Some new heuristical algorithms for minimization of nondeterministic finite automata

B. Melnikov[a], V. Dudnikov[b]

[a] *State Fire Academy of Emercom, 129366, Boris Galushkin Street, 4, Moscow, Russia*
[b] *Togliatti State University, 445020, Belorusskaya Street, 14, Togriatti, Russia*

Abstract

In this paper, we propose an algorithm example for the transformation of so-called complete automaton given by a table of binary relation #. At the same time, we know that for this table for the binary relation #, there exists some corresponding nondeterministic automaton having Waterloo-like badness. The proposed transformation, which is not equivalent, is the serial removal of a state and combining a pair of states. It gives the opportunity to build on the basis of the given relation # some automaton which also has the walibad-property. And, generally speaking, the obtained automaton is different from the known in advance.

Keywords: nondeterministic finite automata; universal automaton; covering set of blocks; covering automaton; Waterloo automaton; basis automaton; complete automaton;

1. Introduction

There are some different invariants for a given regular language; for example, these are some different formalisms for nondeterministic finite automata. Apart the automaton of canonical form, we considered in some papers:

- the basis automaton ([1, 2, 3] etc.);
- the extended basis automaton ([4]);
- and the universal automaton ([3]).

It is important to note, that each of this finite automata is different from the canonical automaton, which is also an invariant for a given regular language. And under certain assumptions, we can consider the complete automaton (automaton $L^{\#}$, see again [3]) also as an invariant for a given regular language.

The universal automaton was considered detailed in [5]. And in [6], we considered algorithms of constructing so-called automaton $\mathcal{COM}(L)$ and obtain a simple proof, that this automaton is equal to the universal automaton (up to renaming states). In fact, we obtained in [6] some algorithms for constructing universal automaton, which are used in this paper.

In some problems of minimization for nondeterministic finite automata ([7, 8, 9, 10, 11] etc.), there is possible a situation, when the covering set of grids ([6, 7, 10]) defines the automaton (so-called covering automaton), which is *not* equal to the given one. Firstly, such an example was obtained in 1970 by Kameda and Weiner, and, following [12], was called Waterloo. The description of this example was given in our terms in [13]; below in this paper, we shall consider some more detailed description. We shall call such constructions *walibads* from ("Waterloo-like badness"), and each *covering* set of grids, which do *not* contain some possible grids, will be called *proto-walibad*. Just note, that such constructions are the property of a language, not of an automaton. Evidently, the existing of such walibads complicates the description of algorithms for minimization of nondeterministic finite automata; at first, it applies to the practical (heuristic) algorithms, see also [13]. Therefore, the problems of finding and description of such constructions are actual; moreover, in the practical algorithms, there is desirable to finding them *before* starting cardinal part of the algorithm.

In this paper, we consider a way to make *automatically* a walibad-construction using the following objects:

- some concrete (known in advance) example of walibad, and, therefore, the known table of binary relation # (see [10] etc.);
- automaton $L^{\#}$ defined by this table # (see [3], and also Subsection 2.3 below).

We emphasize, that in the process of building we used only relation #, and did not use automaton known in advance. This fact makes a possibility to obtain the following objects only on the basis of the given table of relationships # having the property proto-walibad:

- firstly, the searching algorithm for obtaining walibad-construction;
- secondly, the searching algorithm for proving the necessary condition for such construction for an arbitrary nondeterministic automaton;
- and thirdly, the classification of all possible tables of relationships # (having such property) based on the condition of existence of construction walibad corresponding to this table.

This article gives an example of such process. Author hopes that this example is based on clear and common algorithm. He is going to describe the general algorithm in the following publications, together with a proof of the fact that we can automatically receive such construction for any table having walibad-property. Anyway, this process gives the opportunity to realize practical algorithms for constructing *all* automata having the property. In conclusion, we include information about executed computational experiments and also about obtained to date and expected results.

2. Preliminaries

In this section we consider definitions and notation of [10] and some previous our papers. We also state some facts obtained in these papers.

