Deterministic Frequency Pushdown Automata

C. S. Calude[1], R. Freivalds[2], F. Stephan[3]

[1]University of Auckland, NZ
[2]University of Latvia, Riga, Latvia
[3]National University of Singapore, Singapore

CDMTCS-441
September 2013

Deterministic Frequency Pushdown Automata

Cristian S. Calude[1], Rūsiņš Freivalds,[2***] and Frank Stephan[3]

[1] Department of Computer Science, The University of Auckland
Auckland, New Zealand
cristian@cs.auckland.ac.nz

[2] Institute of Mathematics and Computer Science, University of Latvia
Riga, Latvia
Rusins.Freivalds@mii.lu.lv

[3] School of Computing, National University of Singapore
Singapore
fstephan@comp.nus.edu.sg

Abstract. Frequency computation was introduced by Rose [26]. Trakhtenbrot [27] proved the existence of a continuum of functions computable by frequency Turing machines with frequency $\frac{1}{2}$. In contrast, every function computable by a frequency Turing machine with frequency exceeding $\frac{1}{2}$ is recursive. Essentially similar results for finite automata and other types of machines have been proved by Kinber [19] and Austinat, Diekert, Hertrampf, Petersen [4].
In this paper we introduce the notion of frequency pushdown automaton. Answering a question E. Shamir posed at *LATA 2013*, we prove that there is a language which is $\frac{1}{1}$-recognizable but which is not $\frac{2}{n}$-recognizable (for any n) by deterministic frequency pushdown automaton.

1 Introduction

During a discussion of the paper [14] at the conference *LATA 2013* in Bilbao, Spain, E. Shamir asked the whether the results on frequency Turing machines and frequency finite automata hold for pushdown automata as well. The difficulty of the question is in the fact that an (n, n)-Turing machine or an (n, n)-finite automaton can be presented as a Cartesian product of n separate n separate Turing machines or finite automata and this construction does not seem to increase the power of the machine. However, an arbitrary Turing machine can be simulated by an automaton with 3 pushdown tapes [6]. Hence the definition of frequency pushdown automata should avoid the use of several pushdown stores in a single automaton.

The notion of frequency computation was introduced by Rose [26] as an attempt to have a deterministic notion of computation with properties similar to probabilistic algorithms. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of all natural

* Supported by Project 271/2012 from the Latvian Council of Science.
** Corresponding author.

numbers, $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. Fix $m, n \in \mathbb{N}, 1 \leq m \leq n$. The ith component of the m-tuple $(x_1, \cdots, x_m)$ is denoted by $(x_1, \cdots, x_m)_i$.

A function $f : \mathbb{N} \to \mathbb{N}$ is (m, n)-*computable* if there exists a recursive function $R \colon \mathbb{N}^n \to \mathbb{N}^n$ such that for all n-tuples $(x_1, \cdots, x_n) \in \mathbb{N}^n$ of mutually distinct natural numbers we have:

$$\mathrm{card}\{i \mid (R(x_1, \cdots, x_n))_i = f(x_i)\ ,\ 1 \leq i \leq n\} \geq m.$$

Answering a problem by Myhill (see McNaughton [25]), Trakhtenbrot [27] proved that an (m, n)-computable f is recursive iff $2m > n$; on the other hand, if $2m = n$, then f can be not recursive. Kinber [20, 19] extended the results by considering frequency enumeration of sets and proved that the class of (m, n)-computable sets equals the class of recursive sets if and only if $2m > n$.

The notion of frequency computation has been extended to other models of computation. Frequency computation in polynomial time was discussed in full detail by Hinrichs and Wechsung [16]. For resource bounded computations, the behavior of frequency computability is completely different. For example, under any reasonable resource bound, whenever $n' - m' > n - m$ there exist sets which are (m', n')-computable, but not (m, n)-computable. However, scaling down to finite automata, the analogue of Trakhtenbrot [27] result holds again: the class of languages (m, n)-recognizable by deterministic frequency automata equals the class of regular languages if and only if $2m > n$; conversely, for $2m \leq n$, the class of languages (m, n)-recognizable by deterministic frequency automata is uncountable for a two-letter alphabet (cf. Austinat, Diekert, Hertrampf, Petersen [4]). When restricted to a one-letter alphabet, every (m, n)-recognizable language is regular (cf. Kinber [20] and Austinat, Diekert, Hertrampf, Petersen [4]).

Frequency computations became increasingly popular when relations between frequency computation and computation with a small number of queries was discovered [1, 4, 5, 7, 10, 11, 12, 15, 21, 22, 23, 24].

