Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2023, 11, 1471-1481
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

On Some Problems of Extracting the Root from a Given Finite Language

Boris F. Melnikov[1], **Aleksandra A. Melnikova**[2]

[1]Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU-BIT University, Shenzhen, China
[2]Department of Mathematics, Dimitrovgrad Engineering Institute of Technology—Branch of the National Research Nuclear University "MEPhI", Dimitrovgrad, Russian Federation
Email: bormel@mail.ru, super-avahi@yandex.ru

How to cite this paper: Melnikov, B.F. and Melnikova, A.A. (2023) On Some Problems of Extracting the Root from a Given Finite Language. *Journal of Applied Mathematics and Physics*, **11**, 1471-1481.
https://doi.org/10.4236/jamp.2023.115096

Received: March 6, 2023
Accepted: May 28, 2023
Published: May 31, 2023

Abstract

Based on the standard definition of the product (concatenation), the natural non-negative degree of the language is introduced. Root extraction is the reverse operation to it, and it can be defined in several different ways. Despite the simplicity of the formulation of the problem of extracting the root, the authors could not find any description of it in the literature (as well as on the Internet), including even its formulation. Most of the material in this article is devoted to the simplest version of the formulation: the root of the 2nd degree for the 1-letter alphabet, but many of the provisions of the article are generalized to more complex cases. Apparently, for a possible future description of a polynomial algorithm for solving at least one of the described statements of root extraction problems, it is first necessary to really analyze in detail such a special case, that is: either describe the necessary polynomial algorithm, or, conversely, show that the problem belongs to the class of NP-complete problems. Thus, in this article, we do not propose a polynomial algorithm for the problems under consideration; however, the models described here should help in constructing appropriate heuristic algorithms for their solution. A detailed description of the possible further application of such heuristic algorithms is beyond the scope of this article.

Keywords

Formal Languages, Iterations of Languages, Root Extraction, Algorithms, The Boolean

1. Introduction

The definition of a product (otherwise called concatenation) of two languages is introduced in the usual way [1]. Based on the introduced definition of the product, also in the usual algebraic method, an integer non-negative degree of the language is introduced. Root extraction is an inverse operation, and it can be defined in different ways, see below for some details. In the paper, we shall consider the problem of extracting the root from a given finite language.

Despite the simplicity of the formulation of the problem, the authors could not find any description of it in the literature (as well as on the Internet), including even its formulation. At the same time, we believe that all the material of the article is simple; therefore, it is very puzzling that at least there are no formulations of such problems in the monographs known to the authors [1] [2] and others.

Most of the material in this article is devoted to the simplest version of the formulation, the root of the 2nd degree for the 1-letter alphabet, but many of the provisions of the article are generalized to more complex cases. And, apparently, for a possible future description of a polynomial algorithm for solving at least one of the described problem statements, it is first necessary to really analyze in detail such a special case (in our usual notation $\sqrt[2]{A}$, $|\Sigma| = 1$), that is:

- either describe the required polynomial algorithm,
- or, conversely, show that the problem belongs to the class of NP-complete problems.

Let us present the contents of the article by sections. In Section 2, the applied designations and variants of the statements of the tasks solved in the article are given. Section 3 discusses some simple models designed to describe algorithms for solving the problem also in the simple case: for a 1-letter alphabet and a square root; one of the corresponding mathematical models is formulated in the language of graph theory. In Section 4, we shall again assume that N potential roots are obtained based on the in-put data. For some ordering of these potential roots (the natural ordering in the case of a 1-letter alphabet is an increase in the length of the potential root), consider an N-dimensional cube. The material of Section 5 describes the part of the subject of the article that seems to be the only non-trivial one; it considers adding a new coordinate to an already existing solution. It is on the basis of the results of the theorem proved in the section, that we can assert that the search for (at least one) root can be performed as a search for the vertex of the hypercube (the Boolean) closest to the maximum vertex and at the same time not included in any of the taboo planes.

