On the task of extracting the root from the language

B. F. Melnikov, S. Yu. Korabelshchikova, V. N. Dolgov

***Abstract*—In this paper, we consider a special case of constructing the inverse morphism for the given finite language: we try to extract the root from this language. For such extracting, we consider some special simple cases.**

We shall firstly look at some special class of languages, for which the problem is solved easily. Namely, we describe this class, consider some examples, and give some obtained results of computer experiments. Then we define antiderivative roots and give some propositions that show, how to use antiderivative roots for obtaining any root of the given degree.

Then we give an interesting example, which refutes a possible simple algorithm for this task. Therefore, if there exists a polynomial-time algorithm that solves this problem, then it should be formulated more complicated.

We hope that the further development of the theory described in this paper will provide an opportunity to describe a polynomial-time algorithms for solving the general case of this problem.

***Keywords*—formal languages, exponential algorithm, polynomial-time algorithm, root extraction.**

I. Introduction

We use terminology, notation, and some results from [1]. However, this paper is *not* a continuation of that one: here, we only consider one of the problems set there. It would be more correct to say, that this paper is a continuation of the topics of [2], [3], [4], [5].

In this paper, we look at an important case for the hypothesis considered in [1] (we called it "the second hypothesis"): we consider *a special case of constructing the inverse morphism* for the given finite language, i.e., we try to extract the root from the given finite language. Let us note, that our task is much more complicated than, for example, the simple algorithms of extracting the root *from a given word*; the last one can be formulated, e.g., by [6, Ch. 1, Ex. 4, 5].

Thus, we extract the root from the given finite language. We can formulate our problem in the following way. For the given language A, we build such a language B over the given alphabet Σ, for which

$$A = B^n. \quad (1)$$

(All notation associated with the theory of formal languages is standard. We can, for example, assume, that we are applying notation of [6] using the "new" symbol for the empty word, i.e. ε.)

We already used the term "a root of the language"; by (1), B can be defied as "a root of the language of n-th degree". However, this term requires further definitions, we shall consider only some of them in this paper. The main note is that the root, generally speaking, is not the only one (see details below). Therefore, the simple notation $\sqrt[n]{A}$ is used:

- sometimes for the set B such that $B^n = A$;
- and sometimes for *the set of such sets*.

For this reason, we shall *not use* this notation.

For such extracting, we consider some special simple cases, and then give an interesting example, which refutes a possible simple algorithm for this task. (This *incorrect* algorithm can be considered as an analogous to the mentioned algorithm of extracting the root from the given word.) Therefore, if there exists a *polynomial-time* algorithm that solves this problem, then it should be formulated more complicated. We hope that the further development of the theory described in this paper will provide an opportunity to describe a polynomial-time algorithms for solving this problem and the other problems set in [1].

This paper has the following structure. In Section II, we give some reasons that explain the need to consider our task. In Section III, we consider the formal problem setting and give an example.

In Sections IV and V, we shall look at some special class of languages, for which the problem is solved easily. Namely, in Section IV, we describe this class, consider some examples, and give some obtained results of computer experiments. And in Section V, we define *antiderivative roots* and give some propositions that show, how to use antiderivative roots for obtaining any root of the given degree.

In Section VI, we consider the general statement of the problem again. We formulate an method for extracting roots; however, as we said before, this method leads to an incorrect algorithm. But we believe that it is to be formulated, because it reflects possible directions for further work. Some of these directions are shortly described in Conclusion (Section VII).

II. Motivation: why do we consider this task?

Some reasons that explain the need to consider this task were already given in our previous papers, for example, in [1], [4], [5], etc. Let us cite some of them, as well as the motivation not met in these papers.

- In some subclasses of the context-free language class *the equivalence problem is decidable*, unlike the whole class; see [7], [8], [9].
- An example of such a class is the "notorious" class of deterministic context-free languages, the equivalence issue for which was formulated back in the late of 1960s (see, for example, [10, Sect. 4.2]), and then was subsequently resolved. It is not known to the authors of

Received December 31, 2018.

Boris F. Melnikov, Russian State Social University (email: bf-melnikov@yandex.ru).

Svetlana Yu. Korabelshchikova, Northern (Arctic) Federal University named after M. V. Lomonosov (email: s.korabelsschikova@narfu.ru).

