Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2024, 12, 4322-4332
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

On Some Systems of Word Equations for Automata

Nikolai I. Krainukov[1*], Mikhail E. Abramyan[1,2], Boris F. Melnikov[1]

[1]Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU-BIT University, Shenzhen, China
[2]Algebra and Discrete Mathematics Department, Southern Federal University, Rostov-on-Don, Russian Federation
Email: *n.krainiukov@smbu.edu.cn

How to cite this paper: Krainukov, N.I., Abramyan, M.E. and Melnikov, B.F. (2024) On Some Systems of Word Equations for Automata. *Journal of Applied Mathematics and Physics*, **12**, 4322-4332.
https://doi.org/10.4236/jamp.2024.1212265

Received: October 26, 2024
Accepted: December 27, 2024
Published: December 31, 2024

Abstract

In this paper, we use some programing tools and algorithms for solving system of word equation for regular languages. There are many possibilities for presentation of regular languages such as grammars, finite automata, rewriting systems and so on. Some of these systems is presented by system of computational discrete algebra GAP and the possibilities of presentation now in some systems interactive theorem provers (Isabelle, Coq). This computer system can give to detailed understanding of solution of system of word equation, compared the languages and regular expressions of the languages.

Keywords

Finite Automata, Word Equation, Free Algebra, Regular Expressions, Normal Forms

1. Introduction

Formal Languages, Automata and Logic are basic concepts of computer science. In this contribution, we shall see how different presentation for regular languages helps us to solve problems of minimization finite automata and find the normal forms for classes of equivalence of factor algebra of free algebra with generators in alphabet Σ.

The theory of automata [1] and system of word equation already was considered in [2]. Our goal is to pay more attention to applied work with automata, regular expressions and to receive the results of these calculations.

We discussed also minimization problems for the finite automata.

Section 2 contains the basic definition and notation of formal languages, rewriting system, finite automata and some logical notation.

In Section 3, we discuss the algorithm of Knuth-Bendix for constructing a confluent rewriting system for language and a practical example of application of the algorithms.

In Section 4, we apply the computer discrete algebra system GAP to find normal form state languages for automatons.

2. Definitions and Notation

In this section, we remember the definition and notation about formal languages, free monoid, free algebra, automata and rewriting system. The following definitions taken from [3]-[6] will be used.

An alphabet Σ is finite set letters $\Sigma = \{a, b, c, \ldots\}$. A word or string w is finite length sequence of letters over alphabet Σ. We denote as Σ^* the set of all finite words. The set of Σ^* with respect to the concatenation operation forms a free monoid. Semigroup $\Sigma^+ = \Sigma^* \setminus \varepsilon$ is monoid Σ^* without empty word ε. Language L is subset of monoid Σ^*.

A basic operation of free monoid Σ^* is concatenation of two words $w = uv$. The operation of concatenation is defined for languages in the natural way:

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$$

The concatenation closure or Kleene star of a language L:

$$L^* = \{\varepsilon \cup L \cup L^2 \cup \ldots\}$$

Infinite tree presents of monoid $\Sigma^* = \{\varepsilon, a, b, aa, ab, bb, \ldots\}$ over alphabet $\Sigma = \{a, b\}$ (**Figure 1**). Every word w has unambiguous path from root (node 1) of this tree to the leaves.

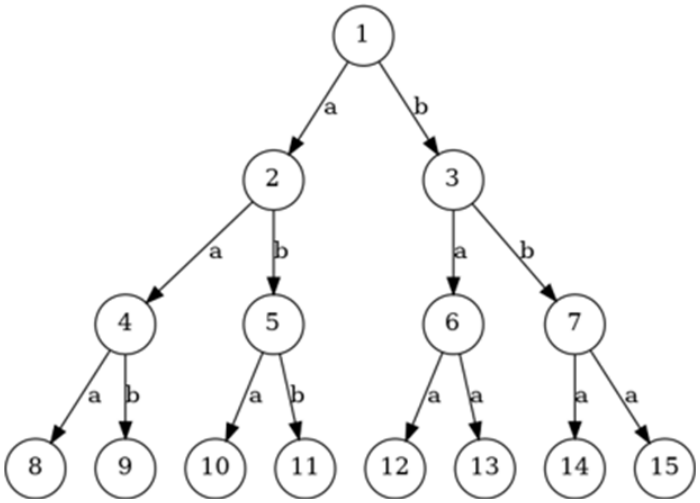

Figure 1. Infinite tree presents the free monoid over alphabet $A = \{a, b\}$.

