Comenius University
Faculty of Mathematics, Physics and Informatics
Department of Computer Science

On descriptional complexity of infinite words

(Master's Thesis)

Miroslava Kemeňová

Advisor: doc. RNDr. Pavol Ďuriš, CSc.

Bratislava, 2006

I hereby declare that the work presented in this thesis is my own, except where otherwise indicated, under the careful supervision of my thesis adviser.

. . . . . .

Acknowledgements

I am very grateful to my advisor Pavol Ďuriš for helpful discussions during the work on this thesis.

I am grateful to my friends from dormitory, for being a surrogate family during all the years I stayed there. I also wish to thank Michal Pokorný for his positive attitude.

Finally, I am indebted to my family for their encouragement and care.

Abstract

We consider several mechanisms generating infinite words over a finite alphabet in real time. The simplest and most used mechanism is iterating a morphism on free monoid. There are various natural generalizations of this method – e.g. substitutions, periodic iteration of morphisms and iterating a deterministic GSM. All these iterative mechanisms are related and can be unified on the framework of TAG systems. The main result of this thesis is proving the existence of a dense hierarchy within the class of binary infinite words obtained by substitution (CD0L TAG system) – depending on cardinality of the control alphabet. Similarly, there is a dense hierarchy within the class of infinite words obtained by iterating a DGSM – depending on number of states of the DGSM. Since the notion of the state of a DGSM is equivalent to the notion of the letter of the control alphabet of a DGSM TAG system, these are the same results for classes DGSM and binary CD0L.

Keywords. Infinite words, D0L TAG systems, iterating a morphism, substitution, iterating a GSM.

Contents

1 Introduction

The theory of words has various applications in many fields of science, but without doubt it is the theoretical computer science that is responsible for the expansion of study in this area. The basic object of this theory is a *word*, either finite or infinite, i.e. a sequence of symbols from a finite set.

In the last decades the research on infinite words extended rapidly. The most of the work was done on the combinatorial properties of infinite words, for example the unavoidability, see [Lo81]. Recent survey articles on this area are [CK97], [Lo02] and [BK03].

The goal of this thesis is to present new results on complexity of infinite words. There are various different ways to define the complexity measure of infinite words. In *computational complexity*, the measure is how much time or space is needed to print the n-th letter of a given infinite word. Quite a lot of research was done on the *subword complexity* – specifying for every n how many different factors of the length n there are within the given infinite word.

We will concentrate on the third type of complexity measure – the *descriptional complexity* which inquires how complicated mechanism is needed to generate a particular infinite word. Typical mechanisms used are iterated morphisms, codings of iterated morphisms, iterated deterministic GSM's and double D0L TAG system, see [Th06] and [CK94].

The comparison of the descriptional and computational complexity of infinite words was brought in [HKL94] and [DM03].

This thesis is organized as follows. In Section 2 we present the basic definitions of the theory of words.

In Section 3 we introduce the most used devices for generation of infinite words.

In Section 4 we prove the result that there is an infinite dense hierarchy within the class of binary codings of infinite words generated by iterating a morphism, namely CD0L TAG system – depending on the number of letters in the alphabet of the iterated morphism.

In Section 5 we show that there is an infinite dense hierarchy within the class of DGSM words, i.e. the infinite words obtained by iterated application of a deterministic GSM – depending on the number of states of the iterated GSM.

2 Preliminaries

Here we fix our terminology and recall basic notions on words, infinite words and mechanisms used to generate them.

Let Σ be a non-empty finite set referred to as an *alphabet*. Elements of Σ are called *symbols* or *letters*, and sequences over Σ are called *words*. A word can be finite or infinite. An empty word, which is denoted as ε, is a sequence of length zero.

We will omit commas and depict the word as:

$$a_0a_1a_2\dots a_n$$

instead of representing it as a formal sequence

$$(a_0, a_1, a_2, \dots, a_n)$$

Let $w = a_0a_1a_2\dots a_n$ be a word with $a_i \in \Sigma$. Then we say that n is the length of w and we represent it as $|w|$. By $|w|_a$, where $a \in \Sigma$ we mean the number of occurrences of letter a in w. We define $w(i)$ to denote the i-th letter of word w.

If the word u is a continuous subsequence of the word w, we call it a *subword* or a *factor* of w. We define an operator $alph(u)$ which denotes the set of all letters of Σ that occur in u.

Let u be a finite word over an alphabet Σ where $\Sigma = \{a_1, a_2, \dots, a_r\}$. We assign to the word u a vector $\vec{u}$ of length r where i-th component of $\vec{u}$ is equal to the number of occurrences of the letter a_i in u (i.e. $|u|_{a_i}$). The norm of the vector $\|\vec{u}\|$ is defined as sum of its components, i.e. it is equal to the length of the word u.

The *catenation*[1] is operation on two words $u = a_1a_2\dots a_n$ and $v = b_1b_2\dots b_m$ and the product of catenation is the word $uv = a_1a_2\dots a_nb_1b_2\dots b_m$. If $w = uv$, we call u the prefix of word w and v the suffix of w.

The set of all finite words over Σ is denoted as Σ^* and it is a free monoid under the operation of catenation. The set of all infinite words over Σ is denoted as Σ^ω. We use the notion Σ^+ for the set $\Sigma^* \setminus \{\varepsilon\}$.

The central object of this thesis is an *infinite word*, often also referred to as ω-word. Formally, it is a mapping from $\mathbb{N}$ into Σ, but we prefer to represent it as: $w = a_0a_1a_2\dots$ where $a_i \in \Sigma$. When there is no danger of confusion we call infinite words simply words.

[1] or *concatenation*

We call an infinite word *marked* if it is of the form

$$w = w_1 \# w_2 \# w_3 \# \ldots$$

where w_i's are finite words over Σ and $\# \notin \Sigma$. The word w_i will be referred to as a *block* of the word w.