Vasily N. Dolgov, independent researcher (email: terenga74@mail.ru).

this paper, that anyone found errors in the proof given in a series of parers of L. Staneviciené (see [11], [12] and some papers cited there). Besides, it is possible, apparently, to bring to the end a not completed proof of V. Meytus, see [13]: that paper does not contain obvious errors. But, *as usual in such situations*, the author of the proof for the problem of equality of deterministic CF-languages (i.e., equivalence of deterministic MP-automata) "is generally recognized" *not* a Russian scientist, but G. Sénizergues, see [14].[1]

Thus, we are referring to subclasses that *do not coincide* with the class of deterministic CF-languages. However, some real programming languages can be described using these subclasses, see some examples in [9]. To describe these subclasses, we often consider pairs of languages satisfying the conditions $A \equiv B$. (As we said before, the corresponding definition was given in [1], [2] etc.)

- Based on the conditions of the relationship $A \equiv B$, we can formulate some necessary and sufficient conditions for the commutation in the global monoid of the free monoid [2] and some of its submonoids, see [3], [5], [15]. In other words, we can formulate some criteria for the fulfillment of equality $AB = BA$, where $A, B \subseteq \Sigma^*$.
- The possible association between the conditions of the relationship $A \equiv B$ and some other algebraic problems can be derived from the results of the papers [16], [17], [18], where ω- and 2ω-languages are considered. [3]
  In our problems, such languages are usually generated by infinite sequences of reflections of a point from the sides of a certain polygon (in other words, they are generated by special billiards). These results are associated with some algorithmic problems for monomial algebras (i.e. associative algebras defined by so-called obstruction languages). This relationship follows from the results of the classic works [19], [20], etc.
  Note that in [1], we could consider the same ω-automata instead of usual nondeterministic finite automata, and formulate equivalence conditions for them.
- The tasks discussed in this paper arise in the graphical description of the CF-languages class and some its subclasses, see [11], [21], [22], [23]. In particular, we obtain them when solving the equivalence problems in these subclasses. [4]
- And oddly enough, so-called "infinite" case (i.e., when we allow infinite languages A or B in the condition $A \equiv B$) is less complicated and less interesting than the "finite" one; some results for the infinite case were given in [24]. However, in some tasks, the need to consider infinite languages A and B arises: for example, when we consider the problem of description of conditions of equivalence of morphic images of so-called bracketed languages. (See papers [8], [9] cited before; we should note once again, that in these works we have considered some other variants of using morphisms of infinite languages for description of some subclasses of the CF-languages class.)

However, as we noted in [1], the most interesting question is to establish a connection between the issues we are considering and the possible equality P=NP. In the mentioned paper, the possible plan of research of this problem is given, namely, the plan of proof of inequality of these classes.

III. The formal problem setting and an example

Thus, we shall extract here the root of the language; i.e., for the given language A, we build such a language B over the given alphabet Σ, for which equation (1) holds. About (1), we note the following:

- either $n \geqslant 2$ is set a priori; then, in case of absence of the required language B, we return the negative answer, also in the polynomial time;
- or n is *not* given; then we have to construct the language B satisfied (1) for maximum possible $n \geqslant 1$.

It is obviously, that two these tasks almost coincide (and do not change the fact of the possibility or impossibility of constructing a polynomial-time algorithm): if there is a solution to the first one, then by considering all the values n from

$$\max_{u \in A} |u|$$

"downto" 1, we get the solution for the second of these tasks.

We also note the following fact. Of course, the task of extracting the root *from a given word* is quite simple; but still now, some of related questions are considered in the famous monograph [6] as not quite trivial exercises. However, it is unlikely that summarizing the relevant results for the words to the case *of the languages* [5] can give interesting results: it is, in fact, the reformulating the tasks for words in the terms of languages.

In this regard, we also note the following fact from [1]. The algorithm of constructing all the variants of representation of the given word u in the form of concatenation of words of some given finite language A *is not polynomial-time*. In this regard, let us consider a trivial example

$$u = a^n, \qquad A = \left\{ a, a^2, \ldots, a^{n-1} \right\}.$$

On this issue, we have the following facts:

- the size of the problem (depending on the n) is polynomial;
- the number of variants for a representation required in our problem is exponential (also depending on n);
- and, therefore, the number of variants for the consideration (depending on the size of the task, we use the brute force method here) is also exponential.

