w^{n+k} decreases for a fixed k.

Proposition 13. *For all words w, u, numbers $n_1 \leq n_2$ and k such that $|w^{n_1+k}|_u \geq 0$ we have:*

$$\frac{|w^{n_1}|_u}{|w^{n_1+k}|_u} \leq \frac{|w^{n_2}|_u}{|w^{n_2+k}|_u}.$$

Proof. From the dual of the Cauchy inequality (Theorem 7) for words [46] we have that:

$$|w^{(n_2-n_1)}w^{n_1}w^k|_u|w^{n_1}|_u \leq |w^{n_2-n_1}w^{n_1}|_u|w^{n_1}w^k|_u.$$

i.e. $|w^{n_2+k}|_u|w^{n_1}|_u \leq |w^{n_2}|_u|w^{n_1+k}|_u$. If $|w^{n_1+k}|_u > 0$ we can order the factors to get the desired form. □

4.4 Subwords of the Thue-Morse word

Investigation concerning subword multiplicities of specific primitive words have been done in the past. Some of these results concerned factors of the Thue-Morse word. For the morphism h generating the Thue-Morse word $h(a) = ab$ and $h(b) = ba$ let $u_n = h^n(a), v_n = h^n(b)$. First we mention two earlier results from others that were achieved in this direction and then present our result concerning subword multiplicities of words from the concatenation power of some word sets. In [48] the following statement is shown:

Proposition 14. *[48] For every word x such that $0 \leq |x| \leq n, |u_n|_x = |v_n|_x$. Furthermore there exists a word x of length $n + 1$ such that $|u_n|_x \neq |v_n|_x$.*

In [2] a similar line of study is followed. Let $t[m]$ denote the prefix of length m of the Thue-Morse word. Then we get:

Proposition 15. *[2] For every prime number p, and for every positive integer n, there exists a positive integer $m = f(p, n)$ such that, for every non-empty word v of length less than or equal to n, $|t[m]|_v \equiv 0$ mod p.*

Our next theorem, although it is more general, has some implications for the scattered subword complexity of non-primitive words too. Namely if we take two sets of words S_1 and S_2 such that all the words in S_1 have the same Parikh vector and all the words in S_2 have the same Parikh vector then, as n increases, the ratio of the number of occurrences of some words w_1, w_2 of the same length in a word $u \in S_1^n$ and in a word $v \in S_2^n$, respectively, will tend towards a number that depends only on the Parikh-images of w_1, w_2, S_1, S_2.

Theorem 11. *For two sets $S_1 = \{u_1, u_2, ..., u_k\}, S_2 = \{v_1, v_2, ..., v_l\}$ with $u_i, v_j \in \Sigma^+$ such that $\psi(S_1) = \{\psi(u_1)\}, \psi(S_2) = \{\psi(v_1)\}$, for some strictly increasing function $g : \mathbb{N} \to \mathbb{N}$, for some functions $f_i : \mathbb{N} \to S_i^+$ with $f_1(n) \in S_1^{g(n)}, f_2(n) \in S_2^{g(n)}$ and for all words $w_1 = a_1a_2...a_r$ and $w_2 = b_1b_2...b_r$ such that $alph(w_1) = alph(w_2) = alph(u_1) = alph(v_1)$:*

$$\lim_{n\to\infty} \frac{|f_1(n)|_{w_1}}{|f_2(n)|_{w_2}} = \frac{|u_1|_{a_1}|u_1|_{a_2}...|u_1|_{a_r}}{|v_1|_{b_1}|v_1|_{b_2}...|v_1|_{b_r}}.$$

Proof. For some n let $f_1(n) = u_{i_1}u_{i_2}...u_{i_{g(n)}}, f_2(n) = v_{j_1}v_{j_2}...v_{j_{g(n)}}$. Now for any n such that $g(n) \geq r$

$$|f_1(n)|_{w_1} = \sum_{t=1}^{g(n)} |u_{i_t}|_{w_1} + \sum_{t_1=1}^{g(n)-1} \sum_{t_2=t_1+1}^{g(n)} \sum_{t_3=1}^{r-1} |u_{t_1}|_{a_1..a_{t_3}} |u_{t_2}|_{a_{t_3+1}...a_r} + ...+$$

$$\sum_{t_1=1}^{g(n)-r+1} \sum_{t_2=t_1+1}^{g(n)-r+2} ... \sum_{t_r=t_{r-1}+1}^{g(n)} |u_{t_1}|_{a_1}|u_{t_2}|_{a_2}...|u_{t_r}|_{a_r}.$$

The formula is similar for $f_2(n)$. This means that we take all possible decompositions of w_1 and all possible combinations of the words u_i in which they appear. Hence the lth term in the formula stands for the sum of decomposing w_1 in l parts $w_1 = w_1'w_2'...w_l'$, choosing l components of $f_1(n)$ and multiplying the number of appearances of w_i' in the ith component picked for all $1 \leq i \leq r$. Formally this lth term is:

$$\sum_{t_1=1}^{g(n)-r+1} \sum_{t_2=t_1+1}^{g(n)-r+2} \cdots \sum_{t_l=t_{l-1}+1}^{g(n)} \sum_{w_1=w_1'w_2'...w_l'} |u_{t_1}|_{w_1'}|u_{t_2}|_{w_2'}...|u_{t_l}|_{w_l'}.$$

Now let us define

$$Term_i(l) = \{|u_1|_{w_1'}|u_2|_{w_2'}...|u_l|_{w_l'} \mid w_i = w_1'w_2'...w_l', u_j \in S_i, 1 \le j \le l\}$$

as the set of the number of occurrences of w_i in words from the set S_i^n. Moreover let

$$fmax_i(n) = \sum_{j=1}^{r} \binom{n}{j} \max Term_i(j),$$

$$fmin_i(n) = \sum_{j=1}^{r} \binom{n}{j} \min Term_i(j).$$

Now it is clear that

$$\frac{fmin_1(g(n))}{fmax_2(g(n))} \le \frac{|f_1(n)|_{w_1}}{|f_2(n)|_{w_2}} \le \frac{fmax_1(g(n))}{fmin_2(g(n))}$$

for every n s.t. $g(n) \ge r$. Notice that $fmax_i$ and $fmin_i$ are polynomials in $g(n)$ of degree r with the terms of the highest degree in them being $\binom{n}{r} \max Term_i(r)$ and $\binom{n}{r} \min Term_i(r)$, respectively so the limit of the fractions above is given by the ratio of these terms. At the same time

$$\max Term_1(r) = \min Term_1(r) = \binom{n}{r} |u_1|_{a_1}|u_1|_{a_2}...|u_1|_{a_r}$$

$$\max Term_2(r) = \min Term_2(r) = \binom{n}{r} |v_1|_{b_1}|v_1|_{b_2}...|v_1|_{b_r}$$

so we get

$$\lim_{n\to\infty} \frac{fmin_1(g(n))}{fmax_2(g(n))} = \lim_{n\to\infty} \frac{fmax_1(g(n))}{fmin_2(g(n))} = \lim_{n\to\infty} \frac{|f_1(n)|_{w_1}}{|f_2(n)|_{w_2}} =$$

$$= \frac{|u_1|_{a_1}|u_1|_{a_2}...|u_1|_{a_r}}{|v_1|_{b_1}|v_1|_{b_2}...|v_1|_{b_r}}.$$

$\square$

The following two examples apply the result for a special kind of primitive words and for non-primitive words.

Example 3. *Let u_n, v_n be factors of the Thue-Morse word as defined at the beginning of this section. Then for any two words w_1, w_2 with $|w_1| = |w_2|$ over the alphabet $\{a, b\}$:*

$$\lim_{n\to\infty} \frac{|u_n|_{w_1}}{|v_n|_{w_2}} = 1.$$

This follows from Theorem 11 because $u_n, v_n \in \{01, 10\}^{2^n}$. Note that this example is also a direct consequence of Ochsenschlager's theorem (Proposition 14 above).

Example 4. *For any words u, v and $w = a_1a_2...a_k$ such that $alph(u) = alph(v) = alph(w)$:*

$$\lim_{n\to\infty} \frac{|u^n|_w}{|v^n|_w} = \frac{|u|_{a_1}|u|_{a_2}...|u|_{a_k}}{|v|_{a_1}|v|_{a_2}...|v|_{a_k}}$$

4.5 Inferring primitivity from partial information

It would be nice to know what subword multiplicities we have to know about a word w so that we can tell whether w is primitive. As it is known, there is no fixed set S of words so that knowing the multiplicities of the elements of S as scattered subwords of a word w would uniquely identify w. As we will see we can establish a similar result concerning primitive words, namely that there is no set S as before based on which one could decide the primitivity of any word. In what follows T_k means an alphabet of k letters. We recall from [55] the following:

Lemma 9. *[55] Assume that $S \subset \Sigma_k^*, k \geq 2$, is a set of words of a finite cardinality i, the longest word in S being of length j. Then there is a bound t_0 such that, whenever $t \geq t_0$, there are different words $w, w' \in \Sigma_k^*$ with $|w| = |w'| = t$ such that $|w|_u = |w'|_u$, for every $u \in S$.*

We can conclude a similar fact about primitive words.