The word u is a prefix of a word v, denoted as $u \le v$, if $v = uw$, for some $w \in \Sigma^*$. We say that u and v are prefix comparable if either $v \le u$, or $u \le v$.

An automaton A [1] [3]-[5] over alphabet Σ consists of a set of states Q,

the initial states $I \subset Q$, the final/terminal states $T \subset Q$, and a set $F \subset Q \times A \times Q$ called the set of edges. The automaton is denoted by $A = (Q, F, I, T)$.

The automaton is finite when the set Q is finite. The language L is recognized by A, denoted $L(A)$, is the set of words in Σ^* which are labels of paths from I to T.

Figure 2 shows the automaton A with three states, the set of initial states $I = \{1\}$, the set of terminal states $T = \{3\}$, the set of edges $F = \{(1,a,1),(1,a,2),(1,a,3),(1,b,3),(3,b,3)\}$. The finite language $L(A) = \{a, ab, b, \ldots\}$ is recognized by automaton A.

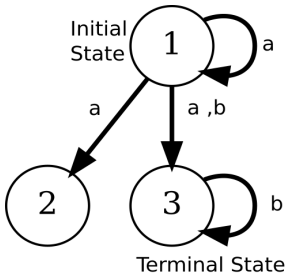

Figure 2. Automaton A with three states.

Let M be a monoid [3] [5]. Recall that a subset $\Sigma \subset M$ of M generates M if every element of M is a product of elements of Σ. If M is a monoid generated by a finite set Σ then there is a homomorphism $\phi : \Sigma^* \to M$, from free monoid Σ^* to monoid M:

$(\alpha\phi)(\beta\phi) = (\alpha\beta)\phi$ for all words α and β in Σ^*.

In this case, the monoid M is isomorphic to $\Sigma^* / \approx$ where $\approx$ is the congruence, equivalence relation compatible with respect to the concatenation on Σ^* defined by:

$$\alpha \approx \beta \Leftrightarrow \alpha\phi = \beta\phi.$$

Let this congruence R is a set of equations of the form $\alpha = \beta$ where $\alpha, \beta \in \Sigma^*$ and where α and β represent the same element $\alpha\phi = \beta\phi$ of monoid M Then R generates a congruence $\approx$ on Σ^* and that R is a set of defining relations for M.

A string-rewriting system R is a subset of $\Sigma^* \times \Sigma^*$. Each element $(l, r) \in R$ of rewriting system R is a (rewrite) rule. Suppose an element $u \in \Sigma^*$ has a subword l and (l, r) is a rule of the rewriting system R, then we can replace the subword l of u by the subword r and obtain a new word v

$$u = xly \to xry = v \quad \text{and} \quad u = xly \leftrightarrow xry = v.$$

For any string-rewriting system R, if u and v are strings such that $u \leftrightarrow_R v$, then for all $x, y \in A^*$, $xuy \leftrightarrow_R xvy$, that the relation $\leftrightarrow_R$ is called a congruence relation too: it is an equivalence relation that is compatible with respect to the concatenation of strings.

We can define the word equation by two steps [2].

First step, we define the expressions over the alphabet A in the variables $X_1,\ldots,X_n$ for given set $CONST$ coefficients of equation and OP set of operation:

$$EX_A\left(CONST;OP;X_1,\ldots,X_n\right),$$

Let $A=\{a,b\}$, $CONST$ is the class of finite languages, and OP contains precisely union and left-concatenation.

For example, the expression α is contained in $EX_A\left(CONST;OP;X,Y,Z\right)$,

$$\alpha = abXUaYUbZ \in EX_A\left(CONST;OP;X,Y,Z\right).$$

Second step, now we can define a system of language equations.