2 Frequency pushdown automata

Let Σ be any finite alphabet, and let Σ^* be the free monoid generated by Σ. The binary alphabet $\mathbb{B}$ is denoted by $\mathbb{B}$. Every subset $L \subseteq \Sigma^*$ is said to be a *language*. The elements of Σ^* are called *strings*; $|x|$ denotes the *length of a string* $x \in \Sigma^*$. By $\chi_L \colon \Sigma^* \to \mathbb{B}$ we denote the *characteristic function* of L.

A *1-deterministic pushdown automaton (PDA)* is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ where Q is a finite set of states, Σ is a finite set which is called the input alphabet, Γ is a finite set which is called the stack alphabet, $q_0 \in Q$ is the start state, $Z \in \Gamma$ is the initial stack symbol, and $F \subseteq Q$ is the set of accepting states. An element $(p, a, A, q, \alpha) \in \delta$ is a transition of M: in state $p \in Q$ and input $a \in \Sigma$ and with $A \in \Gamma$ as topmost stack symbol, M may read a, change the state to q, pop A, replace it by pushing $\alpha \in \Gamma^*$. Hence the transition function is defined as follows: $\delta \colon Q \times \Sigma \times (\Gamma \cup \{\varepsilon\}) \to Q \times (\Gamma \cup \{\varepsilon\})$.

For n-frequency pushdown automata we modify the above definition allowing n input words. However, we need to be aware that for the general case input words can be of distinct lengths. Our definition closely models the definition of n-frequency finite automata (see, e.g. [14]). An *n-deterministic frequency automaton (n-DFA)* is a 7-tuple $\mathcal{A} = [Q, \Sigma, \#, \delta, q_0, \tau, n]$, where $n \in \mathbb{N}$, $n \geq 1$, Q is a finite set of states, q_0 is the initial state, Σ is a finite alphabet and $\#$ is a symbol not in Σ. The mapping $\delta \colon Q \times (\Sigma \cup \{\#\})^n \to Q$ is the transition function; the function $\tau \colon Q \to \mathbb{B}^n$ is the type of state which is used for outputs. The type is interpreted as an n-tuple of answers α_i: its i-th component records whether the i-th input word read from the i-th input up to the current moment belongs to the language. We use the notation $\tau(q, (x_1\#^{\ell_1}, \dots, x_n\#^{\ell_n}))$ to denote the type after reading the inputs the words $(x_1\#^{\ell_1}, \dots, x_n\#^{\ell_n})$.

Next we formally describe the behavior of an n-DFA $\mathcal{A}$. Let $n \in \mathbb{N}_+$, and let $x = (x_1, \dots, x_n) \in (\Sigma^*)^n$ be an input vector. We define $|x| = \max\{|x_i| \mid 1 \leq i \leq n\}$, and $q \circ x = \delta^*(q, (x_1\#^{\ell_1}, \dots, x_n\#^{\ell_n}))$, where $\delta^* \colon Q \times ((\Sigma \cup \{\#\})^n)^*$ is the usual extension of δ on n-tuples of strings, and $\ell_i = |x| - |x_i|$ for all $1 \leq i \leq n$. The output of $\mathcal{A}$ is defined to be the type $\tau(q_0 \circ x)$.

A language $L \subseteq \Sigma^*$ is said to be *(m, n)-recognized* by an n-DFA $\mathcal{A}$ if for each n-tuple $(x_1, \dots, x_n) \in (\Sigma^*)^n$ of pairwise distinct strings the tuples $\tau(q_0 \circ x)$ and $(\chi_L(x_1), \dots, \chi_L(x_n))$ coincide on at least m components. A language $L \subseteq \Sigma^*$ is called *(m, n)-recognizable* if there is an n-DFA $\mathcal{A}$ that (m, n)-recognizes L.

To define n-deterministic frequency pushdown automata (with only one pushdown store) the transition function will be extended for n-tuples $\delta^* \colon Q \times \Sigma^n \times (\Gamma \cup \{\varepsilon\}) \to Q \times (\Gamma \cup \{\varepsilon\})$.

An *n-deterministic frequency pushdown automaton (n-DFPA)* is an 8-tuple $\mathcal{A} = (Q, \Sigma, \#, \Gamma, \delta, q_0, Z, F)$, where $\# \notin \Sigma$ and $(Q, \Sigma \cup \{\#\}, \Gamma, \delta, q_0, Z, F)$ is a PDA.