2. Applicable Designations and Options for Setting the Problem

Let us start the section with a description of the standard notation. For the objects under consideration, the "multiplication" (we shall continue to use it without quotes) is simply the result of the product of two languages; in particular, consider this example for 1-letter alphabet: if

$$\mathrm{C} = \{a^i \mid i \in \mathrm{I}\} \text{ and } \mathrm{D} = \{a^j \mid j \in \mathrm{J}\}$$

(where I and J are some finite subsets of non-negative integers), then

$$\mathrm{C} \cdot \mathrm{D} = \{a^k \mid (\exists\, i \in \mathrm{I}, j \in \mathrm{J})\ (k = i + j)\}.$$

On the basis of the product, the degree of the language is naturally defined; then we proceed to the description of the root extraction operation. Throughout the description of tasks and examples to them, we shall adhere to the following conventions.

1) Everywhere further, the problem of finding the language X, which is the root of the equation

$$X^M = A \tag{1}$$

is solved, where the language A and the number $M \geqslant 2$ are predetermined. In most of the article, we shall solve the equation for $M = 2$. In this case, the *potential root* is some word $u \in \Sigma^*$, such that $u^M \in A$. Unlike them (*i.e.* words), we shall call X by *the root* (not by the potential root), if condition (1) is met. It is obvious that it is possible to find all potential roots in polynomial time (relative to the size of the initial data of the problem).

2) N is the number of potential roots, we number them from 1 to N, here we use "usual Pascal indexing".

3) Only for the case 1-letter alphabet: if "the problem is formed" by raising to the power of M some language (let it be a language B; we assume that when solving, we get the language X), then the representation of this language (as a set, B or X) we perform an enumeration from 0 to some given n; here we use "usual C (or similar algorithmic languages) indexing". In this case, $M \cdot n + 1$ will be the dimension of the source language A (for example, in the case of the 2nd degree, we need to somehow determine the digits from the 0th to the 2nth).

In the last agreement, in all cases (*i.e.*, for both A and B/X), we shall use 4 variants of the designation of a set of words:

- "the most common": for example, $\{a, ab, ba, bab\}$, or

$$\{\varepsilon = a^0,\ a = a^1,\ aaa = a^3,\ aaaaaa = a^6\}. \tag{2}$$

The remaining three variants of the notation will be used both for subsets of the set of potential roots, and for several elements of the set

$$\{\varepsilon, a, aa, \ldots a^{K-1}, a^K\}$$

(in the case of a 1-letter alphabet); in the last case, it will usually be either $K = n$ or $K = 2n$.

Here are examples for the language (2):

- [0 1 3 6], the set of degrees used is written in square brackets in ascending order;
- 1001011 is a binary number in which the digits of the previous list are marked with 1; the digits of the binary number are numbered starting from 0 from right to left;
- 75, it is the number belonging to N_0, representing the decimal equivalent of the previous item.

As already noted, we shall use the same notation when $K = N$, and at the same time we consider subsets of the set of potential roots; however, in this case, we shall write the digits of the binary number from left to right and number them starting from 1. For example, let there be only 5 elements, and the considered subset is {2, 5}:

- [2 5], sometimes just 2 5, even sometimes 25, it will not cause a difference; the empty set is marked in the usual way, *i.e.* ∅;
- 01001; the empty set here is the sequence of zeros;
- 9; empty set 0.

Only subsets of the set of potential roots (recall that we assumed that their number is N) will usually be considered as vertices of an N-dimensional hypercube. Now we define some auxiliary concepts related to such a hypercube.

- The definition of the maximal vertex of the hypercube is naturally (1, 1, ..., 1, 1).
- For further, in the case of considering the problem of extracting the root of the 2nd degree, so-called pairs of potential roots will often be used; in particular, the so-called taboo pairs (more details on them below).

For example, if:

  - we accept the "third agreement" (*i.e.* we consider the case of the 1-letter alphabet only),
  - at the same time, we consider the set [0 1 2 3 6],
  - the pair is {2, 6} (of course, the order of the elements of the pair is arbitrary),

then the following designations of such pair are also possible:

- $\overbrace{2,6}$ (this is just their values); is the same as $\overbrace{6,2}$;
- $\overbrace{\#3,\#5}$ (those are their numbers, counting the numbers on the rank from 1).