Theorem 12. *Assume that $S \subset \Sigma_k^*, k \geq 2$, is a set of words of a finite cardinality. Then we can always find a primitive word z and a non-primitive one z' such that, $|z|_u = |z'|_u$, for every $u \in S \cup \Sigma$.*

Proof. Let w be the concatenation of all the words in S. Note that Lemma 9 is valid for any finite set, hence also for the set of factors of w, so there exist two different words v_1, v_2 which have the same Parikh matrix with respect to w. S is a subset of the set of w's factors, so for some words v_1, v_2, $\Psi_w(v_1) = \Psi_w(v_2)$ implies $|v_1|_u = |v_2|_u$, for every $u \in S$. From $\Psi_w(v_1) = \Psi_w(v_2)$ it follows that $\Psi_w(v_1^t v_2^t) = \Psi_w((v_1 v_2)^t)$. $(v_1 v_2)^t$ is obviously not primitive so now we only have to show that $v_1^t v_2^t$ is primitive. So let us suppose it is non-primitive (remember that $v_1 \neq v_2$). We can distinguish five cases depending on the length of the supposed primitive root of $v_1^t v_2^t$:

1. $|\sqrt{v_1^t v_2^t}| = 1$: in this case both v_1 and v_2 would be powers of the same letter, meaning $v_1 = v_2$, contradiction.

2. $|\sqrt{v_1^t v_2^t}| = 2$: in this case $v_1^2 = v_2^2$ hence $v_1 = v_2$, contradiction.

3. $|\sqrt{v_1^t v_2^t}| = t$: in this case we get directly $v_1 = v_2$, contradiction.

4. $|\sqrt{v_1^t v_2^t}| = 2t$: in this case $v_1^2 = v_2^2$ see 2.

5. $|\sqrt{v_1^t v_2^t}| = t^2$: in this case $v_1^t = v_2^t$ which means $v_1 = v_2$, contradiction.

This way we proved that $v_1^t v_2^t$ is primitive, so choosing $z = v_1^t v_2^t$ and $z' = (v_1 v_2)^t$ we can conclude the proof. □

We can see now that no fixed set is enough for deciding the primitivity of a word. However, as it is shown in [55] if the size of the set depends on the length of the word we want to infer, it is possible to reconstruct the word based on information about scattered subwords.

Theorem 13. *[55] Assume that w and w' are words over the alphabet $\{a, b\}$ with the same Parikh vector (r, s) and that*

$$|w|_{ab^i} = |w'|_{ab^i}, 1 \leq i \leq min(r, s)$$

Then $w = w'$.

However, a non-primitive word can probably be uniquely inferred from less information.

Conjecture 1. *Assume that w and w' are non-primitive words over the alphabet*

$\{a, b\}$ with the same Parikh vector (r, s) and that

$$|w|_{ab^i} = |w'|_{ab^i}, 1 \leq i \leq \frac{min(r,s)}{scd(r,s)}$$

where $scd(r, s)$ is the smallest common divisor of r and s which is greater than 1. Then $w = w'$.

4.6 Subword inequalities

The idea of considering subword histories stems from a simple observation as described in [46]:

$$|w|_{ab} + |w|_{ba} = |w|_a |w|_b$$

for any $w \in \Sigma^*$ and $a \neq b$. From here one can develop all sorts of useful relations between the count of subword occurrences.

Definition 8. *[46] Consider an alphabet Σ and a word $w \in \Sigma^*$. A subword history in Σ and its value in w are defined recursively as follows.*

- *Every $u \in \Sigma^*$ is a subword history in Σ, referred to as monomial, and its value in w equals $|w|_u$.*
- *Assume that SH_1 and SH_2 are subword histories with values α_1 and α_2, respectively. Then*

  $$-(SH_1),\ (SH_1) + (SH_2) \text{ and } (SH_1) \times (SH_2)$$

  are subword histories with values

  $$-\alpha_1,\ \alpha_1 + \alpha_2 \text{ and } \alpha_1\alpha_2,$$

  respectively.

Two subword histories are termed equivalent if they assume the same value in any w. A subword history is linear if it is obtained without using the operation $\times$. Perhaps the most basic question about two subword histories is whether they are equivalent or not. This question is settled by the authors in the same paper they define subword histories.

Theorem 14. *Every subword history is equivalent to a linear subword history. Moreover, given a subword history, an equivalent linear subword history can be effectively constructed.*

Probably the most natural question to ask after equivalence, where applicable, is about inequalities. We write $SH_1 \leq SH_2$ if, for all words w, the value of SH_1 in w is at most that of SH_2 in w. The problem posed in [46] is: is the inequality decidable for two given subword histories?

Theorem 14 tells us that it is enough to consider subword histories that are linear. From now on we will use the term *subword inequality* (SI) rather than the longer *inequality between subword histories*, and we mean basically the same, except for the coefficients of the terms. A SI is of the form:

$$\sum_{i=1}^{m} \alpha_i |w|_{u_i} \leq \sum_{j=1}^{n} \beta_j |w|_{v_j}$$

where the α's and β's are positive integers, the *coefficient*s of the terms. For the sake of simplicity we will write the above SI as $\sum_{i=1}^{m} \alpha_i u_i \leq \sum_{j=1}^{n} \beta_j v_j$.

In [46] the authors give an example of a SI which is true for any word:

$$baab < bab + baaab$$

It turns out that this example encompasses the very essence of the problem. In fact, all SIs that are "extended" versions of the one above hold for any word. We will elaborate in this section on what extended in the previous sentence exactly means. In addition to the notions introduced in the opening chapter and the first section of this chapter, we will often refer to the number of blocks of a given word. A continuous subword z of the word w is a *block* of w if $z = a^k$, for some $a \in \Sigma, k > 0$, and there are no words u, v such that $w = uazv$ or $w = uzav$. In other words, blocks are maximal continuous subwords that are powers of one letter. If $w = a_1^{k_1} a_2^{k_2} ... a_n^{k_n}$ with $a_i \neq a_{i+1}$ for $1 \leq i \leq n-1$, we will call $red(w) = a_1 a_2 ... a_n$ the *reduced form* of w, and $pow(w) = (k_1, k_2, .., k_n)$ the *power vector* of w. First we examine the inequalities where both sides comprise exactly one term.

Theorem 15. *For any two words $u, v \in \Sigma^*$ with $u \neq v$ there exist $w_1, w_2 \in \Sigma^*$ such that:*

- $|w_1|_u < |w_1|_v$ *and*
- $|w_2|_u > |w_2|_v$

Proof. We will treat the problem by decomposing it into two cases.

1. Neither of the words u and v is a subword of the other.

   In this case by choosing $w_1 = v$ and $w_2 = u$ the statement is proved.

2. u is a subword of v (the symmetric case of v being a subword of u can be treated identically).

   In this case $w_2 = u$ takes care of the second part of the statement so we have to find w_1 such that $|w_1|_u < |w_1|_v$. Consider writing v in the form $v = a_1^{k_1} a_2^{k_2} ... a_n^{k_n}$, with $a_i \in \Sigma, k_i \geq 1$ for $1 \leq i \leq n$, and $a_i \neq a_{i+1}, 1 \leq i \leq n-1$. Since u is a subword of v we can write it in the form $u = a_{i_1}^{l_1} a_{i_2}^{l_2} ... a_{i_m}^{l_m}$, where $i_j \leq n$ for $1 \leq j \leq m$, and $i_j < i_{j+1}$ for $1 \leq j \leq m-1$.

   Now consider some word $w_0 = a_1^k a_2^k ... a_n^k$ with

$$k > \max(\max_j\{l_j\}, \max_i\{k_i\})$$

   . The block sequence of an occurrence $I = (i_1, i_2, .., i_r)$ of u in w_0, in symbols *bseq*(I, w_0), obtained by replacing every i in I with $\lfloor (i-1)/k \rfloor + 1$, that is with the index of the w_0 block containing the respective letter (here $\lfloor x \rfloor$ is the integer part of x). For example, an occurrence of *aabb* in *aabbaabb* is $(1, 2, 4, 7)$, and the block sequence corresponding to it is $(1, 1, 2, 4)$, because the two a's were taken from the first block and the b's from the second and the fourth block, respectively. Denote with *bseq*(u, w_0) the number of different block sequences corresponding to all occurrences of u in w_0. For each different block sequence we mark with *bperb*(j) the number of different indices corresponding to the j-th block in u, e.g. for $w_0 = aabbaabb$, $u = aabb$ and the block sequence $(1, 3, 4, 4)$ we get *bperb*$(1) = 2$ and *bperb*$(2) = 1$. Using this notation, the number of occurrences of u in w_0 corresponding to a particular block sequence is

$$\prod_{j=1}^{m} \binom{k \cdot bperb(j)}{l_j}$$

   and this is clearly smaller than or equal to $\prod_{j=1}^{m} \binom{nk}{l_j}$. Hence, summing for all block sequences we get that

$$|w_0|_u \leq bseq(u, w_0) \cdot \prod_{j=1}^{m} \binom{n \cdot k}{l_j}$$

Note that $bseq(u, w_0)$ does not depend on k and n is fixed, so $|w_0|_u$ is bounded from above by a polynomial in k of degree $\sum_{j=1}^{m} l_j = |u|$. At the same time

$$|w_0|_v = \prod_{i=1}^{n} \binom{k}{k_i}$$

that is $|w_0|_v$ is a polynomial in k of degree $\sum_{i=1}^{n} k_i = |v|$. Moreover, u is a subword of v, i.e. $|u| < |v|$, therefore there exists some K such that for any $k > K$:

$$|a_1^k a_2^k ... a_n^k|_u < |a_1^k a_2^k ... a_n^k|_v$$

□

We saw that inequalities between monomial subword histories, i.e. of the form $u \leq v$, hold if and only if $u = v$.
Next we will look at a basic form of SI's, where the coefficients of the terms are all equal to 1, but we shall cite the following result first:

Lemma 10. *[46] Two linear subword histories are equivalent iff they are identical, apart from the order of terms.*

Because of this, if we find the same term appearing on both sides of a SI, we can simply remove them without affecting the relation. From now on we will consider only SI's where there are no identical terms on the two sides. Let us start with the case when the left hand side has one term and the right hand side has two.