Let $n \in \mathbb{N}_+$, and let $x = (x_1, \dots, x_n) \in (\Sigma^*)^n$ be an n-tuple. We define $|x| = \max\{|x_i| \mid 1 \leq i \leq n\}$, and

$$q \circ x = \delta^*(q, (x_1\#^{\ell_1}, \dots, x_n\#^{\ell_n})),$$

where $\ell_i = |x| - |x_i|$ for all $1 \leq i \leq n$. Then the output of $\mathcal{A}$ is defined to be the type $\tau(q_0 \circ x)$. We emphsize that the n-DFPA contains only one pushdown tape which is used to process all n inputs.

A language $L \subseteq \Sigma^*$ is said to be *(m, n)-recognized* by an n-DFPA $\mathcal{A}$ if for each n-tuple $(x_1, \dots, x_n) \in (\Sigma^*)^n$ of pairwise distinct strings the tuples $\tau(q_0 \circ x)$ and $(\chi_L(x_1), \dots, \chi_L(x_n))$ coincide on at least m components. A language $L \subseteq \Sigma^*$ is called *(m, n)-recognizable* if there is an n-DFPA $\mathcal{A}$ that (m, n)-recognizes L.

3 Results

We start with the following obvious facts.

Theorem 1. *Assume that a language L is (m, n)-recognisable by a 2-DFPA. Then:*

(a) The language L is $(m, n+1)$-recognisable by a 2-DFPA.
(b) If $m > 1$ then L is $(m-1, n-1)$-DFPA recognisable.

By Theorem 1, if L is (m, n)-recognisable by a 2-DFPA, then L is $(m-1, n)$-recognisable by a 2-DFPA and for every $k < n$, L is also (k, k)-recognisable by a 2-DFPA.

Proposition 3 in Austinat, Diekert, Hertrampf, Petersen ([4] proved for DFAs is true for DFPAs as well.

Theorem 2. *A continuum of languages are $(1, 2)$-recognizable by a 2-DFPA.*

Proof. We define the following ordering $\leq$ of words in $\mathbb{B}^*$. Let $\lambda = \lambda_1, \lambda_2, \cdots$ and $\mu = \mu_1, \mu_2, \cdots$ be arbitrary words. We say that $\lambda \leq \mu$ if either $(\lambda_1 < \mu_1)$ or $(\lambda_1 = \mu_1) \wedge (\lambda_2 < \mu_2)$ or $(\lambda_1 = \mu_1) \wedge (\lambda_2 = \mu_2) \wedge (\lambda_3 < \mu_3)$ or, etc.

Let $\beta \in \mathbb{B}^*$ and construct the language $L_\beta = \{\lambda \in \mathbb{B}^* \mid \beta \leq \lambda\}$. Notice that if β_1 and β_2 are distinct words and $\beta_1 \leq \beta_2$ then $\beta_2 \in L_{\beta_2}$ but $\beta_2 \notin L_{\beta_1}$.

For every $\beta \in \mathbb{B}^*$ there exists a 2-DFPA $(1, 2)$-recognizing the language L_β. Indeed, when the 2-DFPA works on the pair (λ_1, λ_2), the automaton finds out whether $\lambda_1 \leq \lambda_2$ or $\lambda_2 \leq \lambda_1$. In the first case the automaton accepts λ_2 and rejects λ_1. In the second case the automaton accepts λ_1 and rejects λ_2.

If $\lambda_1 \leq \lambda_2$, then because of the definition of the language L_β only three (out of four) cases are possible, namely: *Case* (a). $\lambda_1 \notin L_\beta$ and $\lambda_2 \notin L_\beta$, *Case* (b). $\lambda_1 \notin L_\beta$ and $\lambda_2 \in L_\beta$ *Case* (c). $\lambda_1 \in L_\beta$ and $\lambda_2 \in L_\beta$.

In all three cases at least one of the result is correct.

□

For the proof of the subsequent Theorem 3 we need the following auxiliary notions.

Definition 1. *Let L be a language and x a word. We say that L_x is a* suffix language of L corresponding to the word x *if $L_x = \{z \mid xz \in L\}$.*

Definition 2. *Let L be a language. We say that a word x is* L-suffix-equivalent to *the word y if $L_x = L_y$.*

Next we present a lemma proved in Freivalds [13].