A crucial notion for us is that of a *morphism* on free monoid, which is a mapping $h : \Sigma^* \to \Sigma^*$, satisfying

$$h(uv) = h(u)h(v) \quad \text{for all } u, v \in \Sigma^*.$$

Necessarily, $h(\varepsilon) = \varepsilon$ and it is sufficient to give only the values $h(a)$ for all letters $a \in \Sigma$ to define the morphism completely. We use notation $h : a \to w$ to denote that $h(a) = w$.

We shall need several specific types of morphisms. We say that the morphism h is:

non-erasing (or ε-free) if $|h(a)| > 0 \quad$ for all $a \in \Sigma$

prolongable on a (or that it satisfies the *prefix condition* on letter a) if $h : a \to aw \quad w \in \Sigma^+$

binary if $|\Sigma| = 2$, e.g. $\Sigma = \{0, 1\}$

coding if it is length-preservable, i.e. $|h(a)| = 1 \quad$ for all $a \in \Sigma$

Other definitions will be presented when needed.

3 Models and examples

The goal of this section is to describe some basic algorithmic methods used to generate the infinite words.

First, however, let us note that the cardinality of the set of infinite words is non-denumerable, and therefore any algorithmic model can cover only a small part of it. A simple argument for that is diagonalization.

We aim our attention to generating of infinite words by *iteration* (repeated application) of simple mappings, specifically morphisms on free monoids or their natural generalizations. We concentrate on the real-time generation of infinite words, therefore erasing is not allowed in our models, i.e. all considered mappings are ε-free.

We present several different but related mechanisms. Relationship among them and unified approach to real-time generation of infinite words was developed in [CK94].

3.1 Iterating a morphism

Probably the simplest and most commonly used procedure generating an infinite word is iterating a non-erasing morphism $h : \Sigma^* \to \Sigma^*$ on letter a on which it satisfies the prefix condition.

Thus $h(a) = ax$ for some $x \in \Sigma^+$ and consequently for all i the word $h^i(a)$ is a proper prefix of $h^{i+1}(a)$. Therefore the word w:

$$w = h^\omega(a) = \lim_{i \to \infty} h^i(a) = axh(x)h^2(x)h^3(x)\dots$$

is well defined, and we say that it is obtained by iterating a morphism h on letter a.

Generation of this word can be viewed also as a self-reading process using the same morphism h on the same letter a differently: Let us consider a potentially infinite tape with one reading and one writing head on it. At the beginning there is a word $h(a) = ax$ written on the tape, the reading head is on the first letter of word x and writing head is in the first empty position on the tape. Now we continue indefinitely by reading current symbol and moving the reading head right to the next symbol and simultaneously writing its morphic image and moving the writing head to the first empty position on tape. This is illustrated in Figure 1.

We call this model *D0L TAG system* and this self-reading mechanism was introduced by Post already in 1920's, see [Mi67]. Connection between iterating a morphism and D0L TAG system is obvious.

We call an infinite word w a *D0L word* if it, or word Sw is defined by either of these methods, where S is a symbol not in Σ. Although these mechanisms define the same class of infinite words, the TAG machine will serve us better when defining more general devices generating infinite words.

Example 3.1. *Even this simple tool can generate quite complicated infinite words. Typical example is the Thue-Morse word,* $t = 0110100110010110...$ *where* $t(n)$ *is the number of* 1*'s in the binary expansion* $bin(n)$ *of* n *modulo* 2*. It can be generated on letter* 0 *by morphism* $h : \{0,1\}^* \to \{0,1\}^*$

$$h : \begin{array}{ccc} 0 & \to & 01 \\ 1 & \to & 10 \end{array}$$

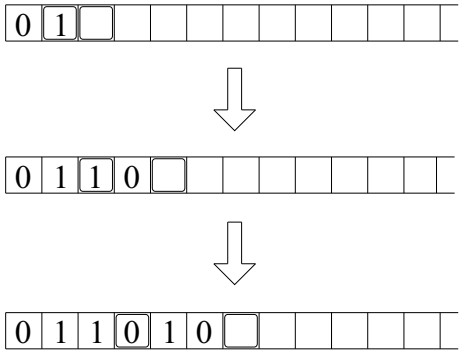

Figure 1: The generation of the Thue-Morse word by the D0L TAG system.

Example 3.2. *Another well-known example is the* Fibonacci word. *This word reflects the Fibonacci sequence into the theory of words. It is also defined by a similar recursion as Fibonacci numbers:*

$$f_0 = a, \quad f_1 = ab, \quad f_{i+1} = f_i f_{i-1}, \text{ for } i \geq 1$$

Then

$$f = \lim_{i \to \infty} f_i$$

is called Fibonacci word and can be obtained by iteration of a simple morphism f on letter a:

$$f: \begin{array}{ccc} a & \rightarrow & ab \\ b & \rightarrow & a \end{array}$$

3.2 Extensions of D0L TAG systems

There are at least two ways to extend the D0L TAG system.

- We can process the D0L word after its generation by some other method.
- We can use more general mappings than morphisms on free monoids.

Substitution (CD0L TAG system), which is a morphic image of a D0L word under a coding, is natural extension of the D0L TAG system of the first type. Formally, let $h : \Sigma_h^* \rightarrow \Sigma_h^*$ be a non-erasing morphism that is prolongable on a and let $c : \Sigma_h^* \rightarrow \Sigma_c^*$ be a coding. Then we call the pair (h, c) a substitution, generating a *CD0L word*:

$$w = c(h^\omega(a))$$

A way to describe this procedure by a TAG system is to use two tapes. The first tape is a *control tape* and generates a D0L word $h^\omega(a)$ and the second tape is a *generating tape* and letters generated by morphism h are translated from the first tape to the second tape by coding c. The result is the word generated on the second tape.

Example 3.3. *CD0L words are obtained from D0L words by merging some letters together. Therefore, for any morphism $h : \Sigma^* \rightarrow \Sigma^*$ the marked infinite word*

$$Sa^{|x|}\#a^{|h(x)|}\#a^{|h^2(x)|}\#\dots$$

is a CD0L word. Concrete examples are:

$$Pow_2 = Sa\#a^4\#a^9\#\dots\#a^{n^2}\#\dots$$

and

$$Sf_1\#f_2\#f_3\#\dots\#f_n\dots$$

where $f_n = a^{F_n}$ and F_n is the n-th Fibonacci number.