Lemma 11. *A SI of the form $z \leq u + v$ holds if and only if for some $x_1, x_2 \in \Sigma^*$ and $a \in \Sigma$:*

- $z = x_1 a x_2$
- $u = x_1 x_2$
- $v = x_1 a^2 x_2$

Proof. Remember that for keeping things simple we omitted the subword notation when writing down a SI. However, here in the proof we will refer to w quite often, so please recall that w comes from the complete form of a SI:

$$|w|_z \leq |w|_u + |w|_v$$

where the inequality is supposed to hold for every $w \in \Sigma^*$.

(IF)

Consider the words $u = a_1 a_2 ... a_m$ and $v = a_{m+3} ... a_n$. An occurrence of ua^2v in a word w is given by the vector $(i_1, .., i_m, i_{m+1}, .., i_n)$, where i_{m+1} and i_{m+2} indicate the position of the a's in the middle. To count the occurrences of ua^2v in w we can proceed by taking all the occurrences where the indices corresponding to u and v are different and multiplying them with the number of possibilities of choosing a's from $w(i_m, i_{m+3})$. So if we fix the position of u and v in w and only consider the possibilities for choosing the a's we get the following:

$$|w(i_m, i_{m+3})|_a \leq 1 + |w(i_m, i_{m+3})|_{aa}$$

and this holds because $\binom{n}{1} \leq \binom{n}{0} + \binom{n}{2}$ for any $n > 0$.

(ONLY IF)

Suppose the three words cannot be written in the form given by the theorem. Now, depending on the length of the words, we have to distinguish among a few cases:

1. $|u| \geq |z|$ and $|v| \geq |z|$: by choosing $w = z$, we get $|w|_u = |w|_v = 0$ and $|w|_z = 1$ which contradicts the SI.

2. $|u| < |z|$ and $|v| < |z|$: if $z = a_1 a_2 ... a_m$ one can find a counterexample of the form $w = a_1^k a_2^k ... a_m^k$. The proof goes similarly to Theorem 15: $|w|_u$ and $|w|_v$ are polynomials of smaller degree than $|w|_z$.

3. $|u| < |z|$, $|v| \geq |z|$ and z is not a subword of v: first, u needs to be a subword of z otherwise $w = z$ leads to a contradiction. Then we have three subcases:

   (a) $red(z) = b_1 b_2 ... b_n \neq red(v)$: again some

$$w = b_1^k b_2^k ... b_n^k$$

will contradict the SI, just like in the proof of Theorem 15.

(b) $red(z) = red(v) = b_1b_2...b_n$ and there exist $j_1 \neq j_2$ such that $pow(z)$ differs from $pow(u)$ at position j_1 and $pow(z)$ differs from $pow(v)$ at position j_2: let $pow(u) = (p_1, .., p_n)$, $pow(z) = (k_1, .., k_n)$ and $pow(v) = (l_1, .., l_n)$ be the power vectors of u, z and v, respectively. In this case there is some $i \leq n$ such that $k_i < l_i$. Now, for a big enough k (see proof of Theorem 15) consider the word

$$w = b_1^k b_2^k ... b_i^{k_i} ... b_n^k$$

(that is, we leave the i-th block unchanged). $|w|_v = 0$ reduces the SI to $|w|_z \leq |w|_u$ and results in $|w|_z$ being a polynomial in k of degree $|z| - k_i$. At the same time $p_j < k_j$ so $|w|_u$ is a polynomial in k of degree smaller than $|z| - k_i$, thus w contradicts the SI. Note that the same argument does not work if a decomposition mentioned in the theorem exists, since then the only block we could leave unchanged for reducing $|w|_v$ to 0 is the exact block where we could make $|w|_z$ grow faster than $|w|_u$, making it impossible to find a counterexample of the mentioned form.

(c) $z = x_1a^{i_z}x_2$, $u = x_1a^{i_u}x_2$ and $v = x_1a^{i_v}x_2$: as we have seen in the (IF) part the SI reduces to

$$\binom{n}{i_z} \leq \binom{n}{i_u} + \binom{n}{i_v}$$

and since either $i_u \neq i_z - 1$ or $i_z \neq i_v - 1$ it is easy to find an n that leads to contradiction.

□

The decomposition in Lemma 11 is not unique for a given left hand side term. For example, if the term $baabba$ is on the left hand side, we can choose the triple (x_1, a, x_2) to be (ba, a, bba) or (baa, b, ba), respectively. The resulting SIs (with dots marking the decomposition):

- $ba.a.bba \leq ba.bba + ba.aa.bba$
- and $baa.b.ba \leq baa.ba + baa.bb.ba$ hold in both cases.

In the proof of the previous lemma we saw that whenever the terms are identical except for one block, the SI reduces to an inequality between binomial coefficients. Let's take, for instance,

$$b.a.b + b.aaa.b \leq bb + b.aa.b + b.aaaa.b$$

It becomes clear that this inequality holds when we express it in terms of binomial coefficients:

$$\binom{n}{1} + \binom{n}{3} \leq \binom{n}{0} + \binom{n}{2} + \binom{n}{4}$$

In general, using some basic properties of binomial coefficients, we can extend the previous lemma to multiple terms on both sides.

Lemma 12. *Let us consider a set of inequalities* $u_i \leq v_i + v_{i+1}, 1 \leq i \leq n$. *If all these inequalities hold and in addition to this,* $v_{i+1} \leq u_i + u_{i+1}$ *for all* $1 \leq i \leq n-1$, *then*

$$u_1 + u_2 + .. + u_n \leq v_1 + v_2 + .. + v_{n+1}$$

also holds.

Proof. From Lemma 11 we know that the conditions formulated in this lemma induce that all the terms on the left hand side and on the right hand side will have the same reduced form and their power vectors will be different in exactly one position, so we get a SI looking like this:

$$x_1ax_2 + x_1a^3x_2 + .. + x_1a^{2k+1}x_2 \leq x_1x_2 + x_1a^2x_2 + .. + x_1a^{2k+2}x_2$$

which reduces to

$$\binom{n}{1} + \binom{n}{3} + .. + \binom{n}{2k+1} \leq \binom{n}{0} + \binom{n}{2} + .. + \binom{n}{2k+2}$$

Now using Pascal's rule , i.e. $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$, and removing identical terms appearing on both sides we arrive at

$$0 \leq \binom{n-1}{2k+2}$$

so we can conclude the proof. □

In $|w|_{ba^kb} \leq |w|_{ba^{k-1}b} + |w|_{ba^{k+1}b}$ the central block of a's needs to be closely bounded from below and above to make up for the w's which have n a's in the middle so that $\binom{n}{k+1} < \binom{n}{k}$. If we allow the coefficients of the terms to be different than 1, then we can relax the strict condition for the number of a's by multiplying the terms on the right hand side to make up for the cases where $\binom{n}{k+1}$ has not outgrown $\binom{n}{k}$ yet. For example $bab \leq bb + baaab$ does not hold but $bab \leq 2bb + baaab$ does. In general for $ba^ib \leq ba^jb + ba^kb$, where $j < i < k$, the term with k a's will be equal to the one with i a's when the containing word will have $i + k$ a's so we have to set the coefficient of the shorter term in such a way that it compensates for the cases when the containing word has less than $i + k$ a's in the middle.

Lemma 13. *A SI of the form $\alpha z \leq \beta_1 u + \beta_2 v$ holds if and only if there exist $x_1, x_2 \in \Sigma^*, a \in \Sigma$ and $0 \leq j < i < k$ such that:*

- $z = x_1 a^i x_2$,
- $u = x_1 a^j x_2$,
- $v = x_1 a^k x_2$ *and*
- $\alpha\binom{n}{i} \leq \beta_1\binom{n}{j} + \beta_2\binom{n}{k}$ *holds for every* $n \geq 0$.

Proof. The proof is basically the same as the one for Lemma 11 except for the last part, because the last condition of this lemma makes up precisely for point 3.(c) in the proof of Lemma 11. □

Now for SI's having arbitrary coefficients we can state our main result, which follows from Lemma 12 and Lemma 13.

Theorem 16. *A SI of the form $\alpha_1 u_1 + .. + \alpha_n u_n \leq \beta_1 v_1 + .. + \beta_{n+1} v_{n+1}$ holds if both $\alpha_i u_i \leq \beta_i v_i + \beta_{i+1} v_{i+1}$ and $\beta_{i+1} v_{i+1} \leq \alpha_i u_i + \alpha_{i+1} u_{i+1}$ hold for every $i \leq n$ and $i \leq n - 1$, respectively.*

Naturally, if we have some true SI's, their sum is also a valid SI, therefore if we can decompose a SI into others of the form mentioned in Theorem 16, then the SI holds. As an example let us look at the following SI:

$$baba + baaaba + baabba < bba + 2baaba + baaaaba + baabbba$$

This SI is the sum of

$$baba + baaaba \leq bba + baaba + baaaaba$$

and

$$baabba \leq baaba + baabbba$$

both of which satisfy Theorem 16, therefore hold for any word.

It can be seen that a terminating algorithm can perform this decomposition (including, of course, the verification whether the components hold), so if one can prove that the only SI's that hold are the ones that can be decomposed in the aforementioned way, the decidability question would be solved.