- For some taboo pair, a taboo hyperplane is such an N−2-dimensional hyperplane of an N-dimensional hypercube for which both coordinates of the elements of the pair are equal to 1.
- Note that the maximum vertex of the hypercube is included in the intersection of any number of any taboo hyperplanes.
- For some hypercube's point $(b_1\ b_2 \ldots\ b_N)$, the value $(b_1\ b_2 \ldots\ b_N)_{+k}$ means the vertex obtained from the previous one by changing the kth coordinate for 1.

Let us clarify a few statements of the tasks to be solved. As it was noted in [3], considering the problem of extracting the root of a given degree for a given finite language, we are actually dealing not with one problem, but with a whole group of problems, since there are several options for the required answer: a) find any (root from the language); b) find all; c) find the minimum (by some metric); etc.

The exponential algorithm for any variant of the root extraction problem is obvious: we only need to consider all the subsets of the set of potential roots, and among these subsets choose the appropriate (the suitable). Therefore, the problem is *to describe possible polynomial algorithms* for these variants of the problem.

3. The Case of the 1-Letter Alphabet and the Square Root: Simple Models for Solving the Problem

Thus, we shall assume that based on the input data we have obtained N potential roots; we repeat that all potential roots can be found using a simple polynomial algorithm. Here we shall try to find a solution to the simplest version of the problem: we need to extract the square root (the root of the 2nd degree), despite the fact that the alphabet consists of 1 letter; but, as already noted, the problem is not simple even in this case. We do not know a polynomial algorithm for such a "simple" option, so we are trying to use various approaches to solve the problem; at the same time, here we shall consider the simplest approaches, they seem to be immediately clear from the description of the problem.

We shall work with a set of potential roots. And, since each root is determined by its length, without the loss of generality, we shall assume that this length is in the range from 0 to some n; moreover, in the examples discussed below, there is always a potential root of the length 0.

Arrange the selected values from 0 to n in the form of marks of rows and columns of the table $(n + 1) \times (n + 1)$, while in the cells of the table we write the sum of the numbers that are the marks of the corresponding row and column. In the cells of the table, note the values included in the given set (in the specific example shown in **Figure 1(a)**, marked with asterisks), also note the values of potential roots (in the figure marked with a green background).

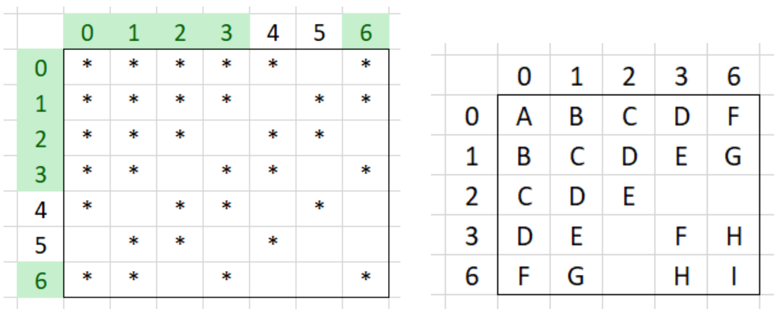
Of course, on each of the diagonals (we call those that are perpendicular to the main diagonal) all the elements correspond to the same value.

It is clear that the problem of extracting the square root can be reformulated as follows: to choose among the "green" elements a subset such that:

- the intersection of each pair of selected elements is marked (with an asterisk);
- on each of the diagonals there is at least one element included in such a pair.

However, with such a reformulation, among the details of this problem, only one reduction to the already mentioned problem of covering the set seems obvious.

Moreover, here we can see a big analogy with the problem of minimizing nondeterministic finite automata, which we have considered in many publications (see [4] [5] [6] [7] [8] and many others): in the root extraction problem solved here, the requirements for the selected subset of potential roots practically coincide with the requirements for the grid (block) to minimize automata. At



(a)

	0	1	2	3	4	5	6
0	*	*	*	*	*		*
1	*	*	*	*		*	*
2	*	*	*		*	*	
3	*	*		*	*		*
4	*		*	*		*	
5		*	*		*		
6	*	*		*			*

(b)

	0	1	2	3	6
0	A	B	C	D	F
1	B	C	D	E	G
2	C	D	E		
3	D	E		F	H
6	F	G		H	I

Figure 1. The first (a) and second (b) variants of the model for the square root in the case of a 1-letter alphabet. Example for the source language [0 1 3 6].

the same time, we will note separately the articles related to the language (automaton) Waterloo [4] [9] [10] and others. Thus, in the model under consideration, we obtain a binary relation, but, unlike the problem of automata minimization, defined on matching sets. In the same way, certain binary relations can be considered as adjacency matrices of the corresponding graphs, which we will do below.