Iterating a deterministic GSM (DGSM TAG system) is an example of the modification of D0L TAG system of the second type – using more powerful mappings than morphism in the iteration. In this case, it is a deterministic generalized sequential machine (a DGSM), i.e. a deterministic finite state transducer. We assume that DGSM is non-erasing and satisfies the analogy of the prefix condition. To generate a *DGSM word* we can also use the TAG model. As in the CD0L case, we will use two tapes - control and generating tape. Control tape records only the current state of the DGSM which influences the generation of infinite word written on generating tape. This model is called DGSM TAG system.

Example 3.4. *The infinite word*

$$BIN = 0\#1\#01\#11\#\ldots\#rbin(n)\#\ldots$$

where $rbin(n)$ is the reverse binary representation of the number n is a DGSM word.

Double D0L TAG system is further generalization which captures both CD0L and DGSM extensions of D0L system. It consists of two infinite tapes each of which has one reading and one writing head on them. Reading heads are moving synchronously while writing heads may progress at a different pace. In each step of generation both reading heads read a letter from the tape and move right to the next letter while both writing heads write the corresponding outputs to first empty places of the tapes. Outputs are specified by rewriting rules of the form:

$$\binom{a}{b} \to \binom{\alpha}{\beta}, \text{ where } a, b \in \Sigma,\ \alpha, \beta \in \Sigma^+$$

Regardless the generality of this system we will refer to one tape as *generating* and to the other as *control* tape. We say that a word is a *double D0L word* if it is generated on the generating tape of double D0L system. That is so because we are interested in generating infinite words rather than pairs of infinite words. This model is presented in Figure 2.

Note that if in each rewriting rule $|\beta| = 1$ we obtain the DGSM TAG system.

We will present some more restrictions of double D0L TAG system - *D0L TAG system with X control*. These are double D0L TAG systems where the control tape is independent of the generating tape and itself generates a word

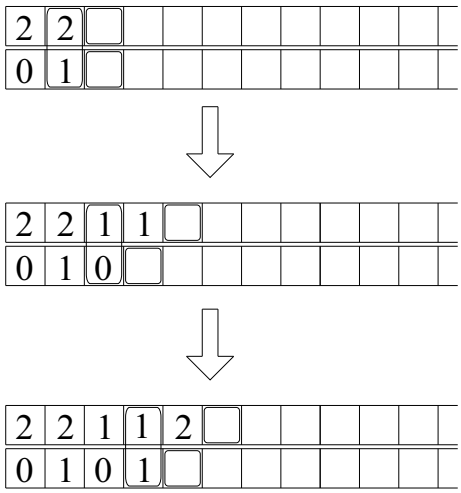

Figure 2: The generation of Kolakoski word by the double D0L TAG system.

of type X. We say also that this infinite word was generated by non-interactive outside control.

We consider only very simple instances where X is either *periodic* or *D0L*. Words obtained by this method are called *D0L words with periodic control* and *D0L with D0L control*, respectively. The rewriting rules of the case of D0L with D0L control are of the form:

$$\binom{a}{b} \to \alpha \quad \text{and} \quad b \to \beta$$

The word written on the control tape of the *D0L with periodic control* is ultimately periodic, i.e. a word of the form Sw^ω. This system is equivalent to *periodic iteration of morphisms*. In this method we use p non-erasing morphisms $h_0, h_2, \ldots, h_{p-1} : \Sigma^* \to \Sigma^*$ to generate the sequence of words $w_0, w_1, w_2, \ldots$, where $w_0 = a \in \Sigma$ and if $w_i = a_0 a_1 a_2 \ldots a_n$ then

$$w_{i+1} = h_0(a_0)h_1(a_1)h_2(a_2)\ldots h_{n \bmod p}(a_n) \tag{1}$$

If h_0 is prolongable on a then the product of iteration of these morphisms in the sense (1) yields a unique infinite word.

Example 3.5. *Even D0L with periodic control can generate infinite words*

with remarkable properties. Perhaps the most famous example is Kolakoski *word which is defined by the rules:*

- *It is made of consecutive blocks of symbols* 1 *and* 2
- *The first block is* 22
- *The length if the i-th block is equal to the value of i-th digit of the word*

It starts as follows:

$$Kol = 2211212212211\ldots$$

and it can be obtained by periodical iteration of morphisms:

$$h_0 : \begin{array}{ccc} 1 & \to & 2 \\ 2 & \to & 22 \end{array} \quad \textit{and} \quad h_1 : \begin{array}{ccc} 1 & \to & 1 \\ 2 & \to & 11 \end{array}$$

Kolakoski word possesses many interesting properties and is also a source of many open problems, see [Kn72] *and* [Sh88].

We can proceed also to generalization of the double D0L TAG system. One of the interesting models is generation of codings of DGSM words because this method combines both considered types of extensions of D0L TAG system at once. This model is known as *CDGSM TAG system.*

More general extension is a *triple D0L system* which consists of three totally interconnected D0L TAG systems where the transition rules are of the form:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \to \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}, \text{ where } a, b, c \in \Sigma,\ \alpha, \beta, \gamma \in \Sigma^+$$

There are many special cases of triple D0L TAG systems other than CDGSM TAG system but there is no motivation to formalize them. Moreover, there are infinite words that cannot be generated even by triple D0L TAG system or even by generalized D0L TAG systems with n tapes. This is sustained by the argument of diagonalization [CK94].

3.3 Hierarchy

Here we sketch the relative generating power of mentioned models. The diagram in Figure 3 shows known relations between different families of infinite words. The solid line denotes inclusion, the arrow meaning the proper inclusion, and the dotted arrow from X to Y denotes that X is not included in Y. Consequently, the bi-directional dotted arrow represents incomparability between X and Y.