Chapter 5

Powers of regular languages

The concept of primitive roots, just like that of primitive words, came up with the paper [12] of Lyndon and Schützenberger and it has received special interest in the algebraic theory of codes and formal languages. As mentioned briefly in the introduction, perhaps the most thrilling question about primitive words over a given alphabet is whether the language formed by them is context-free. Although the formulation is simple - like with other famous open problems - it has proved very difficult to answer it. The first thought, of course, was to try the well-known pumping lemmas for context-free languages. We have several of them, they are easy to use, but as Dömösi et al. proved in [13], they are of no help this time. In [13] another condition for being context-free, namely semi-linearity was investigated. It was shown that $Q \cap (a^*b)^n$ (where $n \geq 1$) is Parikh semi-linear, so this attack failed too. It was shown, that Q is not deterministic context-free [12] and moreover, cannot be generated by an unambiguous context-free grammar [49]. At the same time it can easily be seen that Q is accepted by a deterministic linear bounded automaton. On the other hand, the language of non-primitive words is not context-free by Ogden's lemma. Unfortunately this does not help us with the original question as the class of context-free languages is not closed under complementation.

A related open problem posed by Calbrix and Nivat concerns the regularity of certain languages consisting mainly of non-primitive words. After quickly going through the partial results obtained previously on the matter we are going to solve the problem in the following sections.

5.1 Roots and Powers of Regular Languages

Strongly connected with the notion of primitive words is that of the root of a word. For a nonempty word w, there exists a unique primitive word p and $n \geq 1$ such that $w = p^n$. This primitive word p is called the root of w, denoted by $\sqrt{w}$. The set of all roots of words from a language L form the root of L, denoted by $\sqrt{L}$. Horváth and Kudlek [26] have shown that the membership problem in Q, that is the language of primitive words is decidable in quadratic time. Lischke [40] investigates general relationships between the computational complexity of a language L and that of its root, showing that the complexity of the root can be arbitrarily great. It is easy to see that $\sqrt{\Sigma^*} = Q$, $\sqrt{L} \subseteq Q$ for each language L, and $\sqrt{L} = L$ if and only if $L \subseteq Q$. In this chapter we turn towards an open problem which is, as we will see, in strong connection with the roots of languages.

Calbrix and Nivat [5] define the power $pow(L)$ of a language L. This is the set of all words p^k where $p \in L$ and k is a natural number. It is easy to see that for a regular language, its power may be a regular language too or not. Take for instance the regular language ab^* whose power is not even context-free. Calbrix and Nivat raised the problem to characterize those regular languages whose powers are also regular, and to decide the problem whether a given regular language has this property. They conjecture that "rational languages such that their power is also rational are 'almost' a union of rational subsemigroups of Σ^* and the point is to give the right sense to this almost". We recall the context in which the problem was proposed. The rational ω-languages are characterized by their ultimately periodic words, of the form uv^ω. Of course $uv^\omega = u(v^k)^\omega = uv^l(v^k)^\omega, l, k \geq 0$. Given $M \subseteq \Sigma^\omega$ one defines its "periods": $per(M) = \{v \in \Sigma^+ | \exists u \in \Sigma^*, uv^\omega \in M\} \subseteq \Sigma^*$. An important result is that if M is rational, then $per(M)$ is rational. Note that $pow(per(M)) = per(M)$. We can also consider a partial representation $L \in Rat(\Sigma^*)$ such that $pow(L) = per(M)$, and ask whether $pow(L) \in Rat(\Sigma^*)$. For classes of the Chomsky hierarchy other than the regular languages, the question of the stability of the power operator is easy and has a positive answer. We note the following facts:

- the power of a recursively enumerable language is recursively enumerable too: one can enumerate the same way as $\mathbb{N}^2$;

- the power of a recursive language L is also recursive: a Turing Machine can, in a finite time, look for all decompositions of a word $u \in \Sigma^*$, in the form of a power $u = v^k$, and test if $v \in L$ (on another part of the band);

- we have the same result for context-sensitive languages.

The set of rational languages over a finite alphabet Σ denoted by $Rat(\Sigma^*)$ is the smallest set of languages which contains the finite languages and is closed under union ($\cup$), concatenation ($\cdot$), and star (*) (the reflexive transitive closure of the concatenation). It is known [32] that $Rat(\Sigma^*)$ is also the set of languages which are recognizable by finite automata, and is closed under complement and intersection.

Cachat [4] gives a partial solution to this problem showing that for a regular language L over a one-letter alphabet, it is decidable whether $pow(L)$ is regular. He restricts himself essentially to the special case of a one-letter alphabet and gives an effective solution for it. With this restriction, the languages can be easily represented as sets of integers, and we can use some elementary facts of arithmetic.

For a one-letter alphabet $\Sigma = \{a\}$ each word a^k of a^* is characterized by its length $k \in I\!N$. We identify a given $L \subseteq a^*$ with the set $\{k \geq 0 | a^k \in L\} \subseteq I\!N$.

The product of sets of integers is defined with the usual multiplication: $\forall M, O \subseteq I\!N, M \cdot O = \{m \cdot n | m \in M, n \in O\}$. We might omit the symbol ($\cdot$) for the multiplication. For any $L \subseteq a^*$, $pow(L)$ is isomorphic to $M I\!N$, with $M = \{k \geq 0 | a^k \in L\}$, just because $(a^k)^j = a^{kj}$ for $j \geq 0$. That is to say, multiplication over the integers corresponds to the power operation over words. Now we can formulate the problem in terms of integers: given $M \in Rat(I\!N)$ we want to determine whether $M I\!N \in Rat(I\!N)$.

By convention $nI\!N = \{nk | k \in I\!N\}$, and $[a, b)$ is the segment of integers between a and b (a included, b excluded). In the situation of this lemma, using the terms from [14] we will say that n is eligible as a period for M, and (n, m, I, P) is a representation of the rational language M.

Cachat's main result is the following:

Theorem 17. *[4] For a given language $L \in Rat(I\!N)$, one can decide algorithmically whether $L I\!N \in Rat(I\!N)$. We denote $Inv(m, q) = \{x \in [m, m + q) | gcd(x, q) = 1\}$, the set of integers relatively prime to q ("invertible" in $\mathbb{Z}/q\mathbb{Z}$), between m and*

$m+q-1$. *Algorithm: Rationality test for* $L\mathbb{N}$, L *rational*

Input: $L \in Rat(\mathbb{N})$ *represented by* (q, m, I, P) *with* $L = I \cup (P + q\mathbb{N})$, *where* $q \geq 1, m \geq 0, I \in [0, m), P \in [m, m+q)$

Output: $L\mathbb{N} \in Rat(\mathbb{N})$ *or* $L\mathbb{N} \notin Rat(\mathbb{N})$

1. If $1 \in L$, *then* $L\mathbb{N} \in Rat(\mathbb{N})$, *end.*

2. Else, if $\emptyset \neq Inv(m, q) \cup P \neq Inv(m, q)$, *then* $L\mathbb{N} \notin Rat(\mathbb{N})$, *end.*

3. Else, if $Inv(m, q) \subseteq P$, *obtain the answer with the equivalence:*

$L\mathbb{N} \in Rat(\mathbb{N}) \Leftrightarrow \forall p \leq m+q, pprime, \exists b \geq 1, p^b \in L$, *end.*

4. Else we have $\emptyset = Inv(m, q) \cap P$. *Compute, for each prime divisor* u *of* q *such that* $\exists x \in P, u|x$,

$$I_u = \{\frac{x}{gcd(u,x)} | x \in I\},$$

$$P_u = \{x \in P | u \nmid x\} \cup \bigcup_{x \in P, u|x} \{\frac{x}{u}, \frac{x+q}{u}, ..., \frac{x+q(u-1)}{u}\},$$

and $I'_u = (I_u \cup (P_u + q\mathbb{N})) \cap [0, m)$, $P'_u = (P_u + q\mathbb{N}) \cap [m, m+q)$. *Call recursively the algorithm with* (q, m, I'_u, P'_u), *to determine whether* $L'_u\mathbb{N} \in Rat(\mathbb{N})$, *where* $L'_u = I'_u \cup (P'_u + q\mathbb{N})$ *and* $L'_u\mathbb{N} = \frac{u\mathbb{N} \cap L\mathbb{N}}{u}$. *Collect every answer. Then answer with:* $L\mathbb{N} \in Rat(\mathbb{N}) \Leftrightarrow \forall uprime, u|q$, *such that* $\forall x \in P, u|x, L'_u\mathbb{N} \in Rat(\mathbb{N})$

Cachat also suggests to consider as the set of exponents not only the whole set $\mathbb{N}$ of natural numbers but also an arbitrary regular set of natural numbers. This suggestion is taken up in [27] in the definition of $pow_H(L)$. The authors answer partially the problem of Calbrix and Nivat and the open question of Cachat for languages over any finite alphabet and almost any finite set of exponents.

The class REG of regular sets can be split up into two classes FR and IR of sets whose roots are finite and infinite, respectively. It is proved by Horvath et al. that FR is closed under the power with an arbitrary finite set. Later on they show that the power of any set from IR with any nonempty regular exponent containing none of the numbers 0, 1, 2, is context-sensitive but not context-free. In their paper it is also discussed whether it is decidable for a regular set whether its power with any nonempty regular exponent is regular too. For the set N of all natural numbers as an exponent the only case left open is when the language has a non-regular intersection with its root and the difference between the power of the language and the language itself has a finite root. It is supposed by the authors that also in this case the regularity of the power of the language is decidable. As we mentioned the notion of power used in [5] is extended here:

$$pow_H(L) = \{w^k | w \in L, k \in H\}$$

For a regular grammar G and a nonempty set $H \subseteq N$, Horváth et al. propose the following procedure to decide the quality of $pow_H(L(G))$. Let $L = L(G)$.