Let us consider **Figure 1** in a little more detail. For it:

- the language used to form the example is [0 1 3 6];

then:

- its square (we consider it to be the source language for this task)

$$[0\ 1\ 2\ 3\ 4\ 6\ 7\ 9\ 12]; \tag{3}$$

- the potential roots are [0 1 2 3 6].

It is easy to show that the potential roots 0 and 6 are necessarily included in the answer, as the minimum and maximum in length.

All of the above (with changes corresponding to a specific task) can be applied to any square root extraction problem, however further reasoning applies only to the problem under consideration. In it, 1 is included in the answer for the reason that without it there can be no 1 in the source language (3), and 3 is included for a similar reason. However, 2 cannot be included in the answer, for example, because the element of this matrix (2, 3) is not the asterisk.

Thus, for this particular task, it is possible to prove that the language [0 1 3 6], taken to form an example, is the only possible answer. However, in the general case, we do not have an appropriate "algorithm". The obvious reduction of the problem is only the consideration of the problem of covering the set. And we have not yet been able "to improve the situation" (*i.e.*, to get a description of the polynomial algorithm), but the subsequent models seem quite interesting.

The next model is some modification of the previous one. Here, the rows and columns of the table are marked only with potential roots (which can be considered logical: after all, only subsets of answers are selected from them), and at the same time, instead of asterisks, we can put the sum at the intersection. But we shall not use numerical values of possible sums, because, apparently, it is more obvious to replace them with some other signs, which we do.

In the example considered for the same source language (3), instead of 9 elements of this language, we put the letters $A\ B \dots H\ I$ in the tables; we get the result in the table shown in **Figure 1(b)**.

At the same time, the goal is somewhat different. We also select a subset of potential roots (in this model, they are clear; besides, we do not need the green background), and the differences from the previous model are as follows:

- the intersection of each pair of selected elements is marked (by some letter);
- any of the letters used should be included in at least one such pair.

It is clear that the result of solving the problem is the same, however, here it is immediately visible, we believe that the description of the corresponding algorithms for finding a subset of the set of potential roots and the implementation of these algorithms in the form of computer programs are much simpler.

As the last simple model, let us consider a graph that can be built on the basis of any of the previous models (of course, the second model is more suitable). The vertices of the graph are marked with potential roots, and the edges are marked with letters that coincide with those shown in the table in **Figure 1(b)**.

We also believe that all vertices have loops belonging to them, and we shall not draw them, but we shall also put the letters corresponding to these loops near all the vertices, which fall together with the letters placed on the main diagonal of the table.

Thus, for our example, we get the graph shown in **Figure 2**.

We shall indicate some references to books on graph theory, with which the names of the concepts used in the article are consistent: [11] [12].

The further material of the article can, somewhat simplifying, be characterized as follows: we shall consider the addition of the graph considered at the end of the previous section. At the same time, we note the following.

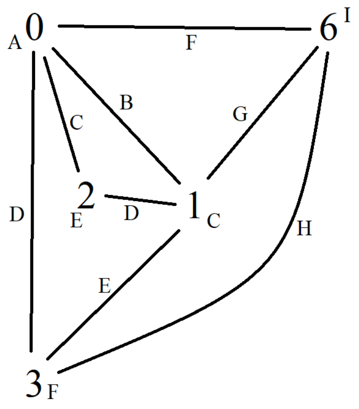


Figure 2. Graph model for the example under consideration.