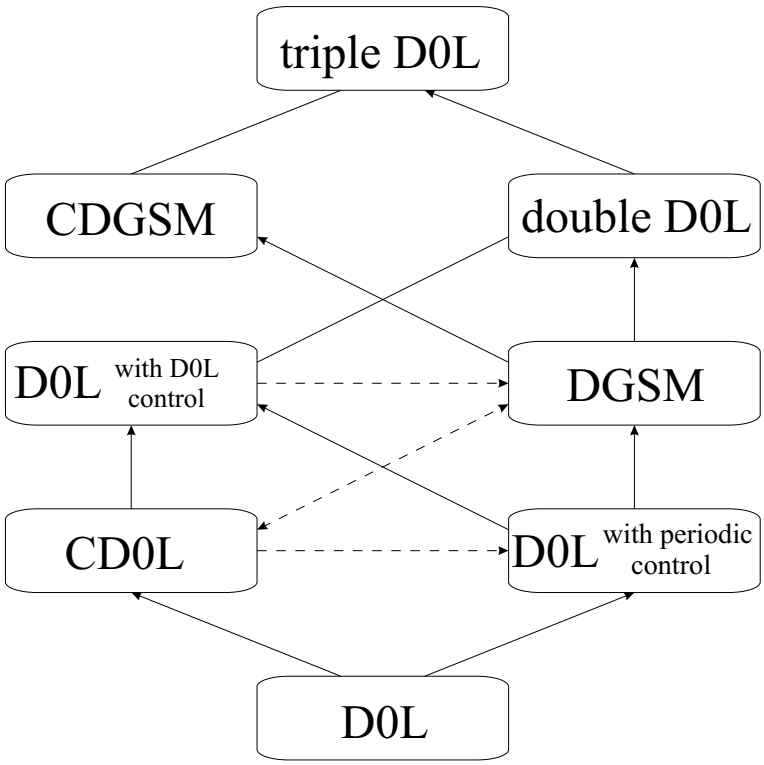

Figure 3: Comparison of the generating power of TAG systems.

The separation of classes *double D0L* and *triple D0L* was shown in [DM03]. The other relations were presented in [CK94]. However, there are many remaining open problems, see [CK94].

4 CD0L TAG system - cardinality of control alphabet

Here we aim our attention only to generation of binary infinite words. We will prove that for any $k \in \mathbb{N}$ there exist a binary infinite word that cannot be generated by CD0L TAG system (h, c) that has less than $k+1$ letters in the control alphabet Σ_h; and since it can be generated by CD0L TAG system having exactly $k+1$ letters in the control alphabet, there is a *dense hierarchy* within the class of binary CD0L words depending on the cardinality of Σ_h.

To establish this let us consider a class of binary marked infinite words

$$Pow_k = Sa^{n_1}\#a^{n_2}\#a^{n_3}\#\ldots\#a^{n_i}\#\ldots$$

where $n_i = i^k$. It is straightforward exercise to prove that these words are not D0L words.

Theorem 4.1. *There is a CD0L TAG system (h, c) generating infinite word Pow_k where cardinality of the control alphabet Σ_h is $k+1$.*

Proof. The proof is inductive. First, we define the substitution (h, c) which generates the word Pow_2:

$$h: \begin{array}{lcl} S & \to & Sa\# \\ a & \to & abb \\ b & \to & b \\ \# & \to & a\# \end{array}$$

And the coding

$$c: \begin{array}{lcl} a & \to & a \\ b & \to & b \\ \# & \to & \# \end{array}$$

We used 3 letters to generate Pow_2 ($\Sigma_h = \{a, b, \#\}$). Furthermore, let us note that in i-th block of generated control word $h^\omega(S)$, there is exactly i occurrences of letter a.

Now let us state the inductive hypothesis: Let the word

$$Pow_k = Sw_1\#w_2\#\ldots\#w_i\ldots$$

be generated by substitution (h_k, c_k) where $\Sigma_{h_k} = \{a_1, a_2, \ldots, a_k, \#\}$ and moreover, following equations hold for all the blocks of the control word $h_k^\omega(S)$:

$$\begin{aligned} |w_i|_{a_1} &= i \\ |w_i|_{a_1} + |w_i|_{a_2} &= i^2 \\ |w_i|_{a_1} + |w_i|_{a_2} + |w_i|_{a_3} &= i^3 \\ &\vdots \\ |w_i|_{a_1} + |w_i|_{a_2} + \ldots + |w_i|_{a_k} &= i^k \end{aligned} \tag{2}$$

We now define substitution (h_{k+1}, c_{k+1}), where $\Sigma_{h_{k+1}} = \{a_1, a_2, \ldots, a_k, a_{k+1}, \#\}$, which generates infinite word Pow_{k+1}.

$$h_{k+1}: \quad \begin{array}{ccc} a_i & \rightarrow & h_k(a_i).{a_{k+1}}^{t_i} \\ a_{k+1} & \rightarrow & a_{k+1} \\ \# & \rightarrow & a_1\# \end{array} \qquad \forall i, 1 \leq i \leq k$$

where t_i satisfies the constraint

$$|h_k(a_i).{a_{k+1}}^{t_i}| = 1 + b_i + b_{i+1} + \cdots + b_k$$

and $b_1, \ldots, b_k$ are coefficients of the polynomial

$$(n+1)^{k+1} - n^{k+1} = b_1.n^k + b_2.n^{k-1} + \cdots + b_k$$

Coding c_{k+1} is defined as follows:

$$\begin{array}{ccc} a_i & \rightarrow & a \\ \# & \rightarrow & \# \end{array} \qquad \forall i, 1 \leq i \leq k+1$$

Clearly, the control word $h^{\omega}_{k+1}(S)$ satisfies the equations (2) of the induction hypothesis. □

Now we are going to head towards the proof of the result that the infinite word Pow_k cannot be generated by a CD0L TAG system with less than $k+1$ letters in the control alphabet. This proof will consist of two main parts. First, we will prove that there is some kind of regularity and periodicity in the generation of this word by any CD0L TAG system. Then we shall use this regularity in the second part of the proof where we will show using the algebraic properties of the word Pow_k that less than $k+1$ letters in the control alphabet is not enough to generate this infinite word.