However, this fact does *not* prove the non-existing polynomial-time algorithm for the whole problem we are considering, i.e. for the construction of inverse morphism required in our problem.

[1] Apparently, an even more obvious example is the following. Yu. Medvedev published a paper in 1956, where, in fact, nondeterministic finite automata were defined and considered. However, the authors of NFAs "are generally recognized" M. Rabin and D. Scott, who described them later, i.e. in 1959, but received the Turing Prize for describing such automata.

[2] Sometimes it is called "supermonoid", but that is not quite true.

[3] For example, we can define 2ω-word α over the given alphabet Σ as any mapping of the type $\alpha : \mathbb{Z} \to \Sigma$.

[4] Moreover, the development of this graphic approach gives an opportunity to describe (also graphically) languages, having types 1 and 0 in the hierarchy of Chomsky.

[5] From the algebraic point of view, it is *a transition from monoids to supermonoids*. (The last term is considered *in some* sources not quite successful, but we will not pay attention to such algebraic subtleties.)

Tab. 1. The count of roots

n / k	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	1	2	3	5	9	15	28	50	95	174	337	637	1231	2373	4618	8974	17567
3	1	2	4	7	13	25	49	95	185	365	721	1430	2844	5655	11259	22462	44829
4	1	2	4	8	15	29	57	113	225	447	889	1773	3537	7062	14109	28195	56361
5	1	2	4	8	16	31	61	121	241	481	961	1919	3833	7661	15317	30625	61233
6	1	2	4	8	16	32	63	125	249	497	993	1985	3969	7935	15865	31725	63445
7	1	2	4	8	16	32	64	127	253	505	1009	2017	4033	8065	16129	32255	64505
8	1	2	4	8	16	32	64	128	255	509	1017	2033	4065	8129	16257	32513	65025
9	1	2	4	8	16	32	64	128	256	511	1021	2041	4081	8161	16321	32641	65281
10	1	2	4	8	16	32	64	128	256	512	1023	2045	4089	8177	16353	32705	65409

IV. About roots of special-type languages: some simple examples and computer experiments

In this section and in the next one, we shall look at some special class of languages, for which the problem is solved easily. Namely, we investigate the question of extracting the root of a given degree from the languages of the type

$$\bigcup_{t_1 \leqslant i \leqslant t_2} \Sigma^i, \text{ where } t_1, t_2 \in \mathbb{N},\ t_1 \leqslant t_2. \tag{2}$$

Obviously, the root extraction operation is, generally speaking, ambiguous: let us consider an example which is an evolution of one considered in Section III.

Example 1. Let

$$A = \left\{ a^8, a^9, \ldots, a^{23}, a^{24} \right\}.$$

Square roots can be considered, among some others, languages

$$B_1 = \left\{ a^4, a^5, \ldots, a^{11}, a^{12} \right\},$$
$$B_2 = B_1 \setminus \left\{ a^6 \right\}, \quad \text{and} \quad B_3 = B_1 \setminus \left\{ a^{10} \right\};$$

we have

$$B_1^{\,2} = B_2^{\,2} = B_3^{\,2} = A,$$

besides

$$B_1 = \left\{ a^2, a^3, a^4, a^5, a^6 \right\}^2 = \left\{ a^1, a^2, a^3 \right\}^4,$$

and, at the same time, we have no language C, for which

$$C^2 = B_2 \quad \text{or} \quad C^2 = B_3.$$

Let us consider another interesting example.

Example 2. Let

$$A = \left\{ a^0, a^1, a^2, a^6, a^8, a^9 \right\},$$
$$B = \left\{ a^0, a^1, a^3, a^6, a^8, a^9 \right\}.$$

Then

$$A^2 = B^2 = \left\{ a^0, a^1, \ldots, a^{18} \right\} \setminus \left\{ a^5, a^{13} \right\},$$

however

$$(A \cup B)^2 = \left\{ a^0, a^1, \ldots, a^{18} \right\} \neq A^2.$$

Let us also note another obvious fact: in order for the root of n-th degree from the language of the type (2) to be extracted, it is necessary and enough to both t_1 and t_2 are divided into n. Let us formulate a simple generalization of the last fact in the form of the following statement.