1. If $H \subseteq \{0, 1\}$ then $pow_H(L) \in REG$.
2. Decide whether $\sqrt{L}$ is finite.
3. If $\sqrt{L}$ is finite then

   if H is finite then $pow_H(L) \in REG$,

   if $H = I\!\!N$ then it is decidable whether $pow_H(L) \in REG$.
4. If $\sqrt{L}$ is infinite then

   if $H \in \{\{2\}, \{0, 2\}\}$ then $pow_H(L) \notin REG$,

   if $H \setminus \{0, 1, 2\} \neq \emptyset$ then

   if $1 \notin H$ or $L \cap \sqrt{L}$ is regular or $pow_H(L) \setminus L$ has an infinite root then $pow_H(L) \notin CF$.

Note that from the original problem of Calbrix and Nivat ($H = N$) the following case is left open in [27]: given a regular language L which has a non-regular intersection with its root, and for which $pow_H(L) \setminus L$ has a finite root is the power of L regular? The authors suppose a positive answer, and we will prove their conjecture in the next section.

5.2 The power of a regular language

To make proofs easier in this section, we will need a corollary of Theorem 6 by Lyndon and Schützenberger. The result was first obtained by Shyr and Yu, but as the original proof is rather long and involved, we present it here with a short and simple proof.

Corollary 3. *[58] Let u, v be primitive words such that $u \neq v$. Then there is at most one non-primitive word in u^+v^+.*

Proof. Let $w = u^m v^n$ be non-primitive. Then either $m = 1$ or $n = 1$ by Theorem 6. So, by symmetry let $uv^n = w^i$ for some primitive word w and $i \geq 2$. We may choose n to be minimal with that property. It is enough to show that all uv^{n+k} are primitive. By contradiction, suppose that uv^{n+k} is not primitive for some $k \geq 1$, that is there exists some $z \in Q$ and $j \geq 2$ such that $z^j = uv^{n+k}$. It follows that $w^i v^k = z^j$.

First consider the case $k \geq 2$. Since $i, j, k \geq 2$, we can apply the Lyndon-Schützenberger theorem and get that $\sqrt{w} = \sqrt{v} = \sqrt{z}$, but then $u = v$, a contradiction.

Now let us see the case $k = 1$. Non-primitivity is invariant to cyclic shifts, so w^i and z^j being non-primitive gives us that $v^n u$ and $v^n uv$ are non-primitive too. Hence there are words $w_1, z_1 \in Q$ such that $w_1^i = v^n u$ and $z_1^j = v^n uv$, moreover $|w_1| = |w|$ and $|z_1| = |z|$. From here $z_1^j = w_1^i v$. As v is a prefix of w_1^i, we have that z_1^j has both periods $|z_1|$ and $|w_1|$. Now we can apply the theorem of Fine and Wilf and get that $z_1 = w_1 = v$. This means $w = z = v$ and then $u = v$, a contradiction again. □

From here we get the following:

Corollary 4. *For all words $x, y, z \in \Sigma^*$ with $y \neq \lambda$ with $|\sqrt{xyz}| \neq |\sqrt{y}|$, there is at most one non-primitive word in the language xy^+z.*

Proof. Suppose there exist numbers $i, j \geq 1$ with $i < j$ such that both xy^iz and xy^jz are non-primitive. Non-primitivity is invariant to cyclic shifts, so zxy^i and zxy^j are non-primitive too.

If zx is non-primitive then we can apply the Lyndon-Schützenberger theorem on zxy^j and get that $\sqrt{zx} = \sqrt{y}$. This would mean $\sqrt{zxy} = \sqrt{y}$ and from here $|\sqrt{xyz}| = |\sqrt{y}|$, a contradiction.

If zx is primitive then from Theorem 3 we have that only one of the words zxy^i and zxy^j is non-primitive, contradicting our original assumption. □

Now we are ready to move on to the open problem mentioned in the previous section. There are easy examples for non-trivial regular languages that do not have a regular power. Besides the one mentioned before one could take $aaa(aa)^*$ whose power $\{a^k : k$ is not a power of $2\}$ is not even context-free (in particular, powers of regular languages are not semi-linear, in general).

In fact, as it turns out, it is quite difficult to come up with examples of regular languages L with regular power other than the ones for which $L = L^*$ or $L = L^* \setminus K$, where either K is finite or $K = \bigcup_{w \in S} w^*$ for some finite set of words S. This seems to justify the conjecture formulated by Calbrix and Nivat cited before. Hence, rather than trying to solve the case left open in [27] one might try a new approach.

Indeed, as we will shortly see, we can give an equivalent criterion for a regular language to have a regular power, i.e., we can now give sense to that 'almost'.

Theorem 18. *Let L be a regular language. Then $pow(L)$ is regular if and only if* $\mathrm{pow}(L) \setminus L$ *is a regular language such that its primitive root is a finite language.*

Proof. The class of regular languages is closed under union and taking the difference of two sets, therefore if $\mathrm{pow}(L) \setminus L$ is a regular language then so is $(\mathrm{pow}(L) \setminus L) \cup L = pow(L)$.
Now let us look at the "only if" part. If $\mathrm{pow}(L)$ is regular then so is $L_{\text{diff}} = \mathrm{pow}(L) \setminus L$. Note that L_{diff} consists solely of non-primitive words. Let n be the number of states of the minimal deterministic automaton accepting L_{diff}. Now suppose that $\sqrt{L_{\text{diff}}}$ is infinite. In this case there must be some $w \in L_{\text{diff}}$ such that $|\sqrt{w}| > n$. On the other hand the pumping property of regular languages tells us that $w = xyz$ for some $xz, y \notin \{\lambda\}$ with $|y| \le n$ such that $xy^iz \in L_{\text{diff}}$ for all $i \ge 0$, so xy^iz is non-primitive for all i. Corollary 4 says that in this case $|\sqrt{xyz}| = |\sqrt{y}| \le |y| \le n$, contradicting the assumption $|\sqrt{w}| > n$.

□

Throughout the following proofs we will need the notion of syntactic monoid. For two words $u, v \in \Sigma^*$ and a language $L \subseteq \Sigma^*$, by saying that $u \equiv v(P_L)$ we mean the following

$$xuy \in L \text{ if and only if } xvy \in L \text{ for all } x, y \in \Sigma^*.$$

For a word $w \in \Sigma^*$ the congruence class $[w]_{P_L}$ consists of all words congruent with w according to P_L, that is $[w]_{P_L} = \{v \in \Sigma^* \mid w \equiv v(P_L)\}$. Since P_L is a congruence relation, $\Sigma^*/P_L = \{[w] \mid w \in \Sigma^*\}$ is a monoid, which is called the *syntactic monoid* of L and denoted by $\mathrm{Synt}(L)$.

Lemma 14. *Let L be a regular language given by an NFA having n states. If $pow(L)$ is regular, then we have*

$$pow(L) \subseteq L \cup \{\sqrt{u}^i \mid u \in L \wedge |u| \leq \max(n^2, m) \wedge i \geq 1\},$$

where m is the size of $\mathrm{Synt}(L)$.

Proof. We have seen in Theorem 18 that $pow(L)$ being regular means that it has to be a subset of $L \cup \bigcup_{u \in U} u^+$ for some finite set U of words. We need to prove that for every $w \in L_{\text{diff}}$ there is a $u \in L$ such that $w \in \sqrt{u}^+$ and $|u| \leq \max(n^2, m)$. Take the shortest $u \in L$ such that w is a power of u. If $|u| > \max(n^2, m)$ then according to the pumping property of regular languages u can be written as xyz for some $y \neq \lambda \neq xz$ such that $xy^jz \in L$ for all $j \geq 0$. Here we can distinguish two cases.

1. If $|y|$ is a multiple of $|\sqrt{u}|$ then $|\sqrt{u}| \leq n$. As $u \in \sqrt{u}^+ \cap L$ we can apply the pumping argument on powers of $\sqrt{u}$ as if it was a unary language. If k is the smallest number for which $\sqrt{u}^k \in L$ then $k \leq n$, or else there would be some numbers p, q with $p < q < k$ such that from the initial state we reach the same state by reading $\sqrt{u}^p$ or $\sqrt{u}^q$, and we could cut out $\sqrt{u}^{q-p}$ from the word. From here we get that there is a word $\sqrt{u}^k \leq n^2$ having the same root as w.

2. We are left with the case when in any decomposition $u = xyz$, $|y|$ is not a multiple of $|\sqrt{xyz}|$ and $|u| > m$. Then we find a decomposition $u = xyz$ with $0 < |y| \leq m$ and $[xy] = [x]$ in $\mathrm{Synt}(L)$. This way we know that $xy^jz \in L$, for all $j \geq 1$. As a consequence of Corollary 4 we also know that at most one of these xy^jz can be a non-primitive word. At the same time L_{diff} has finite root, hence for all but finitely many values of j, $(xy^jz)^+ \subseteq L$, so we find some j such that $xy^jz \in L$ and xy^jz is primitive and at the same time $(xy^jz)^+ \subseteq L$. Due to $[xy] = [x]$ in $\mathrm{Synt}(L)$ we can conclude $(xyz)^+ \subseteq L$. However, we supposed that $w \in L_{\text{diff}}$ is some power of xyz, a contradiction.

So for every $w \in L_{\text{diff}}$ there is some $u \in L$, with $|u| \le \max(n^2, m)$ such that $\sqrt{w} = \sqrt{u}$ and this concludes the proof.

□

To make it easy to see why the latter half of the previous theorem can be checked effectively, we should replace $\max(n^2, m)$ with a bound depending only on the number of states n of the automaton accepting L.

Remark 5. *Let L be a regular language given by an NFA having n states. If $pow(L)$ is regular, then we have*

$$pow(L) \subseteq L \cup \{u^i \mid u \in L \wedge |u| \le 2^{n^2} \wedge i \ge 1\},$$

where m is the size of $\mathrm{Synt}(L)$*.*

Proof. This is clear because $n^2 < 2^{n^2}$ and the syntactic monoid is a divisor of the monoid of Boolean $n \times n$ matrices, so $\mathrm{Synt}(L)$ has size at most 2^{n^2}. □

Let us recall the following result from the paper by Calbrix and Nivat about languages which are equal to their power.