Let $\alpha_1,\alpha_2,\ldots,\alpha_k$ and $\beta_1,\beta_2,\ldots,\beta_k$ be expression in $EX_A\left(CONST;OP;X_1,\ldots,X_n\right)$,

$$\begin{aligned}\alpha_1 &= \beta_1\\ \alpha_2 &= \beta_2\\ &\cdots\\ \alpha_k &= \beta_k\end{aligned}$$

Then this is a system of language equations for variable X,Y,Z, set $CONST$ and OP:

$$\begin{aligned}aX &= Yb\\ aZa &= Y,\end{aligned}$$

The possible solution of this system of equations:

$$X=(ab)^n, Y=a(ab)^{n-1}a, Z=(ab)^{n-1}$$

Recall that an algebra [6] [7] over a field K is a K-vector space A with a binary operation (multiplication) $A\times A\to A$, $(a,b)\to ab$ specified on it, satisfying the following requirements:

1) $a(b+c)=ab+ac$, $(b+c)a=ba+ca$ for any $a,b,c\in A$;

2) $(\lambda a)b=a(\lambda b)=\lambda(ab)$ for any $\lambda\in K$, $a,b\in A$.

We will additionally assume that:

3) there is a unit in A, *i.e.*, an element 1 such that $1a=a1=a$ for any $a\in A$;

4) algebra A is associative, *i.e.*, $(ab)c=a(bc)$ for any $a,b,c\in A$.

Throughout the following, we will additionally assume, that the field K is the field of rational numbers. We can embed monoid Σ^* over alphabet $\Sigma=\{x_1,x_2,\ldots,x_n\}$ into free algebra of polynomials $K[x_1,x_2,\ldots,x_n]$ with homomorphism $\varphi:\Sigma^*\to K[x_1,x_2,\ldots,x_n]$ by definition on letters of alphabet Σ:

$$\varphi(x_i)=x_i, i=1,\ldots,n.$$

Then we can define a lineal presentation [8] of syntactical monoid of the deterministic automaton A by matrices correspondent the transformations of the letters alphabet.

3. Finite Automaton and the System of Equations

At first, we consider the deterministic finite automata [9] [10]. Consider a

deterministic finite automaton (DFA) $DA=(A,Q,\delta,q_0,F)$, where A is the underlying alphabet, $Q=\{q_0,q_1,\ldots,q_{n-2},q_{n-1}\}$ is the finite, nonempty set of states, q_0 is the (single) initial state ($q_0\in Q$), F a subset of ($F\subseteq Q$), is the set of final states, and δ is the transition function:

$$\delta: Q\times A\to Q.$$

A system of word equations associates with DA in the following way:

- CONST is a class of finite languages (it may contain one word).
- OP consists of the operations union and left-concatenation.
- The set of variables be $\{L_i, q_i\in Q\}$

$$L_i=\bigcup a\cdot L_j U\lambda(q_i),$$
$$a\in A, q_j=\delta(q_i,a)$$

where $\lambda(q_i)=\varepsilon$ if $q_i\in F$ and $\lambda(q_i)=\varnothing$ if $q\notin F$.

Let L_i is the language of automaton $DA=(A,Q,r,q_i,F)$ with initial state q_i. If the state q_i is the final state $q_i\in F$ of automaton DA, then the empty word ε belongs language L_i.

For example, **Figure 3** shows the automaton DA with four states.

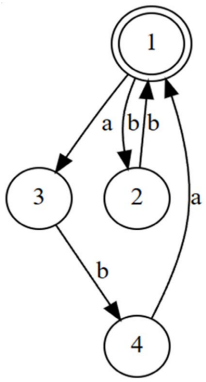

Figure 3. The automaton DA with four states.

The language $L_1=(abaUbb)^*$ are the set of the paths from initial state $q_1=1$ to the final state $q_1=1$.

The system of word equations for automaton on **Figure 3** can be written in form:

$$L_1=aL_3\cup bL_2\cup\varepsilon$$
$$L_2=bL_1$$
$$L_3=bL_4$$
$$L_4=aL_1$$

The solution of this system of equations can be found by substitution or by Gauss's method.

$$L_1=abL_4\cup bL_2\cup\varepsilon=abaL_1\cup bbL_1\cup\varepsilon$$

$$L_1=(aba\cup bb)L_1\cup\varepsilon,\quad L_1=(aba\cup bb)^*$$

Lemma 1. Let the equation $X = N \cdot X \cup M$ over the alphabet A in the variable. Then solution of this equation: $X = N^* M$. (Just substitute it into the equation)

We use the system of computational discrete algebra GAP for calculation with finite automata. There are many functions with operation with finite automaton in package "Automata" such as for creating automata, minimization the number of states, determination and so on.

```
gap> aut1:=Automaton("det",4,2,[[3,,,1 ],[2 ,1,4, ]],[1],[1]);
< deterministic automaton on 2 letters with 4 states >
gap> Display(aut1);
  |  1  2  3  4
----------------
a |  3        1
b |  2  1  4
Initial state:   [ 1 ] Accepting state: [ 1 ]
gap>re1:=AutomatonToRatExp( aut1 );
(abaUbb)*
```

Let consider a nondeterministic finite automaton (NFA) $NA = (A, Q, \nu, I, F)$, where A is alphabet, Q is the finite set of states, I is the set initial state ($I \subset Q$), F, is the set of final states, and ν is the transition function:

$$\nu : Q \times A \to 2^Q.$$

For example, **Figure 4** shows the nondeterministic automaton NA with three states.