Proposition 1: Language

$$\bigcup_{i \in M} \Sigma^i$$

is the root of the n-th degree for language

$$A = \bigcup_{n \cdot n_1 \leqslant i \leqslant n \cdot n_2} \Sigma^i \tag{3}$$

if and only if

$$M \subseteq \{n_1, n_1 + 1, \ldots, n_2\},$$

for which the following condition holds:

$$(\forall i \in \mathbb{N})\, (n \cdot n_1 \leqslant i \leqslant n \cdot n_2 \Rightarrow i = a_1 + a_2 + \cdots + a_n).$$

Here, a_1, a_2, …, a_n are some elements of the set M, not necessarily different ones.

Besides, we have the following bijection between the roots of the n-th degree for language A defined before and the roots of the n-th degree for language

$$B = \bigcup_{n \leqslant i \leqslant n \cdot n_2 - n \cdot n_1 + n} \Sigma^i \ :$$

the set of indices M of the root of language B corresponds the set of indices

$$L = M \cup \{n_1 - 1\}$$

of the root of language A. □

Note that the number of roots of the n-th degree depends on the n and the difference $n_2 - n_1$; the last value is the difference of boundary values of the set of root indices. The notation A (which, certainly, depends on n, n_1, and n_2, that we shall not explicitly mark) will be used in the remainder of the paper.

For further, we denote also $k = n_2 - n_1 + 1$; this value is the cardinality of the set $\{n_1, n_1 + 1, \ldots, n_2\}$. We implemented an algorithm of finding all the possible roots for the given n, t_1, and t_2. The complexity of the calculation of the obtained implementation is equal to

$$O\big(2^{(k-4)} \cdot n^2 \cdot k^2\big);$$

it allows to find all the roots for $k \leqslant 30$. Table 1 (see before) shows the count of roots for some n and k that our computer program obtained. (For $k \in \{1, 2, 3\}$, the values are equal to 1.)

V. Definition and basic properties of antiderivative roots from languages

For the future, let S be *the set of set of indices* for the roots of the n-th degree for the language A defined by (3).

Definition 1: The root of the type

$$\bigcup_{i \in M} \Sigma^i$$

will be called an *antiderivative* one, if M is minimum on inclusion of the set S.

The cardinality of the set of indices will be called the *weight* of the root. □

Example 3. For the given language

$$\bigcup_{2\leqslant i\leqslant 14} \Sigma^i,$$

we have the following 5 different square roots; they have sets of indices

$$\{1,2,3,4,5,6,7\},\ \{1,2,3,4,6,7\},\ \{1,2,4,5,6,7\},$$
$$\{1,2,4,6,7\},\quad \text{and}\quad \{1,2,3,5,6,7\}.$$

The two last sets are minimum ones, they corresponds to antiderivative roots with the weights 5 and 6 respectively.

Let us formulate the obvious properties of the antiderivative in the form of following statements.

Proposition 2: Let $M \in S$, besides let M be a set of indices of minimum cardinality. Then the root corresponding to M is an antiderivative one. □

Note that, generally speaking, the assertion reversed to the last one is incorrect: the antiderivative root is not required to have a minimum weight, see Example 3.

Proposition 3: Let

$$\bigcup_{i\in M}\Sigma^i$$

be an antiderivative root of n-th degree for the language A (3) with the set of indices M. Let

$$(\alpha_1, \alpha_2, \ldots, \alpha_k)$$

be the binary characteristic vector (see [25] etc.) for subset M of the set

$$\{n_1, n_1+1, \ldots, n_2\}.$$

Then the symmetrical vector

$$(\alpha_k, \alpha_{k-1}, \ldots, \alpha_1)$$

also sets the set of indices of (another) antiderivative root. □

Proposition 4: If the antiderivative root of the given language is the only one, then the characteristic vector of its set of indices is symmetrical. □

Proposition 5: Let

$$\bigcup_{i\in M}\Sigma^i$$

be an antiderivative root of n-th degree for the language A (3) with the set of indices M. Then for each set M', such that

$$M \subseteq M' \subseteq \{n_1, n_1+1, \ldots, n_2\},$$

the language

$$\bigcup_{i\in M'}\Sigma^i$$

is the root of n-th degree for the same language. □

Proposition 6: If the weight of the antiderivative root is equal to $w \leqslant k$ and it is the only, then the common number of the roots for the considered language is equal to 2^{k-w}.