2.1. The main definitions

We shall consider nondeterministic finite automaton

$$K = (Q, \Sigma, \delta, S, F) \tag{1}$$

without ε-transitions, i.e., we shall consider transition function δ of automaton (1) by

$$\delta \colon Q \times \Sigma \to \mathcal{P}(Q).$$

We shall assume, that regular language L is given, and use for it notation defined in our cited paper [10]; for example, automata of canonical form for languages L and L^R (i.e., $\widetilde{L}$ and $\widetilde{L^R}$) will be denoted in the following way:

$$\widetilde{L} = (Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi) \text{ and } \widetilde{L^R} = \left(Q_\rho, \Sigma, \delta_\rho, \{s_\rho\}, F_\rho\right).$$

(We shall not consider language$L = \emptyset$, then both these automata *does have* input states.)

Binary relation # and state-marking functions φ^{in} and φ^{out} are defined as follows. For a pair (A, X) of states of automaton $\widetilde{L}$ and $\widetilde{L^R}$, we have

$$A\#X \text{ if and only if } (\exists uv \in L)\left(u \in \mathcal{L}_L^{in}(A), v^R \in \mathcal{L}_L^{in}(X)\right).$$

A function $\varphi_K^{in} \colon Q \to \mathcal{P}(Q_\pi)$ is defined in the following way:

$$\varphi_K^{in}(q) \ni \tilde{q} \text{ if and only if } \mathcal{L}_K^{in}(q) \cap \mathcal{L}_L^{in}(\tilde{q}) \neq \emptyset.$$

And a function

$$\varphi_K^{out} \colon Q \to \mathcal{P}(Q_\rho)$$

is defined in the same way for automata K^R and $\widetilde{L^R}$. See some examples in [10].

The definition of the basis automaton for the given language L was also given in our previous paper; see, for instance, [1, 2, 3, 10] for its transition function. In this paper, we shall use for it the following notation:

$$\mathcal{BA}(L) = (\hat{Q}, \Sigma, \hat{\delta}, \hat{S}, \hat{F}).$$

For a state $\genfrac{}{}{0pt}{}{A}{X} \in \hat{Q}$ of this automaton, we shall write $\alpha(\genfrac{}{}{0pt}{}{A}{X}) = A$ and $\beta(\genfrac{}{}{0pt}{}{A}{X}) = X$.

2.2. The grids, the universal automaton, etc.

Grids and *pseudo-grids* were defined by [6]. If for some pair $P \subseteq Q_\pi$ and $R \subseteq Q_\rho$ we have

$$(\forall A \in P)(\forall X \in R)(A\#X),$$

then $\mathcal{B} = (P, R)$ is pseudo-grid. For it, we shall write $\alpha(\mathcal{B}) = P$ and $\beta(\mathcal{B}) = R$. For some $A \in P$ and $X \in R$, we also shall write $[\genfrac{}{}{0pt}{}{A}{X} \in \mathcal{B}]$.

If for some pseudo-grid $\mathcal{B} = (P, R)$ there exists:

- neither $A \in Q_\pi \backslash P$ such that $\left((P \cup \{A\}), R\right)$ is also a pseudo-grid;
- nor $X \in Q_\rho \backslash R$ such that $\left(P, (R \cup \{X\})\right)$ is also a pseudo-grid,

then $\mathcal{B}$ is a grid.

The definition of the *automaton* $\mathcal{COM}(L)$ for the given language L was also given in [6]. Thus, considering automata $\widetilde{L}$ and $\widetilde{L^R}$, we define automaton

$$\mathcal{COM}(L) = (Q_{\mathcal{COM}}, \Sigma, \delta_{\mathcal{COM}}, S_{\mathcal{COM}}, \mathcal{F}_{\mathcal{COM}}). \tag{2}$$

Here, $Q_{\mathcal{COM}}$ is the whole set of grids. Other elements of (2) are defined in the following way:

- $S_{\mathcal{COM}} = \{\mathcal{B} \in Q_{\mathcal{COM}} | \alpha(\mathcal{B}) \ni s_\pi\}$;
- $\mathcal{F}_{\mathcal{COM}} = \left\{\mathcal{B} \in Q_{\mathcal{COM}} | \beta(\mathcal{B}) \ni s_\rho\right\}$;
- for some pair $\mathcal{B}_1, \mathcal{B}_2 \in Q_{\mathcal{COM}}$(condition $\mathcal{B}_1 = \mathcal{B}_2$ is possible) and some $a \in \Sigma$, we set

$$\delta_{\mathcal{COM}}(\mathcal{B}_1, a) \ni \mathcal{B}_2$$

if and only if both the following conditions hold:

$$\left(\forall p \in \alpha(\mathcal{B}_1)\right)\left(\delta_\pi(p, a) \in \alpha(\mathcal{B}_2)\right) \tag{3}$$

$$\left(\forall r \in \beta(\mathcal{B}_2)\right)\left(\delta_\rho(r, a) \in \beta(\mathcal{B}_1)\right) \tag{4}$$

We also have shown in [6], that $\mathcal{COM}(L)$ coincides with the *universal* automaton for language L (up to renaming states).

Like [10], we define the *covering set* of *grids*: some set $\mathcal{Q} \subseteq Q_{\mathcal{COM}}$ is such one, if

$$(\forall_X^A \in \#)(\exists \mathcal{B} \in \mathcal{Q})([\genfrac{}{}{0pt}{}{A}{B} \in \mathcal{B}]). \tag{5}$$

For the given covering set of grids $\mathcal{Q}$, we define the *covering automaton:*

$$\mathcal{COM}_{\mathcal{Q}}(L) = (\mathcal{Q}, \Sigma, \delta_{\mathcal{COM}}, S_{\mathcal{COM}} \cap \mathcal{Q}, \mathcal{F}_{\mathcal{COM}} \cap \mathcal{Q}). \tag{6}$$

Certainly, the transition function is defined in (6) for states of $\mathcal{Q}$ only.

We shall consider some examples in Section 3.

2.3. The complete automaton

The complete automaton, defined in [3] by the given table of relation # and two input states (i.e., s_π and s_ρ in the notation above), is designed as $L^{\# s_\pi s_\rho}$ (or, simply, as $L^\#$). It is defined in the following way.

$$L^{\# s_\pi s_\rho} = L^\# = (Q_\pi, \Sigma_\#, \delta_\#, \{s_\pi\}, F_\pi),$$

where:

- $\Sigma_\# \{a_{\frac{A}{X}} | A \in Q_\pi, X \in Q_\rho\}$ is the new alphabet;
- $F_\pi = \{f_\pi \in Q_\pi | f_\pi \# s_\rho\}$ is the set of final states;.
- $\delta_\# \left(A, a_{\frac{A}{X}}\right) = \begin{cases} \{B\}, & \text{if } A\#X \\ \emptyset, \text{otherwice} \end{cases}$ is the transaction function.

In fact, using only the given relation #, we obtain an automaton corresponding to this relation.

Thus, for the given regular language L, we have defined the following corresponding objects: two canonical automata, $\tilde{L}$ and $\widetilde{L^R}$, including their states, transition functions etc.; binary relation #; state-marking functions φ^{in} and φ^{out}; basis automaton $\mathcal{BA}(L)$; automaton $\mathcal{COM}(L)$; automaton $L^\#$. For details, see [3, 6, 10].

3. Automaton Waterloo and an nonequivalent covering automaton

Consideration of automaton Waterloo according to our terminology carried at [13]. Let us now show the existence of automaton, for which:

- its states have marking functions φ^{in} and φ^{out} covering all the elements of considered relation #;
- there exists all the possible transitions; in other words, we choose all the transitions of automaton $\mathcal{COM}(L)$ (i.e., the transition satisfying both the conditions (3) and (4));
- however, its language is not L (i.e., language of the given automaton).

For this thing, consider grids corresponding the following states of automaton $\mathcal{COM}(L)$:

$$1, 3, 5, 6, 8, 10, 12 \qquad (7)$$

evidently, that all the 20 elements of relation # (Tab 5) are covered by these states.