Lemma 15. *[5] Let L be a regular language of Σ^*. Then $\mathrm{pow}(L) = L$ if and only if there are regular languages $(L_i)_{1 \le i \le n}$ such that $L = \bigcup_{i=1}^{n} L_i^+$.*

The statement above is useful for testing if a language is equal to its power or not, we only need to specify the languages L_i for an effective construction. Using the syntactic monoid of L gives us the tool we need. We can translate $pow(L) = L$ into the following statement involving the congruence classes of P_L:

$$\bigcup_{u \in L} [u]^+ \subseteq L = \bigcup_{u \in L} [u] \subseteq \bigcup [u]^+.$$

Given an automaton accepting L we can effectively construct its syntactic monoid and from here we can effectively define the set of words in the class $[u]$ for all $u \in L$. In the case of a regular language P_L induces a finite number of classes, so we can decide whether the equality holds or not. Hence, we can state the following.

Proposition 16. *For a regular language L it is decidable whether $pow(L) = L$ holds or not.*

Now we are ready to proceed with the algorithm. Theorem 18 reduces the original problem to an equivalent one of deciding whether the language, in some sense, lacks only a "few" words to be equal to its power. Lemma 14 provides the means to find those "few" missing words and after adding them to our starting language Proposition 16 will tell us whether the result is a power or not, that is whether the power of the original language is regular or not.

Theorem 19. *For a regular language L it is decidable whether* $\text{pow}(L)$ *is regular.*

Proof. We propose the following algorithm:

```
1. Input: an NFA $\mathcal{A} = \{\Sigma, Q, I, F, \sigma\}$.
2. Output: "YES", if $\text{pow}(L(\mathcal{A}))$ is regular, and "NO" otherwise.
3. $U = \emptyset$
4. FOR all words $w \in L(\mathcal{A})$ shorter than $2^{|Q|^2}$:
5. —IF $w^* \setminus L(\mathcal{A}) \neq \emptyset$ THEN:
6. ——IF $\text{pow}((\sqrt{w})^* \cap L(\mathcal{A}))$ is regular THEN add $w$ to $U$
7. ——ELSE output "NO"
8. compute the syntactic monoid for $L' = L(\mathcal{A}) \setminus \bigcup_{u \in U} (\sqrt{u})^*$
9. IF $L' = \text{pow}(L')$ then output "YES"
10. ELSE output "NO"
```

The enumeration of words in $L(\mathcal{A})$ shorter than $2^{|Q|^2}$ can be done in finite time due to the length limit. The condition in line 5 can be checked effectively too. First we have to perform the difference of two regular languages, then check whether the result is empty or not. As it is stated in [27], the condition in line 6 can be verified using Cachat's algorithm [4], because $(\sqrt{w})^* \cap L(\mathcal{A})$ is isomorphic to a unary language, which can be computed effectively. In step 8 we have to compute the syntactic monoid for a regular language, which is the difference of a regular language and the finite union of some regular languages, all effectively presented. If a regular language L is equal to $M \cup N$ for some regular languages M and N, such that $\sqrt{M} \cap \sqrt{N} = \emptyset$, then from the closure properties of the regular class we get that pow(L) is regular if and only if both pow(M) and pow(N) are regular. Moreover, L and M being powers implies N being a power as well. Therefore, in step 9 we only need to check whether a regular language is equal to its power or not; by Proposition 16 this is decidable too. Hence, the algorithm terminates after finitely many steps; however, the complexity is at least exponential due to both Cachat's algorithm and the exponential length bound on the words we need to check in step 4. □

5.3 Conclusion

We managed to characterize regular languages that have regular power following the conjecture of Calbrix and Nivat formulated in [5] and we gave an effective albeit inefficient procedure to decide this property. Although the decision procedure is not a direct extension of previous results [4, 27], Cachat's algorithm is needed in an essential step, which identifies those "few words" in pow(L) missing from L.

Chapter 6

Duplication closure of binary languages

Mathematically speaking duplication is a binary idempotence, that is a relation or operation the type $x = xx$, where the application of the operation on two identical elements gives the same element as a result. This notion of duplication, adapted to strings, is the one standing at the basis of this chapter. It is a relation having a strong motivation from outside of pure mathematics, namely it was first introduced in the context of DNA computing. Several DNA computation models were summarized by Păun et al. in their book on DNA Computing [68].

One of the most frequently occurring phenomena in genome rearrangement is gene duplication or the duplication of a segment of a chromosome [47]. In the process of gene repeating duplication, a stretch of DNA is duplicated to produce two adjacent copies, resulting in a tandem repeat. Several mathematical models have been proposed for the production of tandem repeats including replication, slippage and unequal crossing over [39, 56, 60]. The straightforward way to model such processes is to consider the string operation that takes a factor of a string and inserts a copy of it right next to the original occurrence.

With Σ being the alphabet, and arbitrary words $u, v, w \in \Sigma^*$ with $v \neq \lambda$ we say that the ordered relation $uvw \to uvvw$ is a duplication. Its transitive and reflexive closure $\to^*$ has the following meaning: if $u \to^* v$ we can obtain v from u by iterative duplications. The language generated from a word u through duplication or, in other words, the duplication closure of u, is $u^{\heartsuit}$:

$$u^{\heartsuit} = \{w \in \Sigma^* | u \to^* w\}.$$

The duplication closure of a language L is denoted similarly by $L^{\heartsuit}$:

$$L^{\heartsuit} = \bigcup_{u \in L} u^{\heartsuit}.$$

This operation has been extensively studied in theoretical computer science and bioinformatics. Bovet and Varricchio in a 1992 paper ([3]) proved that closure under duplication, copy languages in their terminology, of arbitrary recursively enumerable languages over a binary alphabet stays regular. Independently, in 1999 the study of duplication as we defined it just above restarted with a paper by Dassow et al. ([11]) in which the regularity of the duplication closure of a word over a binary alphabet is proved.

The notion of bounded duplication was introduced in [38]. On one hand it is motivated by the fact that in biology it is unrealistic to suppose that an arbitrarily large segment of a DNA strand can duplicate. On the other hand, for computational reasons it makes sense to put a bound on the length of the duplicating segments. Often in the cases where an algorithm to compute duplication the distance or to decide if $u \to^* v$ for two words u and v the general case probably presents us with possibly NP hard problems, whereas the bounded version of the same problem can be even linear.

For a positive integer n and words u, v, w as before, the relation $uvw \to_{\leq n} uvvw$ with $|v| \leq n$ is an n-bounded duplication. Again, the reflexive and transitive closure $u \to_{\leq n}^* v$ means that v can be obtained from u through iterated n-bounded duplication and the n-bounded duplication closure of u is

$$u^{\heartsuit \leq n} = \{w \in \Sigma^* | u \to_{\leq n\}^*} w,$$

and the n-bounded duplication closure of a language L is

$$L^{\heartsuit \leq n} = \bigcup_{u \in L} u^{\heartsuit \leq n}.$$

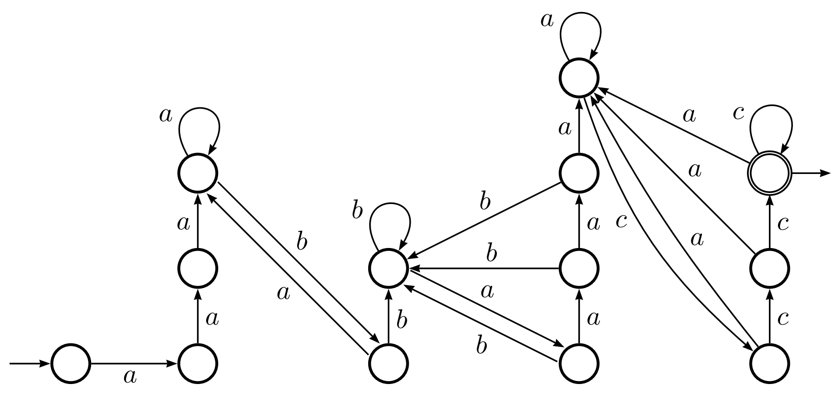

Figure 6.1: Automaton accepting $(a^3b^2a^4c^3)^{\heartsuit\leq 2}$

By substituting $\leq$ with $=$ in the previous definitions we get the notion of n-length or uniformly n-bounded duplication.

Example 5.

$$(ab)^{\heartsuit} = \{ab, aab, aabb, aababb, ...\} = a\{a,b\}^*b$$

$$(ab)^{\heartsuit\leq 2} = \{ab, aab, aabb, aababb, ...\} = a\{a,b\}^*b$$

$$(ab)^{\heartsuit=2} = \{ab, abab, ababab, ...\} = (ab)^+$$

The closure of language classes under bounded and uniformly bounded duplication is settled.

Theorem 20. *[31] For an arbitrary regular language L over a binary alphabet, $L^{\heartsuit\leq n}$ is regular for all $n \geq 1$.*

Theorem 21. *[31] Over an alphabet having at least three letters, the class of regular languages is not closed under n-bounded duplication with $n \geq 4$*

Let us turn now to the missing case of 3-bounded duplication over at least ternary alphabets. It is clear from the next example that if we consider 2-bounded duplication then the size of the alphabet does not really make a difference (see Figure 6.1).

The reason for this is that by duplicating a string of length at most 2 only creates two new blocks in the word and does not "mess up" the order (see [31]). A similar thing happens when considering 3-bounded duplication, as we will shortly see.