Proof. This fact follows from the Proposition 4. □

Thus, as follows from the properties of antiderivative roots, to get all the roots of n-th degree for the language

$$\bigcup_{n\cdot n_1\leqslant i\leqslant n\cdot n_2} \Sigma^i,$$

there is enough to know all the original roots. This coincides with obtaining their set of indices.

Example 4. Let us construct all the root of 3-rd degree of the language

$$\bigcup_{27\leqslant i\leqslant 42} \Sigma^i.$$

The set of indices should be the subset of the set

$$\{9, 10, 11, 12, 13, 14\},$$

i.e. $k = 6$. For $n = 3$ and $k = 6$, we have the only antiderivative root, its characteristic vector is (110011). Then we obtain 4 roots, having the following sets of indices:

$$\{9,10,13,14\},\ \{9,10,11,13,14\},$$
$$\{9,10,12,13,14\},\quad \text{and}\quad \{9,10,11,12,13,14\}.$$

Thus, the task of extracting the root from this type of languages can be reduced to the task of representing some natural numbers from the specified interval $[t_1, t_2]$ as the sum of n natural terms from the interval $[t_1/n, t_2/n]$. Therefore, such a task can be considered as a special case of the knapsack problem. Some possible algorithms for solving this problem *for situations we are considering in this paper*, were considered by one of the authors in [26]. The building set of antiderivative of n-th degree allows to find all the roots from the languages of the type (2). On the other hand, if the set of all roots of a given language is large enough, then the building its antiderivative roots can be considered as the best way for description of this set.

VI. Why do not simple algorithms work?

Let us return to the general statement of the problem.

When considering our task (as well as some tasks related to it), we can suggest the following simple "algorithm" for its solving; let us note in advance, that this algorithm is *incorrect*, this can be seen from the title of this section. However, we believe that it is to be formulated, because it reflects possible directions for further work, in particular, to obtaining *the correct polynomial-time algorithm* for solving the problem we are looking at.

Thus, we will consider the problem of extracting the root of the n-th degree, when the language from which the root is extracted (let it be $A \subseteq \Sigma^*$) and the value $n \geqslant 1$ are given.

Definition 2: The word $u \in \Sigma^*$ is a *potential root* if $u^n \in A$.

The set of all potential roots will be denoted $\overline{\sqrt[n]{A}}$.[6] (We shall never omit n.) □

Example 5. Let

$$A = \{\, abab, abaab, aabab, aabaab \,\}.$$

Then

$$\overline{\sqrt[2]{A}} = \{\, ab, aab \,\}.$$

Let us remark, that

$$\left(\overline{\sqrt[2]{A}}\right)^2 = A.$$

Example 6. Let

$$A = \{\, aa, abab, baba, babbab \,\}.$$

[6] As we said in Introduction, we do not use the simple notation $\sqrt[n]{A}$.

Then let us denote

$$\widetilde{B} = \overline{\sqrt[2]{A}} = \{\, a, ab, ba, bab \,\} \, ;$$

let us remark, that

$$\big(\widetilde{B}\big)^2 \neq A,$$

because, for example,

$$\widetilde{B} \ni aab, \quad \text{and} \quad aab \notin A.$$

However, there exists B, such that $B^2 = A$, that is

$$B = \{\, a, bab \,\} \, ;$$

let us remark, that $B \subseteq \widetilde{B}$.

The following simple statement is given without proof.

Proposition 7: Let $B^n = A$. Then

$$B \subseteq \overline{\sqrt[n]{A}}. \qquad \square$$

Based on the last proposition, we obtain a trivial algorithm for building any B, such that $B^n = A$: we should consider all the subsets of the set $\overline{\sqrt[n]{A}}$. Evidently, this algorithm is *not a polynomial-time* one.

Besides, we could formulate a simple "algorithm" based on the following hypothesis:

1) we consider a word of $\overline{\sqrt[n]{A}}$ having the maximum possible length; if there are 2 or more such words, then we choose an arbitrary one; let this word be u;
2) then we consider a word $v = u^{n-1}$;
3) if $vw \in A$, then we include w in the formed language (let it be C);
4) C has no more words;
5) then $C^n = A$.