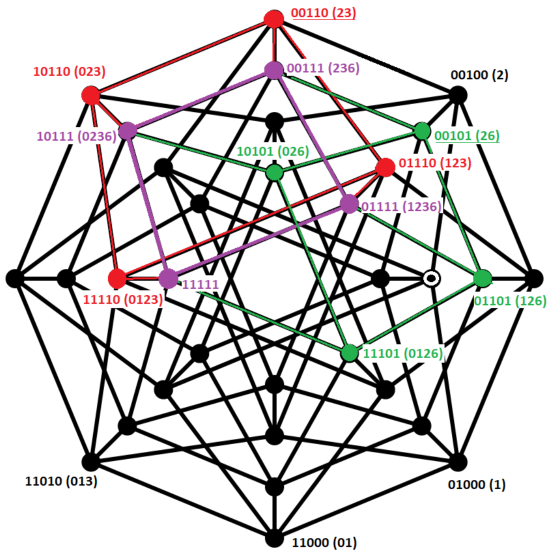


Figure 3. A graph with colored vertices and edges reflecting options for including potential roots.

1) Although we are talking about graphs, we will hardly use the "graph" terminology anymore: it will be applied implicitly.

2) The possible answers found as a result of the models described in the following sections will need to be considered "preliminary" solutions: all of them will need further verification.

4. The Case of the 1-Letter Alphabet and Square Root: Taboo Pairs and Taboo Hyperplanes

In this section, we again assume that N potential roots are obtained based on the input data. For some ordering of these potential roots (the natural ordering in the case of a 1-letter alphabet is an increase in the length of the potential root), consider an N-dimensional cube, in which N binary coordinates of each vertex have the following meaning:

- 0 means that the corresponding potential root is not included in the assumed root-answer;
- 1 means that it is included.

It is clear that the answer to any of the problems of extracting the root can be obtained by iterating over the vertices of such an N-dimensional cube, and each check is carried out in a trivial way in polynomial time. However, due to the fact that the total number of vertices of the cube is 2^N, all such final root extraction algorithms are obtained exponentially.

For further consideration, we shall consider the pairs of potential roots; note in advance that the enumeration of all pairs is polynomial (carried out by a trivial quadratic algorithm). Since each such pair represents two potential roots, we can tell whether their product is valid (whether the sum of their lengths is included in a given word). If the product is not valid, then we shall call such a pair of potential roots the taboo (the taboo pair).

Each such pair in the considered N-dimensional cube defines an N-2-dimensional hyperplane (we will call each of them a taboo plane) and all elements of this hyperplane are obviously not roots.

However, of course, it is impossible to solve the root extraction problem only on the basis of this fact: for example, the vertex of an N-dimensional cube, all coordinates of which are equal to 0, is not included in any taboo plane by definition (since all vertices of all taboo planes have at least two coordinates equal to 1), but when this answer can only be in degenerate cases.

But, despite the example given, the title "taboo plane" fully corresponds to the application of these hyperplanes in the problem we are considering. The application of the set of all taboo planes is understandable. Exactly, based on the material of this article, it can be shown that some improvement in the algorithm for finding (at least one) root is possible: it is enough to sort through only such vertices of an N–dimensional cube (we assume that the vertex is $B = (b_1\ b_2 .. b_N)$, and the source data itself is $A \in 2^{2n}$), which:

- they do not enter into any taboo plane; *i.e.* $(\forall\, i, j \in 1, \dots, n,\ i \neq j)\ (\overbrace{\#i, \#j} \notin$

T(A));

- they have the property ($\exists\ i \in 1, \ldots, n, b_i=0$) ($\exists\ j \in 1,\ldots,n, b_j=1$) ($\overbrace{\#i,\#j} \in$ T(A)).

That is, when changing the value of some coordinate from 0 to 1 (i-th coordinates in the given entry), the previous condition will be violated; only among such vertices of an n-dimensional hypercube and it is necessary to check the fulfillment of condition $B^2 = A$.

Let us consider a specific example. 1-letter alphabet, 2nd degree, at the entrance is the square of the language B = [0 1 3 6] (we specifically note that this language B is unknown in the formulation of the problem), then $A = B^2$ = [0 1 2 3 4 6 7 9 12].