Proposition 17. *Let L be a regular language. Then, $L^{\heartsuit\leq 3}$ is regular.*

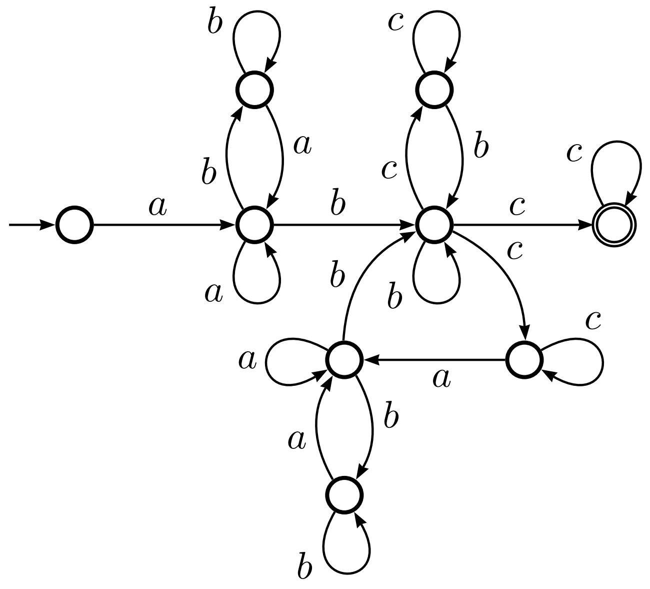

Figure 6.2: Automaton accepting $(abc)^{\heartsuit \leq 3}$

Proof. In Figure 6.2 we can see an automaton accepting the 3-bounded duplication closure of *abc*, the simplest example for a word over an at least ternary alphabet, where 3-bounded duplication differs from 2-bounded duplication. Here we started from the simple automaton that accepts only *abc* and added the loops to cover duplications of length 1, 2 and 3. We can extend the idea to arbitrary words and regular languages. For any regular language L we can construct a non-deterministic finite automaton $\mathcal{A} = \{\Sigma, Q, q_0, F, \sigma\}$ such that for all $u, v \in \Sigma^*$ if $\sigma(q_0, u) \cap \sigma(q_0, v) \neq \emptyset$ then either u is a suffix of v or u and v have a common suffix of length 3. We can achieve this by duplicating the paths which begin with a state reachable by at least two different words that do not have a common suffix of length 3. After this we only need to introduce the loops seen in Figure 6.2 for every state of the automaton. In this case it is clear that the automaton accepts all words that can be obtained from words of the original language by 3-bounded duplication. Also, since we started from an automaton in which any state is reachable only by words ending in the same 3 letters, adding these loops will not introduce words which cannot be obtained by 3-bounded duplication. $\square$

We can see right away that even if the context-free language class is not closed under bounded duplication, we would not be able to prove it neither by using the pumping lemmata seen before nor by Parikh's theorem because duplication closures inherently incorporate pumping. In fact, as the following result tells us, no other proof for this could be given.

Theorem 22. *[31] The class of context-free languages is closed under n-bounded duplication over arbitrary alphabet.*

Uniformly bounded duplications were considered by Leupold et al. [38]. They investigated the closure of words under this operation.

Theorem 23. *[38] For all words $w \in \Sigma^*$, the language $w^{\heartsuit=n}$ is a regular language over arbitrary alphabet Σ for all $n \geq 1$.*

We proceed now to give a new proof of the result by Bovet and Varricchio concerning the regularity of the duplication closure of any binary language.

Theorem 24. *[3] For all recursively enumerable languages $L \subseteq \{a, b\}^*$, the language $L^\heartsuit$ is regular.*

The original argument goes on by showing directly that duplication over a binary alphabet induces a well-quasi order on words and then uses the generalization of the Myhill-Nerode theorem given by Ehrenfeucht et al. to show that all these languages closed under duplication are regular.

Theorem 25. *[14] A language L of a finitely generated free monoid Σ^* is regular if and only if it is closed with respect to a monotone well quasi order.*

The main difference between our method and the one given in [3] lies in proving that duplication is a well quasi order. The original proof is done by reductio ad absurdum supposing that there is an infinite antichain with respect to binary duplication. However, the contradiction comes from a rather involved formal argument. We do this by reducing this order to the scattered subword relation which is known to be a well quasi order [24, 25]. Then the finite basis property of well quasi orders leads us to the eventual result. Our proof is somewhat shorter and we believe it is simpler as well. We recall the notion of blocks from the section on subword inequalities in Chapter 4. In the case of a word $a^{n_1}b^{n_2}....a^{n_k}$ we say that the word is composed of k blocks. By the reduced form("print") of a word $w = a_1^{n_1}a_2^{n_2}...a_m^{n^m}$ we mean $red(w) = a_1a_2...a_m$.

Proposition 18. *If two words u, v are composed of the same number of blocks and u is a scattered subword of v then $u \rightarrow^* v$.*

Proof. Since the words consist of the same number of blocks, the only way that u can be a scattered subword of v is that the ith block of u is a scattered subword of the ith block of v. From this it is obvious that v can be obtained by duplicating some letters inside the blocks of u. □

For a relation to be a well quasi order the following conditions have to be fulfilled:

- there is no infinite decreasing series with respect to this order
- there is no infinite series of pairwise incomparable elements

Lemma 16. *For any fixed n, duplication induces a well quasi order on the words consisting of n blocks.*

Proof. If the number n of blocks is fixed, then regardless of the size of the alphabet there can only be finitely many different prints among the words with n blocks. Hence, it is enough to prove that duplication induces a well quasi order on the words having the same print. As the prints are the same, any word can be represented by a vector of n dimensions holding the length of each block. In this case the duplication relation translates into an element-wise $\leq$. This structure is isomorphic to the duplication relation on binary words with the same number of blocks, and thus by Proposition 18 it is isomorphic to the scattered subword relation. At the same time the scattered subword relation is a well quasi order [24, 25] so we can conclude the proof. □

Theorem 26. *The relation $u \rightarrow^* v$ induces a well quasi order on a binary alphabet.*

Proof. Since for two different words u, v the relation $u \rightarrow^* v$ implies $|u| < |v|$ there is no infinite decreasing series with respect to duplication. As for the second condition, let us suppose that there is an infinite series $w_1, w_2, ...$ of words in which there are no two words w_i, w_j such that $w_i \rightarrow^* w_j$. This is possible in one of the two cases below.

1. There are infinitely many words composed of the same number of blocks. In this case Lemma 16 tells us there can be no infinite antichain among them.

2. There are infinitely many numbers n such that there are finitely many (more than 0) words which are composed of n blocks. It is easy to see that from a word $a^{n_1}b^{n_2}$ it is possible to obtain by duplication any word of the form $a^{n_1}\Sigma^*b^{n_2}$ the following way:

$$a^{n_1}b^{n_2} \rightarrow^* a^{n_1}a^*(ba)^*b^*b^{n_2}.$$

Then by forming the necessary number of blocks we just duplicate the letters inside the block. Let us take the outer two blocks of the words w_i, i.e. $a^{n_{i,1}}...b^{n_{i,2}}$. By the argument used above for words consisting of the same number of blocks there cannot be infinitely many words whose outer blocks are not comparable by duplication. This means that there must be at least one w_i such that there are infinitely many words in the series whose outer blocks can be obtained from $a^{n_{i,1}}...b^{n_{i,2}}$ by duplication. For the sake of simplicity consider the vector of the lengths of the blocks instead of the blocks themselves. Furthermore, let us separate the vectors of the 'a' blocks from the vectors of the 'b' blocks. Then having an infinite antichain means that for any vector v_i there cannot be another vector such that some scattered subvector of it would be greater or equal than v_i element-wise. Now let the vector of the word containing the least number of blocks have $m+1$ values, with the greatest among them being max. From the previous argument we have that every vector of the antichain contains at most m elements $\geq max$, so every vector can be written as $(p_1, ...p_{n_1}, q_1, p_{n_1+1}, ..., p_{n_2}, q_2, ..., q_m, p_{n_m+1}, ..., p_{n_{m+1}})$, where $p_i < max$, $1 \leq i \leq n_{m+1}$, and every q_i can take up arbitrary values, and . The number of the q_i values is fixed so we can divide each of these vectors into $2m+1$ parts with m fixed:

$$(p_1, ...p_{n_1}), (q_1), (p_{n_1+1}, ..., p_{n_2}), (q_2), (...., q_m), (p_{n_m+1}, ..., p_{n_{m+1}})$$

Now we can look at the odd parts of these vectors as words over a finite alphabet, because the number of values they can take up is finite. In a well quasi order we know that in any infinite sequence there must exist a perfect subsequence, i.e. an infinite strictly increasing sequence. So we know that there is an infinite increasing sequence of "words" over the "letters" $\{1, 2, ..., m-1\}$ in every odd position of the above list. Similarly, in the even positions there are words over a unary alphabet, again holding infinite perfect subsequences. This readily gives us that there can be no infinite antichain with respect to the vectors of the letters of the alphabet. Combining these vectors and applying the same argument all over again proves our statement.

□

Theorem 27. *[11] For all words $w \in \{a, b\}^*$ their duplication closure $w^\heartsuit$ is a regular language.*

As we have seen the duplication relation is a well quasi order on Σ^*, and so it possesses the finite basis property, that is every subset of Σ^* has at least one and at most finitely many minimal elements w.r.t. duplication. This in turn means, that for every language $L \subseteq \{a, b\}^*$ there is a finite set of words $M \subseteq L$ such that for all words $w \in L$ there is a $u \in M$ with $u \to^* w$, so the duplication closure of L is the (finite) union of the duplication closures of these minimal elements, that is, the union of finitely many regular languages:

$$L^\heartsuit = \bigcup_{u \in M} u^\heartsuit.$$

Bibliography

[1] A. Apostolico and F. P. Preparata. Optimal off-line detection of repetitions in a string. *Theoret. Comput. Sci.*, 22(3):297–315, 1983.