Lemma 1. *If L is a non-regular language then there exists an ω-word $w^\infty = a_0a_1a_2\cdots$ such that for any distinct prefixes x and y of w^∞ the languages L_x and L_y are distinct.*

Proof. By $\mathcal{L}$ we denote the family of all the languages L_x, $x \in \Sigma^*$. Let $w^\infty = a_0a_1a_2\cdots$ and consider the family

$$\mathcal{L}(w^\infty) = \{L_{a_0a_1a_2\cdots a_m} \mid m \in \mathbb{N}_+\}.$$

Had $\mathcal{L}$ been finite, then the language L would be regular. If $\mathcal{L}(w^\infty)$ is infinite, the exist infinitely many prefixes of w^∞, $y = a_0a_1a_2\cdots a_m, z = a_0a_1a_2\cdots a_n$

such that $m < n$, $L_y \neq L_z$. If u^∞ is obtained from w^∞ by removing the symbols $a_{m+1} \cdots a_n$ then $\mathcal{L}(u^\infty)$ is also infinite. Informally, the proof consist of repeating this procedure infinitely often. Formal proof, however, uses König's lemma [17]. □

Lemma 2. *If L is a non-regular language then there exist two ω-words $w^\infty = a_0a_1a_2\cdots$ and $u^\infty = b_0b_1b_2\cdots$ having the property of Lemma 1.*

Proof. If $w^\infty = a_0a_1a_2\cdots$ is an ω-word with the property in Lemma 1, then the ω-word $u^\infty = a_1a_2a_3\cdots$ also has the property in Lemma 1. If $w^\infty = a_0a_1a_2\cdots$ contains at least two distinct elements a_i and a_k, then $u^\infty \neq w^\infty$. A language L for which only single-letter ω-words $w^\infty = a_0a_1a_2\cdots$ have the property of Lemma 1, is regular. □

Lemma 3. *Let L be a language $(2,2)$-recognizable by a 2-DFPA. Denote the cardinality of the pushdown alphabet by d and the number of the states of the 2-DFPA by k. Let $y = a_0a_1a_2\cdots a_m$ and $z = a_0a_1a_2\cdots a_n$ be two prefixes of the same ω-word such that $m < n$ and $L_y \neq L_z$. Then there exists a word v of length not exceeding $k \cdot d^n$ such that $yv \in L_y$ iff $zv \notin L_z$.*

Proof. If the state and the content of the pushdown after reading the word y from the first input and some word from the second input and after reading the word z from the first input and some (maybe different) words from the second input are the same, then the pushdown computation on every pair (v,u) produces the same result on the first output. The maximum possible length of the pushdown used during $\max(m,n) = n$ steps is n, hence there can be no more than d^n distinct records in the pushdown. □

Theorem 3. *Every $(2,2)$-recognizable language is regular.*

Proof. By Lemma 1 there exists an ω-word $w^\infty = a_0a_1a_2\cdots$ such that for any distinct prefixes x and y of w^∞ the languages L_x and L_y are distinct.

Assume by absurdity that a non-regular language L is $(2,2)$-recognizable by a 2-DFPA with the the pushdown alphabet of cardinality d. Take 2 distinct ω-words $w^\infty = a_0a_1a_2\cdots$ and $u^\infty = b_0b_1b_2\cdots$ with the property in Lemma 1. Let s be the smallest natural number such that $a_s \neq b_s$ and take an n such that $n > s$. Consider the computation of the 2-DFPA on pairs of inputs $(a_0a_1a_2\cdots, a_n, c_0c_1c_2\cdots, c_n)$, for various words $c_0c_1c_2\cdots, c_n$. By w_g we denote the word $a_0a_1a_2\cdots a_{s+g}$. By u_g we denote the word $b_0b_1b_2\cdots b_{s+g}$.

We consider separately the following two possibilities:

1. There exist at least $\log\log n$ distinct words $w_r \in \{w_1, w_2, \cdots, w_n\}$ such that there exists a pair of words $(e_1e_2\cdots e_g, f_1f_2\cdots f_g)$ for which during the computation the pair $(w_re_1e_2\cdots e_g, u_rf_1f_2\cdots f_g)$ at moments $t \in \{n+1, n+2, \cdots, n+g\}$ the 2-DFPA never reaches the part of the pushdown store recorded at moments $t \in \{1, 2, \cdots, n\}$.

2. With the exception of no more than $\log\log n$ words, for all words $w_r \in \{w_1, w_2, \cdots, w_n\}$ there exists a pair of words $(e_1e_2\cdots e_g, f_1f_2\cdots f_g)$ such that for all pairs of words $(w_re_1e_2\cdots e_g, u_rf_1f_2\cdots f_g)$ at some moment $t \in \{n{+}1, n{+}2, \cdots, n{+}g\}$ the 2-DFPA always reaches the part of the pushdown store recorded at moments $t \in \{1, 2, \cdots, n\}$.