Constructing all the possible transitions (i.e., transitions, satisfying (3) and (4)), we obtain automaton of Table 1. The determinization of this automaton gives Table 2. After renaming states in the following "natural" way

$$\{1\} = A, \{3\} = B, \{5\} = C, \{6\} = D, \{6, 8\} = E, \{8,10\} = F, \{10, 12\} = G, \{12\} = H$$

we obtain automaton of Tab. 3. Evidently, it is *nonequivalent* to the given one, because it state F do not contain a-transition, and all other transitions, inputs and outputs coincide with the same objects of the given automaton.

Table 1. An automaton obtained by constructing all possible transitions

	a	b
$\rightarrow 1$	$6, 8$	-
3	$8, 10$	-
5	$10, 12$	3
6	5	12
$\leftarrow 8$	-	-
10	-	6
12	1	-

Table 2. The determinization of automaton

	a	b
$\rightarrow 1$	$6, 8$	-
$\leftarrow 6, 8$	5	12
5	$10, 12$	3
12	1	-
$10, 12$	1	6
3	$8, 10$	-
6	5	12
$\leftarrow 8, 10$	-	6

Table 3. Automaton with renamed states

	a	b
$\rightarrow A$	E	-
B	F	-
C	G	B
D	C	H
$\leftarrow E$	C	H
$\leftarrow F$	-	D
G	A	D
H	A	-

4. Walibads and proto-walibads

In this section, we consider strict definitions of walibads and proto-walibads.

Definition 1. *If for a covering set of grids (defined by (5)) we have $Q \neq Q_{\mathcal{COM}}$, then we shall call this set by **proto-walibad**.*

Example 1 Consider the following table of relation #:

Table 4. Table of relation #

	X	Y	Z
A	#	#	
B	#	#	#
C		#	#

For this relation, we have the following grids:

$$(1)\ \{A,B\} \times \{X,Y\}\ (2)\ \{B,C\} \times \{Y,Z\}$$
$$(3)\ \{B\} \times \{X,Y,Z\}\ (4)\ \{A,B,C\} \times \{Y\}$$

A proto-walibad is the set consisting of grids (1) and (2). Remark once again, that, by Subsection 2.3, we can construct finite automaton corresponding the given table of relation #.

Definition 2. *If for the given regular language (or for the given finite automaton) there exists a proto-walibad, for which the covering automaton is not equal to the given one, then the given language (the given automaton) is called **walibad**.*

Remark that automaton Waterloo satisfies this definition (see Sections 3. However, we can show that there exists *no* walibads having the table of relation # of Example 1. (I.e., there exist proto-walibads, which are *not* walibads.) For instance, this fact can be shown by brute force algorithm, which may be implemented using the remainder of this paper. However, we will not consider this example in more details.

5. Constructing walibad using its table of

In this section, we consider an example of *inequivalent* transformation of automaton $L^{\#}$. This transformation preserves the given table of binary relations #.We give a specific example of work of such an algorithm. In the next paper, we shall show how to obtain *any* walibad based on this example. Such sequence of actions can be obtained using some brute force algorithm modifying the complete automaton $L^{\#}$ for the given language L; in fact, it uses the given binary relation # only.

Thus, algorithm of this section use *some previously known* walibad. Consider once again automaton Waterloo. Remark that the language of its automaton $L^{\#}$ includes 80 letters (i.e., the table of transition of automaton $L^{\#}$ for language Waterloo includes 80 columns). Therefore, we do not give the entire table in this paper, we shall consider the used letters only.