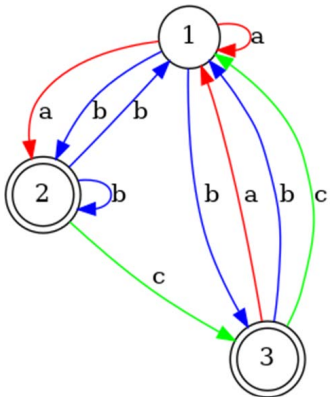

Figure 4. The automaton NA with three states.

The corresponding system of equations for nondeterministic automaton NA is then:

$$L_1 = aL_1 \cup aL_2 \cup bL_2 \cup bL_3$$

$$L_2 = bL_1 \cup bL_2 \cup cL_3 \cup \varepsilon$$

$$L_3 = aL_1 \cup bL_1 \cup cL_1 \cup \varepsilon$$

The solution of this system of equations can be find by substitution or by Gauss's method:

$$L_1 = aL_1 \cup (a \cup b)L_2 \cup b(a \cup b \cup c)L_1 \cup b$$

$$L_2 = bL_1 \cup bL_2 \cup c(a \cup b \cup c)L_1 \cup c \cup \varepsilon$$

$$L_3 = (a \cup b \cup c)L_1 \cup \varepsilon$$

The usage of the system algebra GAP gives the solution of these system equations:

```
gap> Display(naut1);
  |  1          2         3
--------------------------------
 a | [ 1, 2 ]             [ 1 ]
 b | [ 2, 3 ]   [ 1, 2 ]   [ 1 ]
 c |            [ 3 ]      [ 1 ]
Initial state:   [ 1 ]
Accepting states: [ 2, 3 ]
gap> ren1:=AutomatonToRatExp( naut1 );
((aUb)b*(c(aUbUc)Ub)Ub(aUbUc)Ua)*((aUb)b*(cU@)Ub)
 gap> naut2:=Automaton("nondet",3,3,
[[[1,2],,1],[[2,3],[1,2],1],[,3,1]],[2],[2,3]);
< non deterministic automaton on 3 letters with 3 states >
gap> ren2:=AutomatonToRatExp( naut2 );
((c(aUbUc)Ub)(b(aUbUc)Ua)*(aUb)Ub)*((c(aUbUc)Ub)(b(aUbUc)Ua)*bUcU@)
gap> naut3:=Automaton("nondet",3,3,
[[[1,2],,1],[[2,3],[1,2],1],[,3,1]],[3],[3,3]);
< non deterministic automaton on 3 letters with 3 states >
gap> ren3:=AutomatonToRatExp( naut3 );
((aUbUc)((aUb)b*bUa)*((aUb)b*cUb))*((aUbUc)((aUb)b*bUa)*(aUb)b*U@)
```

The results solutions are the languages $L_1 = \mathbf{ren1}, L_2 = \mathbf{ren2}, L_3 = \mathbf{ren3}$, where regular expressions **ren1, ren2, ren3** build from the correspondence nondeterministic automata **naut1, naut2, naut3**.

For nondeterministic automaton, $NA = (A, Q, \nu, I, F)$, we can define the adjacency matrix:

$$M_a, a \in A, M_{i,j} = 1, \text{if } \nu(q_i, a) = q_j, \text{else } 0.$$

The size of matrix M_a is equal $N \times N$, where N is a number of state nondeterministic automaton NA [1] [11] [12]. There is homomorphism $\varphi : A^* \to AM$ from free monoid A^* to matrix algebra AM with generators M_a.

There is path that is the word w from state q_i to state q_j. The homomorphism $\varphi : A^* \to AM$ maps the word w to a product of matrix G_w and element $G_{i,j}$ is the path from state q_i to state q_j in nondeterministic automaton D.