However, *this hypothesis is false*, and we can show this fact in the following example.

Example 7. Let

$$B = \{\, ab, abb, bba \,\} \, ,$$

and let $A = B^2$ be *the given language*. Then A contains the word $abbba$. Besides:

1) $B = \overline{\sqrt[2]{A}}$;
2) abb is a word of $B^{2-1} = B$, such that it has the maximum possible length, i.e. 3;
3) $abbba = abb \cdot ba$,

but $ba \notin B$.

We believe, that this example shows the need to build more complex algorithms.

VII. Conclusion. Some directions for further work

Thus, we need to build more complex algorithms. However, of course, all algorithms for this problem should be polynomial-time ones: the trivial exponential algorithm was given in previous section.

(In describing the remaining problems for further solution, we again use the terminology of [1] and other cited works. Namely, we use the terms $\equiv$ and mp^+ defined there.)

We hope that in the future, we shall be able to describe *polynomial-time algorithms* for solving both the problem formulated in this paper and also one of the two following more complex problems:

- for *two* given languages A and B, such that $A \neq B$, but $A \equiv B$, we have to build language D, such that $A, B \in \mathrm{mp}^+(D)$;
- for the given finite language A, we have to build language D, such that $A \in \mathrm{mp}^+(D)$, besides, language D is minimal according to a metrics; the possibilities for its choices for the language D are "natural", some of possible metrics are also discussed in [1].

It is important to note that according to [2], [3], [5], the solution of one of them automatically involves the solution of the other. Moreover, it is easy to show that the description of the polynomial-time algorithm for solving one of them will give a description of a comparable algorithm for solving the other.

Acknowledgment

We express our gratitude for some consultations to A. Verëvkin (Ulyanovsk State University, Russia).

References

[1] Melnikov B., Vylitok A., and Melnikova E. *Iterations of languages and finite automata*. International Journal of Open Information Technologies. 2017, vol. 5, no. 12, pp. 1–7. (in Russian, http://injoit.org/index.php/j1/article/view/496)

[2] Melnikov B. *The equality condition for infinite catenations of two sets of finite words*. International Journal of Foundations of Computer Science. 1993, vol. 4, no. 3, pp. 267–274.

[3] Melnikov B. *Description of special submonoids of the global supermonoid of the free monoid*. News of higher educational institutions. Mathematics. 2004, no. 3, pp. 46–56. (in Russian, https://elibrary.ru/item.asp?id=9083702)

[4] Korabelshchikova S., and Melnikov B. *Maximum prefix codes and subclasses of the context-free language class*. Bulletin of Northern (Arctic) Federal University. Series: Natural Sciences. 2015, no. 1, pp. 121–129. (in Russian, https://elibrary.ru/item.asp?id=23141792)

[5] Korabelshchikova S., and Melnikov B. *On the common root of elements of the global supermonoid*. Bulletin of Northern (Arctic) Federal University. Series: Natural Sciences. 2016, no. 3, pp. 91–96. (in Russian, https://elibrary.ru/item.asp?id=27126372)

[6] Salomaa A. *Jewels of formal language theory*. Computer Science Press, Inc. (Maryland, USA), 1981, 157 p.

[7] Melnikov B. *Some consequences of the equivalence condition of unambiguous bracketed grammars*. Bulletin of Moscow University, ser. 15 (Moscow University Computational Mathematics and Cybernetics). 1991, no. 3, pp. 51–53. (in Russian, https://elibrary.ru/title_about.asp?id=8373)

[8] Dubasova O., and Melnikov B. *About one extension of CF-languages class*. Programmirovanie (Programming and Computer Software). 1995, no. 6, pp. 46–58. (in Russian, https://elibrary.ru/title_about.asp?id=7966)

[9] Melnikov B., and Kashlakova E. *Some grammatical structures of programming languages as simple bracketed languages*. Informatica (Lithuania). 2000, vol. 11, no. 4, pp. 441–454. (https://www.mii.lt/informatica/htm/INFO222.htm)

[10] Ginsburg S. *The Mathematical Theory of Context-free Languages*. McGraw-Hill Ed. (N.Y., USA), 1966, 245 p.