Theotential roots in our example are as follows: B = [0 1 2 3 6], and the set of taboo pairs is

$$T = \{\overbrace{2,3}, \overbrace{2,6}\} = \{\overbrace{\#3,\#4}, \overbrace{\#3,\#5}\}.$$

Let us consider **Figure 3** of a 5-dimensional hypercube. In this figure:

- For clarity, not all of the 32 vertices of the cube (so as not to "overload" the drawing with information), but the markings of the rest can be determined based on the markings given. The marks are 5-digit binary numbers containing, if necessary, insignificant 0 in the highest digits, they are written before the notation in parentheses (see below).
- At the same time, the "origin of coordinates" is not specifically marked with binary code (so as not to "overload" the drawing); this is a vertex drawn with a black inner circle.
- In parentheses after the binary notation of the hypercube vertex, we write the values of the corresponding potential roots without spaces; these are the values themselves, not their numbers (*i.e.*, in the example under consideration, 2, not #3), and we select only those values that correspond to 1 in the binary notation of the hypercube vertex.
- The hyperplane (actually an ordinary 3-dimensional cube) corresponding to the taboo pair is shown in red $\overbrace{2,3}$. The "anchor" vertex of the hyperplane, *i.e.* the vertex in which only two coordinates defining this hyperplane have values of 1, is underlined.
- Similarly, the hyperplane corresponding to the taboo pair $\overbrace{2,6}$ is shown in green.
- The intersection of these two hyperplanes (actually an ordinary square) is shown in purple. Of course, the maximum point of the hypercube belongs to this intersection.

We can consider such a hypercube in the form of a boolean (and it is customary to draw a boolean differently), it is more convenient to work with the latter because you can draw "edges up" from any of its taboo vertices, while marking new vertices (which we get into at the same time) as taboo ones. But, certaily,

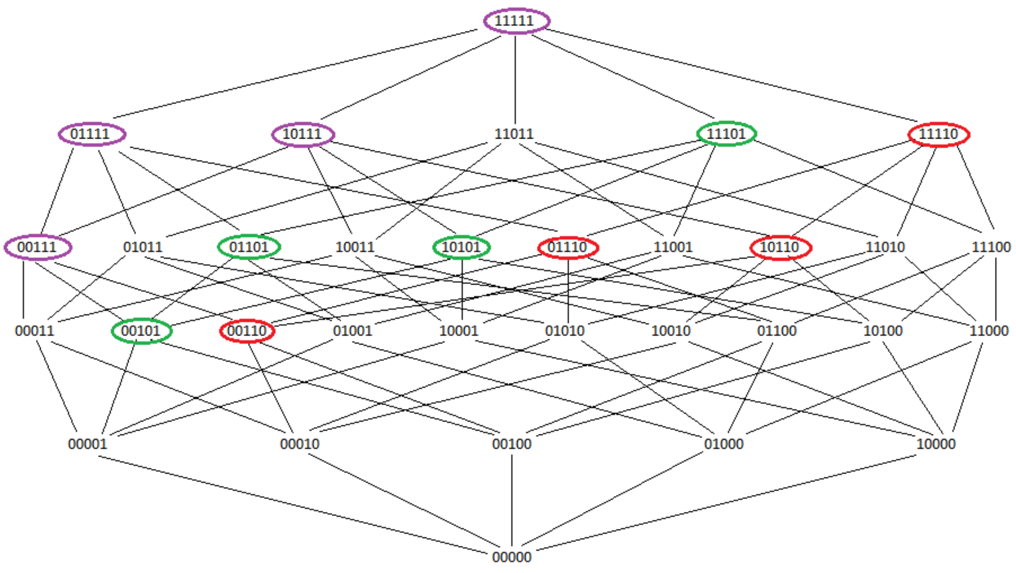


Figure 4. The Boolean graph of dimension 5 for the problem under consideration [0 1 3 6].

these 2 variants of graphs are isomorphic, **Figure 4** shows the Boolean graph for the problem under consideration. The same colors are used as in **Figure 3**. In fact, the graph is isomorphic to the graph shown in **Figure 4**.

So, we will consider a Boolean corresponding to an N-dimensional cube.