In the following three lemmas we will state some properties of the morphism which will serve us in the first part of the mentioned proof. In the first lemma we will prove that the function $alph(h^x(u))$ is ultimately periodic for every word u.

Lemma 4.1. *For every morphism $h : \Sigma^* \to \Sigma^*$ and every word w there exist positive integers Z and C such that for any $i_1, i_2, j \in \mathbb{N}$:*

$$alph(h^{Z+i_1C+j}(w)) = alph(h^{Z+i_2C+j}(w)) \tag{3}$$

Proof. Let G be a oriented graph where vertices represent all subsets of Σ and an oriented edge leads from the vertex A to vertex B iff B is the set of all letters occurring in the morphic images of letters from A, i.e. if $x \in \Sigma$ then

$$x \in B \iff \exists a \in A,\ x \text{ is a factor of } h(a)$$

The morphic image of every letter is uniquely defined, therefore there is exactly one outgoing edge from each vertex. Since the graph is finite, the path from every vertex leads after several steps to a cycle.

Then Z is the length of the path from vertex $alph(w)$ to the beginning of the cycle and the length of this cycle shall stand for C. Hence the equation (3) follows directly from the definition of the graph. □

Let us now note that the infinite word Pow_k does not contain two blocks of the same length. This leads to another useful property of the morphism, as it is stated in following two lemmas.

Lemma 4.2. *Let h be a morphism generating a marked infinite word w consisting of only finite number of blocks of the length l for every $l \in \mathbb{N}$. Let v be some factor of w occurring infinitely many times in w. Then for any positive integer i the word $h^i(v)$ contains at most one symbol #.*

Proof. The claim is trivial. □

Lemma 4.3. *Let w be a marked infinite word generated by substitution (h, c) on the letter S, consisting of only finite number of blocks of the length l for every $l \in \mathbb{N}$. Let v be some factor of the word $h^\omega(S)$ occurring infinitely many times in $h^\omega(S)$. Then for any positive integer i the word $h^i(v)$ contains at most one letter that is translated to the symbol # by the coding c.*

Proof. Follows directly from previous lemma. □

Lemma 4.4. *For any positive integer P, the polynomials*

$$\begin{gathered}(x+P)^k - x^k \\ (x+2P)^k - x^k \\ (x+3P)^k - x^k \\ \vdots \\ (x+kP)^k - x^k\end{gathered}$$

are linearly independent.

Proof. The degree of each of these polynomials is $k-1$ and its coefficients are written into the rows of a matrix:

$$\begin{pmatrix} kP & \binom{k}{2}P^2 & \binom{k}{3}P^3 & \dots & P^k \\ 2kP & 4\binom{k}{2}P^2 & 8\binom{k}{3}P^3 & \dots & 2^kP^k \\ 3kP & 9\binom{k}{2}P^2 & 27\binom{k}{3}P^3 & \dots & 3^kP^k \\ \vdots & \ddots & & & \\ \vdots & & \ddots & & \\ \vdots & & & \ddots & \\ kkP & k^2\binom{k}{2}P^2 & k^3\binom{k}{3}P^3 & \dots & k^kP^k \end{pmatrix}$$

This matrix can be easily reduced to well-known symmetric Vandermonde's matrix (see [BL67]), and therefore these k polynomials are linearly independent. □

With the above lemmas we can prove the final theorem of this section.

Theorem 4.2. *The infinite word Pow_k cannot be generated by a substitution h, c where $|\Sigma_h| < k+1$.*

Proof. Let there be a CD0L system (h, c) generating the infinite word Pow_k where the control word $w = h^\omega(S)$ is over Σ where $|\Sigma| = r < k+1$. Let us divide the letters of the alphabet into two sets Σ_a and $\Sigma_\#$ according to the letter to which they are translated by coding c. We will call all letters of the set $\Sigma_\#$ simply *hashes* and the letters of the set Σ_a *a-letters*.

Let the last occurrence of the symbol appearing only finitely many times in the control word w be in the position N of the word w. Then let us fix as u some factor of w appearing on the tape right to this symbol (i.e. in

the position greater than N) such that u is written on tape between reading and writing head at some time of the generation. That means that all letters of u occur infinitely many times in w and since word u is written between heads at some step of generation, the writing head will follow with writing the iterated morphic images of u on the tape so that the rest of the word will be

$$uh(u)h^2(u)h^3(u)\dots$$

According to Lemma 4.1 there are positive integers Z_a, C_a for each letter a occurring in the word u such that Z_a, C_a satisfy the equation (3). Let Z and C be integers defined as follows:

$$Z = max\{Z_a|\ a \text{ is a letter of the word } u\}$$

and

$$C = lcm^2\{C_a|\ a \text{ is a letter of the word } u\}$$

Then the condition

$$alph(h^{Z+jC+l}(a)) = alph(h^{Z+l}(a)),\ \text{for all } j, l$$

holds for every letter a of the word u.

This means if $u = u_1u_2\dots u_t$, with $u_i \in \Sigma$, then for any positive integer l the words

$$h^{Z+l}(u_i), h^{Z+l+C}(u_i), h^{Z+l+2C}(u_i), h^{Z+l+3C}(u_i), \dots \tag{4}$$

consist of the same letters.

Moreover, since the letter u_i is occurring infinitely many times in the word w, we obtain from Lemma 4.3 that either there is exactly one occurrence of hash in each of the words of the sequence (4), or there are no hashes within these words.

Consequently, for any positive integer l there is the same number of hashes in each of the words

$$h^{Z+l}(u), h^{Z+l+C}(u), h^{Z+l+2C}(u), h^{Z+l+3C}(u), \dots \tag{5}$$

We shall now consider two cases:

Case 1: There is an integer $n \geq Z$ such that $h^n(u)$ contains at least two hashes. Consequently, also the words

$$h^n(u), h^{n+C}(u), h^{n+2C}(u), \dots \tag{6}$$

[2]the least common multiple

contain at least two hashes.