[11] Stanevichene L. *About one tool of context-free languages research*. Cybernetics. 1989, no. 4, pp. 135–136. (in Russian, http://www.kibernetika.org/contents/89.doc)

[12] Gomozov A., and Stanevichene L. *A generalization of regular expressions*. Informatica (Lithuania). 1999, vol. 10, no. 1, pp. 27–44. (https://www.mii.lt/informatica/htm/INFO140.htm)

[13] Meytus V. *The decidability of the problem of equivalence for deterministic push-down automata*. Cybernetics and system analysis. 1992, no. 5, pp. 20–45. (in Russian, http://www.kibernetika.org/contents/92.doc)

[14] Sénizergues G. *L(A)=L(B)? Decidability results from complete formal systems*. Theoretical Computer Science. 2001, vol. 251, no. 1–2, pp. 1–166.

[15] Melnikov B. *Some equivalence problems for free monoids and for subclasses of the CF-grammars class*. Number theoretic and algebraic methods in computer science, Proceedings of the International Conference, World Scientific Publ. 1995, pp. 125–137. (https://doi.org/10.1142/9789814532532)

[16] Melnikov B. 2ω-finite automata and sets of obstructions of their languages // The Korean Journal of Computational and Applied Mathematics. 1999, vol. 6, no. 3, pp. 565–574. (https://doi.org/10.1007/BF03009949)
[17] Melnikov B. *On ω-languages of special billiards* Discrete Mathematics. 2002, vol. 14, no. 3, pp. 95–108. (http://www.mathnet.ru/dm)
[18] Melnikov B., and Melnikova E. *Some more on the billiard languages and corresponding forbidden languages*. Proceedings of International Conference on Infinity in Logic & Computation. Cape Town. 2007, pp. 35–36.
[19] Ufnarovskii V. *Combinatorial and asymptotic methods in algebra*. Results of science and technology, ser. Modern Problems of Mathematics, Algebra. 1990, vol. 6, pp. 5–177. (in Russian, http://mi.mathnet.ru/intf160)
[20] Belov A., Borisenko V., and Latyshev V. *Monomial algebras*. Journal of Mathematical Sciences, 1997, vol. 87, no. 3, pp. 3463–3575.
[21] Vylitok A. *About building the graph of a push-down automaton*. Bulletin of Moscow University, ser. 15 (Moscow University Computational Mathematics and Cybernetics). 1996, no. 3, pp. 68–73. (in Russian, https://elibrary.ru/title_about.asp?id=8373)
[22] Vylitok A., Zubova M., and Melnikov B. *An extension of the class of finite automata to specify the context-free languages*. Bulletin of Moscow University, ser. 15 (Moscow University Computational Mathematics and Cybernetics). 2013, no. 1, pp. 39–45. (in Russian, https://elibrary.ru/item.asp?id=18862999)
[23] Generalova T., Melnikov B., and Vylitok A. *On the extension of the finite automata class for context-free languages specification*. International Journal of Open Information Technologies. 2018, vol. 6, no. 9, pp. 1–8. (in Russian, http://injoit.org/index.php/j1/article/view/602)
[24] Brosalina A., and Melnikov B.. *Commutation in global supermonoid of free monoids*. Informatica (Lithuania). 2000, vol. 11, no. 41, pp. 353–370. (https://www.mii.lt/informatica/htm/INFO220.htm)
[25] Birkhoff G., and Mak Lane S. *A DServey of Modern Algebra*. Macmillan Publishing Co., Inc. (N.Y., USA), 1977, 500 p.
[26] Zyablitseva L., Korabelshchikova S., and Chesnokov A. *Linear codes correcting errors and their counting algorithms*. Heuristic Algorithms and Distributed Computing. 2014, vol. 1, no. 3, pp. 47–59. (in Russian, https://elibrary.ru/item.asp?id=22376177).

Boris Feliksovich MELNIKOV,
Professor of Russian State Social University,
http://www.rgsu.net/,
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300.

Svetlana Yuryevna KORABELSHCHIKOVA,
Associated Professor (Docent)
of Northern (Arctic) Federal University
named after M. V. Lomonosov, Russia,
https://narfu.ru/en/university/,
email: s.korabelsschikova@narfu.ru,
elibrary.ru: authorid=711810.

Vasily Nikolaevich DOLGOV,
Independent Researcher, Russia,
email: terenga74@mail.ru,
elibrary.ru: authorid=700000.