In the first case, there are at least $\log\log n$ distinct words $w_r \in \{w_1, w_2, \cdots, w_n\}$ defining $\log\log n$ distinct suffix languages among $L_1, L_2, \cdots, L_n$, and these languages can be distingushed by words of length not exceeding g. However, our 2-DFA has no information to distinguish these languages. In the second case, the 2-DFA during the time $t \in \{n{+}1, n{+}2, \cdots, n{+}g\}$ has erased too much information in the pushdown to distinguish between the suffix languages concerning the second input. □

To prove the subsequent theorems we need some tools from algorithmic information theory.

Definition 3. *The numbering $\Psi = \{\Psi_0(x), \Psi_1(x), \Psi_2(x), \ldots\}$ of 1-argument partial recursive functions is* computable *if the 2-argument function $U(n, x) = \Psi_n(x)$ is partial recursive.*

Definition 4. *The numbering Ψ is* reducible *to the numbering η if there exists a total recursive function $f(n)$ such that, for all n and x, $\Psi_n(x) = \eta_{f(n)}(x)$.*

Definition 5. *A computable numbering φ of all 1-argument partial recursive functions is a* Gödel numbering *if every computable numbering (of any class of 1-argument partial recursive functions, not only of the class of all partial recursive functions) is reducible to φ.*

Definition 6. *A Gödel numbering ϑ is a* Kolmogorov numbering *if for arbitrary computable numbering Ψ (of any class of 1-argument partial recursive functions) there exist two constants $c > 0, d > 0$, and a total recursive function $f(n)$ such that:*

1. *for all n and x, $\Psi_n(x) = \vartheta_{f(n)}(x)$,*
2. *for all n, $f(n) \leq c \cdot n + d$.*

Theorem 4 (Kolmogorov Theorem, [3]). *There exists a Kolmogorov numbering.*

Definition 7. *Let ϑ be a Kolmogorov numbering. A word $w \in \mathbb{B}^n$ has ϑ-*complexity *r if:*

1. *$\vartheta_r(0) = w$,*
2. *$\vartheta_q(0) \neq w$, for all $q < r$.*

Definition 8. *A word $w \in \mathbb{B}^n$ has maximal ϑ-complexity if:*

1. *$w \in \mathbb{B}^n$ has ϑ-complexity r,*
2. *every $v \in \mathbb{B}^n$ distinct from w has ϑ-complexity less than r.*

Definition 9. *A pair of words (x, z) has ϑ-complexity r if:*

1. *$\vartheta_r(0) =< x, z >$,*
2. *for every prefix y of the word x and for every prefix u of the word z the pair of words (L_y, L_u) has ϑ-complexity less than r.*

Kolmogorov complexity has been used many times to describe the degree of incompressibility of an object [2, 8, 18]. The following characterization of the notion of algorithmic independence [9] for words will be used in what follows.

Definition 10. *A pair of words (x, z) has maximal ϑ-complexity if there is a natural number r such that:*

1. *(x, z) has ϑ-complexity r,*
2. *for every prefix y of the word x and for every prefix u of the word z the ϑ-complexity of the pair of (y, u) is less than r.*

The main idea of the proof of the subsequent Theorem 5 is to consider a language which can be recognized by a DPA using essentially the pushdown. We use $PALINDROMES = \{w \in \mathbb{B}^* \mid w = w^{rev}\}$ (w^{rev} is the reversal of w) for such a language. Since a $(2, 2)$-DFPA has to produce correct results for two inputs but the automaton has only one pushdown store, the main idea of the proof is look at the computation of an *independent* pair of words and get a contradiction with the assumption of $(2, 2)$-recognizability of the language. To this aim we consider several useful notions and introduce several auxiliary functions.

By $\alpha(k)$ we denote the distance m between the initial stack symbol Z_0 and the current position Z_m of the head of the pushdown at the moment k. By $\beta(k) = \alpha(2n - k + 1)$ we denote the distance m between the initial stack symbol Z_0 and the current position Z_m of the head of the pushdown at the moment $2n - k + 1$. By $\gamma(k)$ we denote the distance m between the initial stack symbol Z_0 and the current position Z_m of the head of the pushdown at the moment $4n - k + 1$.

By $\alpha[k_1, k_2]$ we denote the set consisting of the integers $\{\alpha(k_1), \alpha(k_1 + 1), \cdots, \alpha(k_2)\}$. By $\beta[k_1, k_2]$ we denote the set consisting of the integers $\{\beta(k_1), \beta(k_1 + 1), \cdots, \beta(k_2)\}$. By $\gamma[k_1, k_2]$ we denote the set consisting of the integers $\{\gamma(k_1), \gamma(k_1 + 1), \cdots, \gamma(k_2)\}$.