Let us fix the first two occurrences of hashes in the word $h^n(u)$, say $\#_1$ and $\#_2$. These are delimiting a block of a-letters that will be in the end translated by the coding c to some block of the infinite word Pow_k, let it be the s-th block.

The hash $\#_1$ is a symbol of the image $h^n(u_p)$ for some letter u_p of u and $\#_2$ is a letter of $h^n(u_q)$ for some letter u_q of u and moreover, $p \neq q$. This follows directly from Lemma 4.3 and the fact that all letters in u are occurring infinitely many times in w.

Note that the first two occurrences of hashes in all words of the sequence (6) are again letters belonging to the images of the letters u_p and u_q, therefore these hashes are again symbols $\#_1$ and $\#_2$. But since for any l there is the same number of hashes in the words of the sequence (5), there is a positive integer M such that the block delimited by these symbols $\#_1$ and $\#_2$ in the word $h^{n+jC}(u)$ is coded by c to $(s+jM)$-th block of the infinite word Pow_k.

Now let us concentrate on the sequence of words

$$h^n(u_p), h^{n+C}(u_p), h^{n+2C}(u_p), h^{n+3C}(u_p), \ldots \tag{7}$$

We already noted that there is exactly one hash in each of these words, concretely the symbol $\#_1$. But this hash has originated from some letter e_1, i.e. $h^C(e_1)$ contains the letter $\#_1$. Since there are the same letters in these words, also the symbol e_1 must occur exactly once in each of these words. Using the analogous reasoning we can build a sequence

$$\ldots\ e_3 \xrightarrow[h^C]{} e_2 \xrightarrow[h^C]{} e_1 \xrightarrow[h^C]{} \#_1, \tag{8}$$

i.e. e_i belongs to the image $h^C(e_{i+1})$ and must occur exactly once in each word of the sequence (7).

Since number of different letters in Σ is only finite and the morphism is 'deterministic', there must be a cycle within the derivation path (8), i.e. there is a symbol $e = e_z$ for some z and a positive integer g such that $h^{gC}(e)$ contains both symbols $\#_1$ and e.

By analogous method we prove that there is a symbol d and a positive integer g' such that the symbol d occurs in each of the words

$$h^n(u_q), h^{n+C}(u_q), h^{n+2C}(u_q), h^{n+3C}(u_q), \ldots$$

exactly once and moreover, $h^{g'C}(d)$ contains both symbols $\#_2$ and d.

Now let us put $P = gg'C$ and consider the sequence of words

$$h^n(u), h^{n+P}(u), h^{n+2P}(u), h^{n+3P}(u), \ldots \tag{9}$$

Since (9) is a subsequence of the sequence (6) it satisfies similar conditions. Concretely, each of these words contains the same number of hashes and there is a positive integer M' such that the first two hashes of $h^{n+jP}(u)$ are delimiting the block of a-letters translated to the $(s + jM')$-th block of the word Pow_k.

Let y_i denote the word delimited by e and d in the word $h^{n+iP}(u)$. Then translating the word ey_id by the morphism h^P will produce the word $qey_{i+1}dq'$ where $q, q' \in \Sigma^*$.

Moreover, since in each of the words of the sequence (9) the symbols $\#_1$ and e ($\#_2$ and d, respectively) belong to the image $h^P(e)$ ($h^P(d)$, respectively), the distance between these symbols in all words of the sequence (9) is the same.

Let us now consider the vectors $\vec{y_i}$ assigned to the words y_i. It is obvious that there exists a matrix A which is a matrix of transformation of the vector $\vec{y_i}$ to the $\vec{y_{i+1}}$ for any $i \in \mathbb{N}$.

Note that there is no occurrence of symbols equivalent to e or d in the words $y_i, i \in \mathbb{N}$, therefore the components of the vectors $\vec{y_i}$ corresponding to these letters are equal to zero. Therefore for fixed $i \in \mathbb{N}$ the k vectors $\vec{y_{i+1}} - \vec{y_i}, \vec{y_{i+2}} - \vec{y_i}, \ldots, \vec{y_{i+k}} - \vec{y_i}$ are linearly dependent, i.e.

$$(\exists a_1, a_2, \ldots, a_k \neq 0) \quad a_1(\vec{y_{i+1}} - \vec{y_i}) + \ldots + a_k(\vec{y_{i+k}} - \vec{y_i}) = 0 \tag{10}$$

Furthermore, since $\vec{y_{i+1}} = \vec{y_i}A$, we get that the equation (10) holds for each $i \in \mathbb{N}$ with the same coefficients a_j's, i.e.

$$(\exists a_1, a_2, \ldots, a_k \neq 0)(\forall i \in \mathbb{N}) \quad a_1(\vec{y_{i+1}} - \vec{y_i}) + \ldots + a_k(\vec{y_{i+k}} - \vec{y_i}) = 0$$

Moreover, this equation holds also for norms of these vectors:

$$(\exists a_1, a_2, \ldots, a_k \neq 0)(\forall i \in \mathbb{N}) \quad a_1\|\vec{y_{i+1}} - \vec{y_i}\| + \ldots + a_k\|\vec{y_{i+k}} - \vec{y_i}\| = 0$$

Let us now note that the expression

$$\|\vec{y_{i+r}} - \vec{y_i}\|, \ 1 \leq r \leq k \tag{11}$$

denotes the difference in length between the words y_{i+r} and y_i. Since the distance between the symbol $\#_1$ and e ($\#_2$ and d, resp.) is same in all words

of the sequence (9), the expression (11) represents also the difference in length between the $(s+(i+r)M')$-th and the $(s+iM')$-th block of the word Pow_k.

Therefore, for each $i \in \mathbb{N}$ the norms of the vectors $\vec{y_{i+1}} - \vec{y_i}, \vec{y_{i+2}} - \vec{y_i}, \ldots, \vec{y_{i+k}} - \vec{y_i}$ are equal to the polynomials

$$\begin{gathered}(i+M')^k - i^k \\ (i+2M')^k - i^k \\ (i+3M')^k - i^k \\ \vdots \\ (i+kM')^k - i^k\end{gathered}$$

But according to Lemma 4.4 these polynomials are linearly independent which is a contradiction with (11).