We mark corresponding cell by gray (see left top shaded cell of Tab. 5 below). From the table of automaton $L^{\#}$, we similarly choose all the columns (letters) marked by a (on Tab. 5, they are also marked by gray):

Table 5. Constructing walibad using its table of #

	$\overset{E}{Y}$	$\overset{F}{Y}$	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{G}{U}$	$\overset{F}{V}$	$\overset{G}{V}$	$\overset{C}{W}$	$\overset{B}{P}$	$\overset{C}{P}$	$\overset{E}{Q}$	$\overset{A}{S}$
$\rightarrow A$	E	F									E	
B	E	F				F	G					
C					G	F	G					
D								C	B	C		
$\leftarrow E$								C	B	C		
$\leftarrow F$			A	B					B	C		
G			A	B								A
H			A	B								A

Let us remark, that for each row of the obtained table, all the painted cells have the same value; besides, their value is the value of transition function of canonical automaton (see before). Then we can combine all corresponding columns of the obtained table by the brute force algorithm while the process of obtaining a walibad, "absorbing" unmarked cells by marked ones (see Tab. 6 for letter a):

Table 6. Process of obtaining a walibad

$L^{\#}$	a
$\rightarrow A$	E
B	F
C	G
D	C
$\leftarrow E$	C
$\leftarrow F$	B
G	A
H	A

Similarly, we use the given letter b (see Tab. 7 and Tab. 8). Thus, combining tables for letters a and b, we obtain transition function for (canonical) automaton Waterloo.

Table 7. Process of obtaining a walibad

	U^{B}	W^{H}	R^{D}
$\rightarrow A$			
B			
C	B		
D		H	
$\leftarrow E$		H	
$\leftarrow F$			D
G			D
H			

Table 8. Process of obtaining a walibad

$L^{\#}$	b
$\rightarrow A$	-
B	-
C	B
D	H
$\leftarrow E$	H
$\leftarrow F$	D
G	D
H	-

Theorem 1. *Let some table of binary relation # be given. Let there exist corresponding automaton K and for it, there exists covered automaton which is not equivalent to K, Then we can obtain K by brute force algorithm from the automaton $L^{\#}$, deleting some letters of $L^{\#}$ and combining some other pairs of letters of $L^{\#}$.*

6. Example of incomplete combining of transitions of automaton $L^{\#}$

Consider Tab. 5 once again. There was already mentioned, that we show in the table only a subset of letters for automaton $L^{\#}$. (The full table contains 80 letters.) It is very important to remark the following thing: although the considered table of # is also such table for the language Waterloo, but it is possible to show, that automaton $L^{\#}$ has *no* walibad property. (However, it certainly has proto-walibad property.)

Thus, let us consider the following Tab. 9, which can be obtained combining Tab. 5 and 7.

Let us consider some comments. We continue to consider automaton $L^{\#}$ for the binary relation # of automaton Waterloo. As we already said, 30 edges of its basis automaton (Tab. 10) corresponds to 15 edges of automaton of Tab. 9, i.e., 15 its letters. In the first line, we show the letter of corresponding edge of automaton Waterloo (by its transition function of Tab. 1). In the second line, we show double subscript of corresponding letter of automaton $L^{\#}$, see some details of such construction in [3]. Initial (A) and final states (E and F) of automaton of this table also corresponds to Initial and final states of the canonical automaton for language Waterloo. In 15 chosen of automaton $L^{\#}$ columns, we painted cells (transitions) corresponding to transitions of the basis automaton. The number of painted cells is less than 30: for example, the left top cell of this table corresponds 2 transitions of the basis automaton, namely

$$A\#Y \overset{a}{\to} E\#X \text{ and } A\#Y \overset{a}{\to} E\#P.$$

Table 9. New combined table

	a	a	a	a	b	a	a	a	a	b	a	a	a	b	a
	$\overset{E}{Y}$	$\overset{F}{Y}$	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{B}{U}$	$\overset{G}{U}$	$\overset{F}{V}$	$\overset{G}{V}$	$\overset{C}{W}$	$\overset{H}{W}$	$\overset{B}{P}$	$\overset{C}{P}$	$\overset{E}{Q}$	$\overset{D}{R}$	$\overset{A}{S}$
$\to A$	E	F									E				
B	E	F				F	G								
C					G	F	G								
D								C	B	C					
$\leftarrow E$								C	B	C					
$\leftarrow F$			A	B					B	C				D	
G			A	B								A		D	A
H			A	B								A			A

Then we make *one of possible* variant of combining some columns (as we said before, we “absorb” unmarked cells by marked ones). Namely, our algorithm *gives the positive result* by the following variant of combining columns (i.e., letters of alphabet of automaton $L^{\#}$):

- columns marked $\overset{B}{Z}$, $\overset{C}{W}$, $\overset{B}{P}$, $\overset{C}{P}$ and $\overset{S}{A}$ are combining together (remark that we have no other painted cells in these columns);
- and each other column is *not* used for combining.