[2] J. Berstel, M. Crochemore, and J.-E. Pin. Thue-morse sequence and p-adic topology for the free monoid. *Discrete Mathematics*, 76(2):89–94, 1989.

[3] D. P. Bovet and S. Varricchio. On the regularity of languages on a binary alphabet generated by copying systems. *Inf. Process. Lett.*, 44(3):119–123, 1992.

[4] T. Cachat. The power of one-letter rational languages. In Kuich et al. [36], pages 145–154.

[5] H. Calbrix and M. Nivat. Prefix and period languages of rational *mega*-languages. In *Developments in Language Theory*, pages 341–349, 1995.

[6] M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Process. Lett.*, 12(5):244–250, 1981.

[7] M. Crochemore, S. Z. Fazekas, C. S. Iliopoulos, and I. Jayasekera. Bounds on powers in strings. In M. Ito and M. Toyama, editors, *Developments in Language Theory*, volume 5257 of *Lecture Notes in Computer Science*, pages 206–215. Springer, 2008.

[8] M. Crochemore and L. Ilie. Maximal repetitions in strings. *J. Comput. Syst. Sci.*, 2007. In press.

[9] M. Crochemore, L. Ilie, and L. Tinta. Towards a solution to the "runs" conjecture. In P. Ferragina and G. M. Landau, editors, *Combinatorial Pattern Matching*, LNCS. Springer-Verlag, Berlin, 2008. In press.

[10] M. Crochemore and W. Rytter. Squares, cubes and time-space efficient string-searching. *Algorithmica*, 13(5):405–425, 1995.

[11] J. Dassow, V. Mitrana, and G. Păun. On the regularity of duplication closure. *Bulletin of the EATCS*, 69:133–136, 1999.

[12] P. Dömösi, S. Horváth, and M. Ito. Formal languages and primitive words. *Publicationes Mathematicae*, 42(3-4):315–321, 1993.

[13] P. Dömösi, S. Horváth, M. Ito, L. Kászonyi, and M. Katsura. Formal languages consisting of primitive words. In Z. Ésik, editor, *FCT*, volume 710 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 1993.

[14] A. Ehrenfeucht, D. Haussler, and G. Rozenberg. On regularity of context-free languages. *Theor. Comput. Sci.*, 27:311–332, 1983.

[15] A. Ehrenfeucht and G. Rozenberg. On regularity of languages generated by copying systems. *Discrete Applied Mathematics*, 8:313–317, 1984.

[16] S. Z. Fazekas. On inequalities between subword histories. *Int. J. Found. Comput. Sci.*, 19(4):1039–1047, 2008.

[17] S. Z. Fazekas. Powers of regular languages. In V. Diekert and D. Nowotka, editors, *Developments in Language Theory*, volume 5583 of *Lecture Notes in Computer Science*, pages 221–227. Springer, 2009.

[18] S. Z. Fazekas, M. Ito, and K. Shikishima-Tsuji. Duplication closure of languages over a binary alphabets. In *International Conference on Automata, Languages and Related Topics*, 2008. Debrecen, Hungary.

[19] S. Z. Fazekas and B. Nagy. Primitive words and permutations. In *4th Conference of PhD Students in Computer Science*, 2004. Szeged, Hungary.

[20] S. Z. Fazekas and B. Nagy. Scattered subword complexity of non-primitive words. *Journal of Automata, Languages and Combinatorics*, 13(3/4):233–247, 2008.

[21] A. S. Fraenkel and J. Simpson. How many squares can a string contain? *J. Comb. Theory, Ser. A*, 82(1):112–120, 1998.

[22] F. Franek, R. J. Simpson, and W. F. Smyth. The maximum number of runs in a string. In M. M. . K. Park, editor, *Proc. 14th Australasian Workshop on Combinatorial Algorithms*, pages 26–35, 2003.

[23] M. Giraud. Not so many runs in strings. In C. Martin-Vide, editor, *2nd International Conference on Language and Automata Theory and Applications*, 2008.

[24] L. H. Haines. On free monoids partially ordered by embedding. *J. Comb. Theory*, 6:94–98, 1969.

[25] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2(2):326–336, 1952.

[26] S. Horváth and M. Kudlek. On classification and decidability problems of primitive words. *Pure Math. Appl.*, 6(2-3):171–189, 1995.

[27] S. Horváth, P. Leupold, and G. Lischke. Roots and powers of regular languages. In M. Ito and M. Toyama, editors, *Developments in Language Theory*, volume 2450 of *Lecture Notes in Computer Science*, pages 220–230. Springer, 2002.

[28] L. Ilie. A simple proof that a word of length has at most 2 distinct squares. *J. Comb. Theory, Ser. A*, 112(1):163–164, 2005.

[29] L. Ilie. A note on the number of squares in a word. *Theor. Comput. Sci.*, 380(3):373–376, 2007.

[30] C. S. Iliopoulos, D. Moore, and W. F. Smyth. A characterization of the squares in a Fibonacci string. *Theoret. Comput. Sci.*, 172(1–2):281–291, 1997.

[31] M. Ito, P. Leupold, and K. Shikishima-Tsuji. Closure of language classes under bounded duplication. In O. H. Ibarra and Z. Dang, editors, *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 238–247. Springer, 2006.

[32] S. C. Kleene. Representation of events in nerv nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

[33] R. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *Proceedings of the 40th IEEE Annual Symposium on Foundations of Computer Science*, pages 596–604, New York, 1999. IEEE Computer Society Press.

[34] R. Kolpakov and G. Kucherov. On maximal repetitions in words. *J. Discret. Algorithms*, 1(1):159–186, 2000.

[35] M. Kudlek and V. Mitrana. Closure properties of multiset language families. *Fundam. Inform.*, 49(1-3):191–203, 2002.

[36] W. Kuich, G. Rozenberg, and A. Salomaa, editors. *Developments in Language Theory, 5th International Conference, DLT 2001, Vienna, Austria, July 16-21, 2001, Revised Papers*, volume 2295 of *Lecture Notes in Computer Science*. Springer, 2002.

[37] K. Kusano, W. Matsubara, A. Ishino, H. Bannai, and A. Shinohara. New lower bounds for the maximum number of runs in a string. *CoRR*, abs/0804.1214, 2008.

[38] P. Leupold, C. Martín-Vide, and V. Mitrana. Uniformly bounded duplication languages. *Discrete Applied Mathematics*, 146(3):301–310, 2005.

[39] G. Levinson and G. Gutman. Slipped-strand mispairing: a major mechanism for dna sequence evolution. *Molec. Biol. Evol.*, 4:203–221, 1987.

[40] G. Lischke. The root of a language and its complexity. In Kuich et al. [36], pages 272–280.

[41] M. Lothaire. *Combinatorics on Words*. Addison Wesley, Reading, MA, 1983.

[42] M. G. Main. Detecting leftmost maximal periodicities. *Discret. Appl. Math.*, 25:145–153, 1989.

[43] M. G. Main and R. J. Lorentz. An $O(n \log n)$ algorithm for finding all repetitions in a string. *J. Algorithms*, 5(3):422–432, 1984.

[44] A. Mateescu and A. Salomaa. Matrix indicators for subword occurrences and ambiguity. *Int. J. Found. Comput. Sci.*, 15(2):277–292, 2004.

[45] A. Mateescu, A. Salomaa, K. Salomaa, and S. Yu. A sharpening of the parikh mapping. *ITA*, 35(6):551–564, 2001.

[46] A. Mateescu, A. Salomaa, and S. Yu. Subword histories and parikh matrices. *J. Comput. Syst. Sci.*, 68(1):1–21, 2004.

[47] A. Meyer. Molecular evolution: Duplication, duplication. *Nature*, 421:31–32, 2003.

[48] P. Ochsenschlager. Binomialkoeffizenten und shuffle-zahlen. *Technischer Bericht, Fachbereit Informatik, T.H. Darmstadt*, 1981.

[49] H. Petersen. On the language of primitive words. *Theor. Comput. Sci.*, 161(1&2):141–156, 1996.

[50] S. J. Puglisi, J. Simpson, and W. F. Smyth. How many runs can a string contain?, 2007. Personal communication, submitted.

[51] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer, 2004.

[52] W. Rytter. The number of runs in a string: Improved analysis of the linear upper bound. In B. Durand and W. Thomas, editors, *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 184–195. Springer, 2006.

[53] W. Rytter. The number of runs in a string. *Inf. Comput.*, 205(9):1459–1469, 2007.

[54] A. Salomaa. Counting (scattered) subwords. *Bulletin of the EATCS*, 81:165–179, 2003.

[55] A. Salomaa. Connections between subwords and certain matrix mappings. *Theor. Comput. Sci.*, 340(1):188–203, 2005.

[56] C. Schlotterer and D. Tautz. Slippage synthesis of simple sequence dna. *Nucleic Acids Res.*, 20:211–215, 1992.

[57] T.-F. Serbanuta. Extending parikh matrices. *Theor. Comput. Sci.*, 310(1-3):233–246, 2004.

[58] H.-J. Shyr and S.-S. Yu. Bi-catenation and shuffle product of languages. *Acta Inf.*, 35(8):689–707, 1998.

[59] A. Thue. Über unendliche Zeichenreihen. *Norske Vid. Selsk. Skr. I Math-Nat. Kl.*, 7:1–22, 1906.

[60] R. Wells. Molecular basis of genetic instability of triplet repeats. *J. of Biological Chemistry*, 271:2875–2878, 1996.