Case 2: For any integer $n \geq Z$ the morphic image $h^n(u)$ contains at most one hash. This can be easily reduced to the first case:

First, there must be integers $t > s \geq Z$ such that both $h^s(u), h^t(u)$ contain exactly one hash, otherwise following from some position of the tape the writing head would be writing only a-letters, a contradiction with the definition of the word Pow_k.

Then if we substitute the word u for the word

$$h^s(u)h^{s+1}(u)\ldots h^t(u)$$

and the morphism h for the morphism h^{t-s+1} the proof follows as in the first case. □

5 Number of states of a DGSM

In this section we present the result that there is a dense infinite hierarchy within the class of DGSM words – based on the number of states of the iterated DGSM. Actually, this is the similar result to the one presented for the class of CD0L words in the previous chapter, since the notion of state is equivalent to that of a letter of the control tape of the DGSM TAG system.

We shall need following class of binary marked words over $\Sigma = \{a, \#\}$: Let us fix a positive integer k and define an infinite word

$$Lin_k = Sc_1\#c_2\#c_3\#\dots$$

where $c_n = w_{1,n}\#w_{2,n}\#\dots\#w_{k,n} = a^n\#a^{2n}\#a^{3n}\#\dots\#a^{kn}$. We shall call the words c_n *components* of Lin_k and its subwords $w_{i,n}$ delimited by symbols $\#$ the *units* of c_n. We say that the unit $w_{i,n}$ *holds a position* i in the word c_n.

Thus, explicitly the infinite word Lin_k looks like follows:

$$Lin_k = Sa\#a^2\#\dots\#a^k\#a^2\#a^4\#\dots\#a^n\#a^{2n}\dots\#a^{kn}\#a^{n+1}\dots$$

Theorem 5.1. *For each positive integer k there exists a DGSM TAG system M_k having exactly k states which generates the infinite word Lin_k.*

Proof. Clearly, the word Lin_k is obtained by a deterministic DGSM TAG system M_k with states $q_1, q_2, \dots q_k$ and the following set of transition rules:

$$\begin{pmatrix} S \\ q_1 \end{pmatrix} \to \begin{pmatrix} Sa\#a^2\#a^3\dots\#a^k\# \\ q_1 \end{pmatrix}$$
$$\begin{pmatrix} a \\ q_i \end{pmatrix} \to \begin{pmatrix} a \\ q_i \end{pmatrix}, \text{ for each } i \in \{1, 2, \dots, k\}$$
$$\begin{pmatrix} \# \\ q_i \end{pmatrix} \to \begin{pmatrix} a^i \\ q_{(i+1) \bmod k} \end{pmatrix}, \text{ for each } i \in \{1, 2, \dots, k\}$$

□

Similarly as in the proof of Theorem 4.2 we will prove there is some kind of regularity in the generation of the word Lin_k which leads to the following result.

Theorem 5.2. *The infinite word Lin_k cannot be generated by a DGSM TAG system having less than k states.*

Proof. Suppose there is a DGSM TAG system M generating the infinite word Lin_k and the number of states of M is lower than k. Let Q denote the set of those states of GSM M that are read from the control tape infinitely many times at those steps of computation when there is a symbol # read from the generating tape. Note that since both heads move synchronously these states are represented by letters written 'under' the symbol # on the control tape.

If we generalize the notion of a rewriting rule from translating one letter to translating a word $w \in \Sigma^+$ we get that there is a positive integer B such that a rewriting rule for each state $q_s \in Q$ and for each $i \geq B$ must be of a form:

$$\begin{pmatrix} \#a^i \\ q_s \end{pmatrix} \rightarrow \begin{pmatrix} a^x \#a^y \\ q_z \end{pmatrix}, \text{ for some } x, y \in \mathbb{N} \text{ and } q_z \in K \tag{12}$$

This is obtained as follows. First, since a rule of this form is applied infinitely many times for q_s (differing only in number of a's following the symbol #), there must be at most one occurrence of letter # on the right-hand side of (12). Otherwise there would be infinitely many blocks with the same length in the word Lin_k which is obviously a contradiction.

In fact, there must be exactly one occurrence of # on the right side of the rewriting rule (12) or else the number of letters # written between reading and writing head would decrease in time which is simply false.

That means that from some point of the computation the number of hashes between heads does not vary in time and therefore there are positive integers t, n_1 such that every n-th block of the word Lin_k where $n \geq n_1$ is with some delay translated to the $(n+t)$-th block by the DGSM TAG system M.

Thus translating i-th unit of the n-th component will be as follows:

$$\begin{aligned} w_{1,n} &\rightarrow w_{p,n+r} \\ w_{2,n} &\rightarrow w_{p+1,n+r} \\ &\cdots \\ w_{i,n} &\rightarrow w_{k,n+r} \\ w_{i+1,n} &\rightarrow w_{1,n+r+1} \\ &\cdots \\ w_{k,n} &\rightarrow w_{k-i,n+r+1} \end{aligned}$$

But since the GSM M is non-erasing t must be a multiple of k. Otherwise there would be a block with an image shorter than the block itself. Indeed,

if for any n the unit $w_{j,n}$ is translated to the unit $w_{1,n+r+1}$ where $j \neq 1$, then putting $n \geq \frac{r+1}{i-1}$ yields a contradiction.

Let us now consider the word

$$v = a^{n.k!}, n \geq n_1$$

This is obviously a factor of Lin_k. Furthermore,

$$v = w_{i,\frac{n.k!}{i}}, \text{ for each } i \in \{1, 2, \ldots, k\}$$

which means that for each position i there is a component in which v is the unit holding the position i. Since the number of states of the GSM M is lower than k, there are at least 2 of these positions where the left delimiting hash is read in the same state. Let it be the positions c, d.