We obtain automaton

Table 10. One of possible variant of combining some columns

	a	b	c	d	e	f	g	h	i	j	k
$\to A$	E	F								E	
B	E	F					F	G			
C					B	G	F	G			
D				C					H		
$\leftarrow E$				C					H		
$\leftarrow F$			A	B							D
G			A	A							D
H			A	A							

(We re-denote 11 letters of used alphabet by $a, b, \dots, k$.) Thus, we consider the last automaton as *the given* one.

We omit the process of its determinization; its obtaining table of relation # (denoting obtained states of canonical automaton for mirror language L^R in the same way, as for on Tab. 4) coincides with the original relation # considered before.

After constructing next objects (which are similar to objects constructed in [5, 6]) we obtain following universal automaton:

Table 11. Universal automaton

$\mathcal{COM}(L)$	a	b	c	d	e	f	g	h	i	j	k
$\to$ (1) $A \times YQ$	6 7 8 14	8 9 10 13 14								6 7 8 14	
$\to$ (2) $AB \times Y$	6 7 8 14	8 9 10 13 14									
(3) $B \times YV$	6 7 8 14	8 9 10 13 14					8 9 10 13 14	10 13			
(4) $BC \times V$							8 9 10 13 14	10 13			
(5) $C \times UV$					2 3 4	10 11 12 13	8 9 10 13 14	10 11 12 13			
(6) $DE \times VP$				4 5					12 13		
$\leftarrow$ (7) $E \times XWP$				4 5					12 13		
$\leftarrow$ (8) $EF \times XP$				4							
$\leftarrow$ (9) $F \times XZPR$			12	2 3 4							
(10) $FG \times ZR$			12	2							6 14
(11) $G \times ZRS$			12	12							6
(12) $GH \times ZS$			12	12							
(13) $FGH \times Z$			12	2							
(14) $DEF \times P$				4							

Let us remark, that the values of its state-marking functions φ^{in} and φ^{out} (which are given in the first column together with numbers used for names of states). As before, we omit the symbols of sets $\{and\}$: for instance, we write $Y\ Q$ and 6 7 8 14 instead of $\{Y, Q\}$ and $\{6, 7, 8, 14\}$ respectively. The equivalence of the last and the given automata could be proved in the usual way.

Using obtained universal automaton, we construct the following covered one, choosing the subset of the set of grids of universal automaton $\{1, 3, 5, 6, 8, 10, 12\}$ (similarly Section 3):

The last automaton is *not* equivalent to the given one; this fact can be shown, for example, also similarly to Section 3: the last automaton has *no* loop corresponding to the loop of basis automaton

$$B\#Y \overset{a}{\rightarrow} F\#P \overset{d}{\rightarrow} B\#V \overset{g}{\rightarrow} F\#Z \overset{c}{\rightarrow} B\#Y \qquad (8)$$

Table 12. Covered automaton

	a	b	c	d	e	f	g	h	i	j	k
$\rightarrow (1)\ A \times YQ$	6 8	8 10								6 8	
$(3)\ B \times YV$	6 8	8 10					8 10	10			
$(5)\ C \times UV$					3	10 12	8 10	10 12			
$(6)\ DE \times WP$				5					12		
$\leftarrow (8)\ EF \times XP$											
$(10)\ FG \times ZR$			1								6
$(12)\ GH \times ZS$			1	1							

Remark that edges of this "missing" loop correspond to edges of the "missing" loop of the covered automaton considered in Section 3. It is also important to remark, that the last automaton does contain loop marked $adjc$; however, this loop does *not* correspond the above loop (8). Anyway, the nonequivalence of the two automata (on Tab. 12 and Tab. 10) can be simple shown by constructing two equivalent canonical automata.