By $\psi[k_1, k_2]$ we denote the maximum element in the set $\alpha[k_1, k_2]$. By $\chi[k_1, k_2]$ we denote the minimum element in the set $\beta[k_1, k_2]$.

Lemma 4. *For arbitrary positive integers (k_1, k_2), if $i \in \beta[k_1, k_2], j \in \beta[k_1, k_2]$ and $i < j$, then every integer n with $i \leq n \leq j$ is in $\beta[k_1, k_2]$.*

We fix a Kolmogorov numbering ϑ of all one-argument partial recursive functions. By v_1 we denote the word $w2w^{rev}$ where w is the word such that: 1) $w2w^{rev}, |w| = n$, 2) w has maximal ϑ-complexity. **!!! what is M_n?** Similarly, by v_2 we denote the word $w2w^{rev}$ where w is the word such that: 1) $w2w^{rev}, |w| = n^2$, 2) w has maximal ϑ-complexity. Finally, by (v_3, v_4) we denote the pair $(v_3, v_4) = (w2w^{rev}, \hat{w}2\hat{w}^{rev})$ such that: 1) $\mid w \mid= n$ and $\mid \hat{w} \mid> n$, 2) the pair (v_3, v_4) has maximal ϑ-complexity.

The maximal ϑ-complexity of (v_3, v_4) implies existence of a constant c, *independent of* n such that both the following two conditions are true: 1) v_3 has ϑ-complexity no less than $n - c$ and 2) v_4 has ϑ-complexity no less than $\mid \hat{w} \mid -c$.

Now consider the language

$$M = \{w2w^{rev} \mid w \in \mathbb{B}^*\}$$

which is clearly $(1, 1)$-recognizable by a 2-DFPA. Assume by contradiction that M is also $(2, 2)$-recognizable by some 2-DFPA.

We consider the computation by the assumed 2-DFPA on a pair (v_3, v_4) such that the with maximal ϑ-complexity such that $v_3 = w2w^{rev}$ with $\mid w \mid= n$ and $v_4 = \hat{w}2\hat{w}^{rev}$ with $\mid \hat{w} \mid> n$.

Consider the moment when n symbols have been read from the inputs. At this moment the length of the pushdown store is no less than $d \cdot n$ where $d < 1$ is an appropriate constant.

Lemma 5. *Consider the pair (v_3, v_4) with maximal ϑ-complexity such that $v_3 = w2w^{rev}$ with $\mid w \mid= n$ and $v_4 = \hat{w}2\hat{w}^{rev}$ with $\mid \hat{w} \mid> n$. Then*

$$(\exists d \leq 1)(\exists c)(\forall k \leq n)[\alpha(k) > d \cdot k - c].$$

Proof. Immediately from the maximality ϑ-complexity of the pair (v_3, v_4). □

We consider separately two possibilities:

1. $(\exists e < 1)(\exists(k_1, k_2))\{\mid k_2 - k_1 \mid\geq e \cdot n$ and $\chi[k_1, k_2] \geq \psi[k_1, k_2]\}$,
2. $(\forall e < 1)(\forall(k_1, k_2))\{\mid k_2 - k_1 \mid< e \cdot n$ or $\chi[k_1, k_2] < \psi[k_1, k_2]\}$,.

In the first case the DFPA has recorded the information on the symbols of the first input word read during the moments $(k_1, k_1 + 1, \cdots, k_2)$ in the squares Z_i of the pushdown store, where $i \in \alpha[k_1, k_2]$ and in the states of the finite memory, but the (supposedly) palindromic image of this part of the first input word is read at a different position of the pushdown store. The information about the distinct parts of the word have to be "transported" but the automaton lacks means to do this. Because of the maximality ϑ-complexity of the pair (v_3, v_4) this information cannot be compressed more than by an additive constant.

On the other hand, the existence of the assumed 2-DFPA $(2, 2)$-recognizing the language would allow to compress v_3 for $const \cdot n$ bits. Indeed, we construct an algorithm $\mathcal{B}$ using as an input

$$\mid v_3 \mid + \mid v_4 \mid -(k_2 - k_1) + \lceil \log k_1 \rceil$$

bits of information describing all the symbols in the words (v_3, v_4) with the exception of the symbols $2n + 1 - k_2, \cdots, 2n + 1 - k_1$. The algorithm $\mathcal{B}$ tests all possible replacements of the missing $(k_2 - k_1)$ symbols. In one and only one case the assumed 2-DFPA $(2,2)$-recognizing the language M accepts the first input. This allows to reconstruct the pair (v_3, v_4) using only

$$\mid v_3 \mid + \mid v_4 \mid -(k_2 - k_1) + \lceil \log k_1 \rceil$$

bits of the input. Since ϑ is a Kolmogorov numbering, ϑ uses no more that

$$\mid v_3 \mid + \mid v_4 \mid -(k_2 - k_1) + \lceil \log k_1 \rceil + \text{const}$$

bits of the input. This is a contradiction with the maximality ϑ-complexity of the pair (v_3, v_4).