For example, for automaton NA the matrices M_a, M_b, M_c and G_a, G_b, G_c have the correspondents structure:

```
gap> Ma;
[ [ 1, 1, 0 ],
  [ 0, 0, 0 ],
  [ 1, 0, 0 ] ]
gap> Mb;
[ [ 0, 1, 1 ],
  [ 1, 1, 0 ],
  [ 1, 0, 0 ] ]
gap> Mc;
[ [ 0, 0, 0 ],
  [ 0, 0, 1 ],
  [ 1, 0, 0 ] ]
gap> Ga;
[ [ a, a, 0 ],
  [ 0, 0, 0 ],
  [ a, 0, 0 ] ]
gap> Gb;
[ [ 0, b, b ],
  [ b, b, 0 ],
  [ b, 0, 0 ] ]
gap> Gc;
[ [ 0, 0, 0 ],
  [ 0, 0, c ],
  [ c, 0, 0 ] ]
gap> Aabc:=FreeAssociativeAlgebraWithOne(Rationals,"a","b","c");
<algebra-with-one over Rationals, with 3 generators>
gap>  A:= Algebra( Aabc, [ Ma, Mb, Mc ] );
<free left module over AlgebraWithOne( Rationals, ... ), and ring,
with 3 generators>
```

Lemma. The homomorphism $\varphi: MA \to AM$ map word $w+v \in MA$ free left module MA over free algebra A to the matrix $\varphi(w) = P_{i,j}$, where $P_{i,j}$ – path that the the two word w, v from state q_i to state q_j [5] [6].

Example is continued:

For automaton, NA we have the results:

```
gap> P1:=Ga^3+Gb*Ga*Gc^2;
[ [ (1)*a^3,   (1)*a^3,        <zero> ],
[ (1)*b*a*b^2,  <zero>,          <zero>],
[ (1)*a^3+(1)*b*a*b^2, (1)*a^3, <zero> ] ]
gap> P2:= Ga^2+Gb^2*Ga*Gc^2+Ga*Gb*Gc^2*Ga^5;
[ [ (1)*a^2+(2)*b^2*a*b^2+(2)*a*b^3*a^5,(1)*a^2+(2)*a*b^3*a^5,<zero>],
  [ (1)*b^2*a*b^2,                    <zero> ,              <zero>],
  [ (1)*a^2+(1)*a*b^3*a^5, (1)*a^2+(1)*a*b^3*a^5,           <zero>]]
```

The results are for matrices **P1** show the path from state q_1 to state q_1 by the word $w = \mathbf{a}\text{^}3$ and for matrices **P2** to from state q_3 to state q_2 by the word $w = \mathbf{a} * \mathbf{b}\text{^}3 * \mathbf{a}\text{^}5$.

4. Apply Computer Discrete Algebra System GAP for Find Normal Form of State Languages for Automaton

The computer discrete algebra system GAP has more than 130 different packages. The package name *KBMag* reflects the Knuth-Bendix algorithm and program for constructing a rewriting system from a finitely presented semigroup, monoid or group.

The words in a rewriting system created in GAP for use by *KBMag* are defined over an alphabet that consists of the generators of a free monoid, called the word-monoid of the system.

First, we define nondeterministic automaton NA:

```
gap> aut1:= NFAtoDFA( naut1 );
< deterministic automaton on 3 letters with 7 states >
gap> Display(aut1);
  |  1  2  3  4  5  6  7
-------------------------
 a |  2  2  1  4  2  1  2
 b |  3  5  2  4  5  1  5
 c |  4  6  7  4  7  1  1
Initial state:   [ 1 ]
Accepting states: [ 2, 3, 5, 6, 7 ]
gap> ta:=Transformation([ 2,  2,  1,  4,  2,  1,  2]);
Transformation( [ 2, 2, 1, 4, 2, 1, 2 ] )
gap> tb:=Transformation([ 3,  5,  2,  4,  5,  1,  5]);
Transformation( [ 3, 5, 2, 4, 5, 1, 5 ] )
gap> tc:=Transformation([ 4,  6,  7,  4,  7,  1,  1]);
Transformation( [ 4, 6, 7, 4, 7, 1, 1 ] )
gap> ma1:=Monoid([ta,tb,tc]);
<transformation monoid of degree 7 with 3 generators>
gap> Size(ma1);
73
```