7. Conclusion

As we said before, author is planning to give in the next paper strict formulations of the brute force algorithm of transformation of automaton $L^{\#}$ deleting some letters of its alphabet and combining some other pairs of its letters.

We have realized corresponding computer programs. Using the table of language Waterloo as its only input, we obtain result (i.e., automaton on Tab. 10, which is walibad) in about 1 hour (CPU clock speed was about 3 GHz). Therefore, as we said before, we consider as the main result the possibility of obtaining in near future answer the question, whether or not Waterloo is the "minimal" automaton having walibad property over the alphabet of 2 letters. (The minimality is defined in some natural way, for example, by the number of states of corresponding basis automaton.)

References

[1] Melnikov, B. Some properties of the basis finite automaton / B. Melnikov, A. Melnikova // The Korean Journal of Computational and Applied Mathematics, 2002. – Vol. 9(1). – P. 131-150.

[2] Melnikov, B. A new algorithm of constructing the basis finite automaton / B. Melnikov, A. Melnikova // Informatica (Lietuva), 2002. – Vol.13(3). – P. 299-310.

[3] Melnikov, B. Some more on the basis finite automaton / B. Melnikov, A. Melnikova // Acta Informatica: "Univ. Sapientiae", 2013. – Vol. 5(2). – P. 227-244.

[4] Melnikov, B. Extended nondeterministic finite automata / B. Melnikov // Fundamenta Informaticae, 2010. – Vol. 104(3). – P. 255-265.

[5] Lombardy, S. The universal automaton / S. Lombardy, J. Sakarovitch // Logic and automata: "Amsterdam University Press", 2008. – P. 457-504.

[6] Dolgov, V. Some more algorithms for Convway's universal automaton / V. Dolgov, B. Melnikov // Acta Informatica: "Univ. Sapientiae", 2014. – Vol. 6(1). – P. 5-20.

[7] Melnikov, B. Once more about the state-minimization of the nondeterministic finite automata / B. Melnikov // The Korean Journal of Computational and Applied Mathematics, 2000. – Vol. 7(3). – P. 655-662.

[8] Polák, L. Minimalizations of NFA using the universal automaton / L. Polák // Int. J. Found. Comput. Sci., 2005. – Vol. 16(5). – P.999-1010.

[9] Geldenhuys, J. Reducing nondeterministic finite automata with SAT solvers Springer. / J. Geldenhuys, B. van der Merwe, L. van der Zijl // Finite-State Methods and Natural Language Processing. Lecture Notes in Computer Science, 2010. – Vol. 6062. – P. 81-92.

[10] Melnikov, B. Once more on the edge-minimization of nondeterministic finite automata and the connected problems / B. Melnikov // Fundamenta Informaticae, 2010. – Vol. 104(3). – P. 267-283.

[11] Yo-Sub, Han. State elimination heuristics for short regular expressions / Han. Yo-Sub // Fundamenta Informaticae, 2013. – Vol. 128. – P. 445-462

[12] Kameda, T. On the state minimization of nondeterministic finite automata / T. Kamada, P. Wiener // IEEE Trans. on Comp., 1970. – Vol. C-19(7). – P. 617-627.

[13] Melnikov, B. The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method Proceedings / B. Melnikov, A. Tsyganov // International Symposium on Parallel Architectures, Algorithms and Programming, PAAP, 2012. – P.194-201.

[14] Jiang, T. Minimal NFA problems are hard / T. Jiang, B. Ravikumar // SIAM J. Comput., 1993. – Vol. 22(6). – P. 1117-1141.

[15] Melnikov, B. Discrete optimization problems – some new heuristic approaches / B. Melnikov // 8th Int. Conf. on High Performance Computing and Grid IEEE Comp.: "Soc. Press Ed.", 2005. – P. 73-80.

[16] Melnikov, B. Some special heuristics for discrete optimization problems / B. Melnikov, A. Radionov, V. Gumayunov // 8th Int. Conf. on Enterprise of Information Systems Pathos (Cyprus), 2003. – P. 91-95.