In the second case we consider processing the pair (v_3, v_4) where $v_3 \in M$, $v_4 \in M, \mid v_3 \mid= 2n+1$ and $\mid v_4 \mid= 4n+1$.

Let the set Γ of the stack symbols contain no more than 2^s distinct symbols. Then a word of u symbols in the pushdown contains no more than sn bits of information. The condition

$$(\forall e < 1)(\forall (k_1, k_2))\{\mid k_2 - k_1 \mid < e \cdot n \text{ or } \chi[k_1, k_2] < \psi[k_1, k_2]\}$$

implies (for $e = \frac{1}{s}, k_2 = n$ and $k_1 = n - \frac{n}{2s}$) the inequality

$$\chi[n - \frac{n}{2s}, n] < \psi\left[n - \frac{n}{2s}, n\right],$$

hence

$$\left(\exists k \in \left[n - \frac{n}{2s}, n\right]\right)(\beta(k) < \frac{n}{s}\right).$$

Consequently, at the moment k each of the two inputs have received a word of the length $k \in [n - \frac{n}{2s}, n]$, this pair of words has maximal ϑ-complexity and this information is stored in the frequency 2-DFPA on a pushdown of a length not exceeding $\frac{n}{2s}$, a contradiction.

We have proved:

Theorem 5. *The language $M = \{w2w^{rev} \mid w \in \mathbb{B}^*\}$ is $(1,1)$-recognizable, but not $(2,2)$-recognizable by a 2-DFPA.*

Theorem 5 can be strengthened as follows:

Theorem 6. *The language $M = \{w2w^{rev} \mid w \in \mathbb{B}^*\}$ is $(1,1)$-recognizable but for all n it is not $(2,n)$-recognizable by a 2-DFPA.*

Idea of the proof. Following the example of Definition 10 we introduce the notion "an n-tuple of words has maximal ϑ-complexity". The language M has the following important property: the first half of any word in the language M determines the word uniquely.

In Theorem 5 proof of Theorem 5, the first input word is shorter than the other input words. Denote its length by s. The lengths of all input words are in $[s, h \cdot s]$ with an appropriate constant h. Like in the proof of Theorem 5, we distinguish two possibilities. Either during the moments $[s, 2s]$ the head on the pushdown store reads “nearly all the symbols recorded during the moments $[1, s]$” or it is not so. In the first case the DFPA can be forced to err on all the input words with the exception of the first input word. In the second case the DFPA can be forced to err on the first input word but a similar argument can be used for the second, the third,..., the $(n-1)$-th input word.

Open problem. *For which $n \in \mathbb{N}_+$ there exist $(2, n)$-recognizable and regular languages?*

References

[1] Ablaev, F.M., Freivalds, R.: Why sometimes probabilistic algorithms can be more effective. In: Gruska, J., Rovan, B., Wiedermann, J. (eds.) Mathematical Foundations of Computer Science 1986, Proceedings of the 12th Symposium, Bratislava, Czechoslovakia, August 25-29, 1986. Lecture Notes in Computer Science, vol. 233, pp. 1–14. Springer, Berlin (1986)

[2] Ambainis, A., Apsitis, K., Calude, C., Freivalds, R., Karpinski, M., Larfeldt, T., Sala, I., Smotrovs, J.: Effects of Kolmogorov complexity present in inductive inference as well. In: Li, M., Maruoka, A. (eds.) Algorithmic Learning Theory, 8th International Conference, ALT ’97, Sendai, Japan, October 6-8, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1316, pp. 244–259. Springer, Berlin (1997)

[3] A.N.Kolmogorov: Three approaches to the quantitative definition of information. Problems in Information Transmission 1(1), 1–7 (1965)

[4] Austinat, H., Diekert, V., Hertrampf, U., Petersen, H.: Regular frequency computations. Theoretical Computer Science 330(1), 15–21 (2005)