5. On Adding a New Coordinate to an Existing Solution

The material in this section seems to be the most non-trivial in this article. It is on the basis of the results of the theorem proved below that we can assert that the search for (at least one) root can be performed as a search for the vertex of the hypercube (Boolean) closest to the maximum vertex and at the same time not included in any of the taboo planes. It can also be argued that the problem can be solved in the same way for an arbitrary alphabet.

Theorem 1. Let B = $(b_1\ b_2 \ldots\ b_N) \in 2^N$ be some root of equation (1), $M = 2$. Let $b_k = 0$ for some $k \in 1,\ldots, N$. Let also for every $i \in 1,\ldots, N \setminus \{k\}$, such that $b_i = 1$, the condition $\overbrace{k,i} \notin$ T(A) is satisfied. Then $B_{+k} = (b_1\ b_2 \ldots\ b_N)_{+k}$ is also the root of equation (1).

Proof. Obviously, $B^n \subseteq (B_{+k})^n$. Therefore, if the inequality $(B_{+k})^n \neq A$ were fulfilled, then there would be a rank $i \in 1,\ldots, N \setminus \{k\}$, such that $b_i = 1$, and at the same time, when squared, the product would have a new a digit equal to 1. Since we have added only one, the k-th digit, to B, the latter fact is possible only at $\overbrace{k,i} \in$ T(A), which contradicts the condition of the theorem. □

Thus, it is on the basis of the results of this theorem that it can be argued that in order to find at least some solution to the original problem, it is possible to look for "untaboo" Boolean points lying closest to the maximum vertex. The solution (if there is at least one) will necessarily be among those "untaboo" points for which there is no neighboring "untaboo" one lying closer to the maximum vertex.

Founding

This work was supported in part by the Higher Education Stability Support Program of Chinese Universities (section “Shenzhen 2022 - Science, Technology and Innovation Commission of Shenzhen Municipality”).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Salomaa, A. (1981) Jewels of Formal Language Theory. Computer Science Press, Rockville, Maryland, 144 p.

[2] Hopcroft, J., Motwani, R. and Ullman, J. (2006) Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company Reading, Massachusetts, 550 p.

[3] Melnikov, B. (2022) Semi-Lattices of the Subsets of Potential Roots in the Problems of the Formal Languages Theory. Part I. Extracting the Root from the Language. *International Journal of Open Information Technologies*, **10**, 1-9. (in Russian)

[4] Kameda, T. and Weiner, P. (1970) On the State Minimization of Nondeterministic Finite Automata. *IEEE Transactions on Computers*, **C-19**, 617-627. https://doi.org/10.1109/T-C.1970.222994

[5] Jiang, T. and Ravikumar, B. (1993) Minimal NFA Problems Are Hard. *SIAM Journal on Computing*, **22**, 1117-1141.

[6] Melnikov, B. and Vakhitova, A. (1998) Some More on the Finite Automata. *The Korean Journal of Computational and Applied Mathematics* (*Journal of Computational and Applied Mathematics*), **5**, 495-505. https://doi.org/10.1007/BF03008877

[7] Lombardy, S. and Sakarovitch, J. (2008) The Universal Automaton. Logic and Automata. Amsterdam University Press, 457-504.

[8] Melnikov, B. (2018) Regular Languages and Nondeterministic Finite Automata (Monograph). Russian Social State University, Moscow, 179 p. (in Russian)

[9] Melnikov, B. (1995) Subclasses of the Context-Free Languages Class (Monograph). Moscow State University, Moscow, 174 p. (in Russian)

[10] Dolgov, V. and Melnikov, B. (2014) On the Automatic Construction Algorithms of Waterloo-Like Finite Automata Based on Complete Automata. *Heuristic Algorithms and Distributed Computing*, **1**, 24-45. (in Russian)

[11] Eremeev, A., Zaozerskaya, L. and Kolokolov, A. (2000) The Problem of Covering a Set: Complexity, Algorithms, Experimental Studies. *Discrete Analysis and Operations Research*, *Ser*. 2., **7**. 22-46. (in Russian)

[12] Harary, F. (1969) Graph Theory. Addison-Wesley Publishing, Massachusetts, 274 p. https://doi.org/10.21236/AD0705364