We have already shown that following from the n_1-th block each unit $w_{i,n}$ is translated to the unit $w_{i,(n+\frac{t}{k})}$ and therefore $w_{c,\frac{n.k!}{c}}$ must be translated to $w_{c,(\frac{n.k!}{c}+\frac{t}{k})}$ and $w_{d,\frac{n.k!}{d}}$ to $w_{d,(\frac{n.k!}{d}+\frac{t}{k})}$. But this is a contradiction with the fact that the GSM M is deterministic. □

6 Conclusion

In this thesis we investigated various mechanisms generating infinite words. The most used mechanism is iterating a morphism. We concentrated on two basic extensions of this method – coding the morphic word after its generation, namely CD0L TAG system; and using more powerful mechanism in the iteration, namely DGSM TAG system.

We proved two main results:

- There is a dense hierarchy within the class of binary CD0L words – depending on the cardinality of the control alphabet.
- There is a dense hierarchy within the class of DGSM words – depending on the number of states of the iterated deterministic GSM.

Since the notion of state of a deterministic GSM is equivalent to the notion of the letter of a control alphabet of a TAG machine, these are the same results for classes binary CD0L and DGSM.

It has to be emphasized that there seems to be very few results of nature: "Certain infinite word cannot be generated by certain mechanism." We hope that ideas from proofs of theorems 4.2 and 5.2 may be useful in searching for concrete words that cannot be generated by CD0L and DGSM TAG systems.

Finally, we would like to refer to one open problem of this type, presented in [CK94] – namely, whether there is an infinite word generated by morphisms iterated periodically which cannot be generated by any CD0L TAG system.

References

[BL67] C. Birkhoff and S. Mac Lane, Algebra. *New York: MacMillan,* 1967.

[BK03] J. Berstel and J. Karhumäki, Combinatorics on Words - A Tutorial, *Bull. EATCS 79*. 2003.

[CK94] K. Culik II and J. Karhumäki, Iterative devices generating infinite words, *Int. J. Found. Comput. Sci.* 5, 1994.

[CK97] C. Choffrut and J. Karhumäki, Combinatiorics of words, In: A. Salomaa and G. Rozenberg (eds.), *Handbook of Formal Languages, Vol. 1.* Springer-Verlag, 1997.

[DM03] P. Ďuriš and J. Manuch, On the computational complexity of infinite words, *Theoret. Comput. Sci.* 1-3, 2003.

[HU79] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[HKL94] J. Hromkovič, J. Karhumäki, and A. Lepistö, Comparing descriptional and computational complexity of infinite words, In: J. Karhumäki, H. Mauer, and G. Rozenberg (Eds., *Results and Trends in Theoretical Computer Science* LNCS 812. Springer-Verlag, 1994.

[Kn72] D. Knuth: Solution to Problem E 2307, *Amer. Math. Monthly 86*, 1972.

[Lo81] M. Lothaire: *Combinatorics on Words.* Addison-Wesley, 1981.

[Lo02] M. Lothaire: *Algebraic Combinatorics on Words. Encyclopedia of Mathematics 90*, Cambridge University Press, 2002.

[Mi67] M.L. Minsky, *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[Sh88] J. Shallit, A Generalization of Automatic Sequences, *Theoretical Computer Science* 61, 1988.

[Th06] A. Thue, Über unendliche Zeichenreihen, *Norske Vid. Selsk. Skr. I. Mat. Nat. Kl., Christiana* 7, 1906.

Resumé

Táto diplomová práca je venovaná zložitosti nekonečných slov, ktorá je popisovaná generatívnou silou rôznych výpočtových mechanizmov. Najznámejším a najpoužívanejším z týchto mechanizmov je iterovanie homomorfizmu. Sú dve základné možnosti rozšírenia tejto metódy:

- Aplikovať nejakú ďalšiu metódu na nekonečné slovo po jeho vygenerovaní iterovaním homomorfizmu.
- Používať v iterovanom procese silnejšie zobrazenie, ako je homomorfizmus.

Substitúcia (CD0L-TAG systém), je príkladom rozšírenia prvého typu. Pri tejto metóde sa nekonečné slovo vygenerované iterovaním homomorfizmu preloží kódovaním na výsledné nekonečné slovo. Príkladom druhého typu rozšírenia je DGSM TAG systém, v ktorom sa iteruje deterministické GSM. Existujú mnohé iné rozšírenia metódy iterovania homomorfizmu, ktoré sú zjednotené na báze TAG systémov (D0L s periodickou kontrolou, double D0L, DGSM, atď.).

Hlavnými výsledkami tejto práce sú dve tvrdenia:

- V triede binárnych nekonečných slov generovaných CD0L TAG systémami je maximálne hustá hierarchia vzhľadom na počet písmen kontrolnej abecedy (abecedy iterovaného homomorfizmu).
- V triede DGSM – slov generovaných iterovaním deterministického GSM je maximálne hustá hierarchia vzhľadom na počet stavov iterovaného GSM.

Keďže počet stavov GSM je ekvivalentný mohutnosti kontrolnej abecedy TAG systému, hovoria tieto dva tvrdenia o tej istej vlastnosti daných tried.

Diplomová práca je zoradená nasledovne:
Úvodná časť je venovaná rôznym pohľadom na zložitosť nekonečných slov a oboznámeniu s obsahom práce.
Druhá časť pozostáva z definícií a základnej terminológie z oblasti teórie slov a jazykov.
Tretia časť popisuje základné výpočtové modely generujúce nekonečné slová

a príklady konkrétnych nekonečných slov, ktoré tieto modely generujú. Tieto modely definujú triedy nekonečných slov, vzťahy medzi ktorými sú uvedené v závere tejto časti.
V štvrtej časti je dokázaný výsledok o existencii hustej hierarchie v triede binárnych CD0L slov.
V piatej časti je dokázaný výsledok o existencii hustej hierarchie v triede DGSM slov.
V závere práce poukazujeme na otvorené problémy, ku riešeniu ktorých môže táto práca pomôcť.