Then we construct syntactic monoid for transformation states of deterministic automaton **aut1.** The size of syntactic monoid **ma1** is equal 73.

```
MappingByFunction( <transformation monoid of size 73, degree 7 with 3
generators>, <fp monoid on the generators
[ m1, m2, m3 ]>, function( x ) ... end, function( x ) ... end )
gap> m1:=Image(mhom1);
<fp monoid on the generators [ m1, m2, m3 ]>
gap> k1:=KnuthBendixRewritingSystem(m1);
```

```
R := KBMAGRewritingSystem( m1 );
gap> Size( R );
73
gap> GrowthFunction( R );
x_1^6+12*x_1^5+25*x_1^4+22*x_1^3+9*x_1^2+3*x_1+1
```

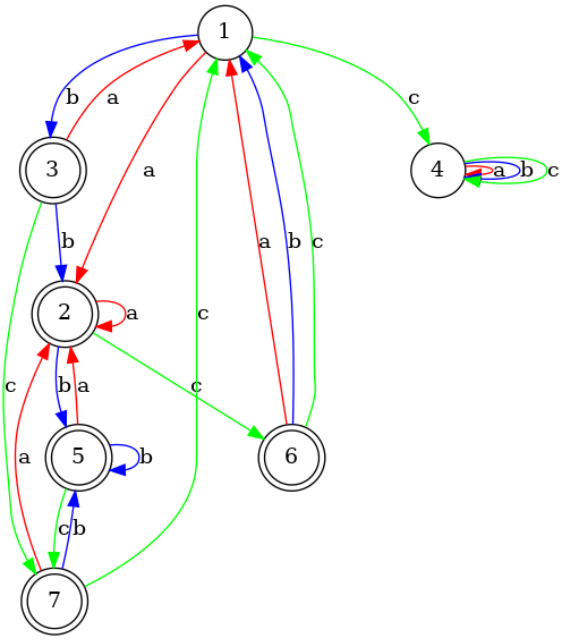

Figure 5. Deterministic automaton ***aut*1** with seven states.

Returns the growth function of the set of irreducible words in the rewriting system R. This is a rational function, of which the coefficient of x^n in its Taylor expansion is equal to the number of irreducible words of length n.

This deterministic automaton *aut*1 seven states and the syntactic monoid *ma*1 has seven congruence classes.

5. Conclusions

The system GAP has many packages to solve problems with finite automata, formal languages and other algebraic structures: monoid, free algebra, matrix algebra.

Usage system GAP for: solving the systems of equations for finite automata, finding the regular expressions for languages describing state-to-state paths, using a rewriting system to reduce regular expression and transform it into its normal form.

Funding

This work is supported by a grant from the research program of Chinese universities "Higher Education Stability Support Program" (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] Aho, A. and Ullman, J. (1973) The Theory of Parsing, Translation, and Compiling. Prentice Hall.

[2] Leiss, E. (1999) Language Equations. Springer-Verlag. https://doi.org/10.1007/978-1-4612-2156-2

[3] Brauer, W. (1984) Automation Theory: An Introduction to the Theory of Finite Automata. Vieweg + Teubner Verlag.

[4] Rozenberg, G. and Salomaa A. (1997) Handbook of Formal Languages. Volume 1. Word, Language, Grammar. Academic Press. https://doi.org/10.1007/978-3-642-59136-5

[5] Lallement, G. (1979) Semigroups and Combinatorial Applications. Wiley & Sons, 376.

[6] Berstel, J. and Perrin, D. (2008) Theory of Codes. Academic Press, 345.

[7] Winberg, E.B. (2005) Course of Algebra. M. Factorial Press. (In Russian)

[8] Berstel, J. and Perrin, D. (2005) Codes and Automata. Springer, 545.

[9] Melnikov, B. (2017) The Complete Finite Automaton. *International Journal of Open Information Technologies*, **5**, 9-17.

[10] Melnikov, B. (2018) Regular Languages and Nondeterministic Finite Automata. RGSU Publisher. (In Russian)

[11] Melnikov, B. and Dolgov, V. (2022) Simplified Regular Languages and a Special Equivalence Relation on the Class of Regular Languages. Part I. *International Journal of Open Information Technologies*, **10**, 12-20. (In Russian)

[12] Abramyan, M. (2021) Computing the Weight of Subtasks in State Minimization of Nondeterministic Finite Automata by the Branch and Bound Method. *University proceedings. Volga Region. Physical and Mathematical Sciences*, **2**, 46-52. (In Russian)