[5] Balodis, K., Kucevalovs, I., Freivalds, R.: Frequency prediction of functions. In: Kotasek, Z., Bouda, J., Cerna, I., Sekanina, L., Vojnar, T., Antos, D. (eds.) Mathematical and Engineering Methods in Computer Science. Lecture Notes in Computer Science, vol. 7119, pp. 76–83. Springer, Berlin (2012)

[6] Bārzdiņš, J.: On a class of turing machines (minsky machines). Algebra i Logika 1(6), 42–51 (1962)

[7] Beigel, R., Gasarch, W., Kinber, E.: Frequency computation and bounded queries. Theoretical Computer Science 163(1-2), 177–192 (1996)

[8] Calude, C.: Borel normality and algorithmic randomness. In: Rozenberg, G., Salomaa, A. (eds.) Developments in Language Theory, At the Crossroads of Mathematics, Computer Science and Biology, Turku, Finland, 12-15 July 1993. pp. 113–129. Lecture Notes in Computer Science, World Scientific, Singapore (1993)

[9] Calude, C.S., Zimand, M.: Algorithmically independent sequences. Information and Computation 208(1), 292–308 (2010)

[10] Case, J., Kaufmann, S., Kinber, E.B., Kummer, M.: Learning recursive functions from approximations. J. Comput. Syst. Sci. 55(1), 183–196 (1997)

[11] Degtev, A.N.: On (m, n)-computable sets. In: Moldavanskij, D.I. (ed.) Algebraic Systems, pp. 88–99. Ivanovo Gos. Universitet (1981), (in Russian)

[12] Freivalds, R.: Complexity of probabilistic versus deterministic automata. In: Bārzdiņš, J., Bøjrner, D. (eds.) Baltic Computer Science. Lecture Notes in Computer Science, vol. 502, pp. 565–613. Springer, Berlin (1991)

[13] Freivalds, R.: Space and reversal complexity of probabilistic one-way turing machines. In: Karpinski, M., van Leeuwen, J. (eds.) North-Holland Mathematics Studies, vol. 102, pp. 39–50. North-Holland (1997)

[14] Freivalds, R., Zeugmann, T., Pogosyan, G.R.: On the size complexity of deterministic frequency automata. In: Adrian Horia Dediu, Carlos Martin-Vide, B.T. (ed.) Language and Automata Theory and Applications. Lecture Notes in Computer Science, vol. 7810, pp. 287–298. Springer, Berlin (2013)

[15] Harizanov, V., Kummer, M., Owings, J.: Frequency computations and the cardinality theorem. The Journal of Symbolic Logic 57(2) (1992)

[16] Hinrichs, M., Wechsung, G.: Time bounded frequency computations. Information and Computation 139(2), 234–257 (1997)

[17] Kőnig, D.: Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe. Akademie-Verlag, Leipzig (1936)

[18] Khoussainov, B., Semukhin, P., Stephan, F.: Applications of kolmogorov complexity to computable model theory. The Journal of Symbolic Logic 72(3), 1041–1054 (2007)

[19] Kinber, E.B.: Frequency computations in finite automata. Cybernetics and Systems Analysis 12(2), 179–187 (1976)

[20] Kinber, E.B.: Frequency calculations of general recursive predicates and frequency enumerations of sets. Soviet Mathematics 13, 873–876 (1972)

[21] Kinber, E.B., Smith, C.H., Velauthapillai, M., Wiehagen, R.: On learning multiple concepts in parallel. J. Comput. Syst. Sci. 50(1), 41–52 (1995)

[22] Kummer, M.: A proof of Beigel's cardinality conjecture. The Journal of Symbolic Logic 57(2), 677–681 (1992)

[23] Kummer, M., Stephan, F.: The power of frequency computation (extended abstract). In: Reichel, H. (ed.) Fundamentals of Computation Theory, 10th International Symposium, FCT '95, Dresden, Germany, August 22-25, 1995. Lecture Notes in Computer Science, vol. 965, pp. 323–332. Springer, Berlin (1995)

[24] Kummer, M., Stephan, F.: Recursion theoretic properties of frequency computation and bounded queries. Information and Computation 120(1), 59–77 (1995)

[25] McNaughton, R.: The theory of automata, a survey. Advances in Computers 2, 379–421 (1961)

[26] Rose, G.F.: An extended notion of computability. In: International Congress for Logic, Methodology and Philosophy of Science, Stanford University, Stanford, California, August 24 - September 2, 1960, Abstracts of contributed papers (1960)

[27] Trakhtenbrot, B.A.: On the frequency computation of functions. Algebra i Logika 2(1), 25–32 (1964), (in Russian)