Waterloo-like finite automata and algorithms for their automatic construction

B. F. Melnikov, E. A. Melnikova

***Abstract*—In the problems of minimization of nondeterministic finite automata, there may be situations, when a covering set of blocks defines an automaton, which is not equivalent to the original one. For the first time, such an example was obtained in 1970 by Kameda and Weiner, and according to their paper, was given the name Waterloo. The existence of such constructions ("walibad" in our terminology, from "Waterloo-like badness") greatly complicates the description of practical algorithms for the state minimization of nondeterministic finite automata. Hence the problems of the search and description of such constructions arouse, and, where possible, it should be done before applying the said algorithms of minimization.**

In this paper, we propose an example of algorithm 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. We emphasize, that in the process of building we used only relation $\#$, and did not use automaton known in advance.

***Keywords*—nondeterministic finite automata, universal automaton, covering set of blocks, covering automaton, Waterloo automaton, basis automaton, complete automaton.**

I. Introduction and motivation

This paper can be considered a continuation of [1]: in that paper, the *complete* finite automata were considered, and here we continue to consider the connection of complete automata with the problems of state minimization for nondeterministic finite automata, see [2]–[9] etc. Note in advance that the whole direction consists not only of the theoretical part (including the description of new variants of minimization algorithms, the proofs of their correctness, etc.), but also of the practical part. This practical part includes the development and implementation of heuristics for minimizing automata with a large number of states, and we already considered (e.g., in [9]) some *algorithmic* questions related to the implementation of the corresponding heuristic algorithms. The automaton obtained as a result of vertex minimization [1] can be considered an analogue of the so-called minimal code $\mu_{min}(G)$ of adjacency matrix of a graph G considered as a complete invariant for the graph, see [10] etc. That is, when we consider such a finite automaton, we obtain one more complete invariant, in addition to the usual canonical automaton, as well as the basic and universal automata considered in our previous papers, see [11]–[15] etc. [2]

In the development of practical algorithms for minimizing nondeterministic finite automata, one must take into account the possible situation when the covering set of grids (blocks) [3] defines an automaton (the so-called covering automaton), *not* being equivalent to the given one. For the first time, such an example was obtained by Kameda and Weiner, and according to their paper [17] was called *Waterloo*. Apparently, after that publication (1970), there were *no more* published similar examples in the literature, i.e., finite automata (regular languages) possessing such properties. In our terminology, the description of this example was briefly outlined in [7]; below we give a more detailed description. We call this construction a *walibad* (from the expression "Waterloo-like badness"), and any covering subset of the grid set that does *not* include *all* the grids is called *proto-walibad*. The presence of walibads greatly complicates the description of practical algorithms for the state minimization of nondeterministic finite automata, and, therefore, there are problems of searching and describing such constructions, and, if possible, *before* applying the minimization algorithms.

This paper first shows, *how to build a walibad construction automatically*, based on:

- a specific concrete example of walibad, and, therefore, corresponding table of binary relation $\#$ (see [11], [13], [15] etc.);
- automaton $K^{\#}$, defined by us on the basis of such a binary relation (such a table) in [1].

This allows to make *a descriptions based on one table of the $\#$-relation only* (the relation has to have the above property proto-walibad):

1) an exhaustive algorithm for constructing walibad;
2) an exhaustive algorithm for checking the necessary condition for such a construction for an *arbitrary* nondeterministic automaton;
3) a classification of all possible tables of the $\#$-relation (possessing this property) by the condition of the existence of the walibad-construction corresponding to this table.

Received October 22, 2017.

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

Elena A. Melnikova, Russian State Social University (email: ya.e.melnikova@yandex.ru).

[1] In the presence of some simple formulated criteria ordering such automata. Such criteria can be easily formulated in some different ways, in this paper, we will not focus on this problem.

[2] Note that according to the facts shown in [1], the complete finite automaton is also an invariant of the regular language, however, unlike the other examples mentioned here, it is not a complete invariant of it. The title "complete" (for the automaton under consideration) is related not to the completeness of the invariant, but to the completeness of the set of possible edges, see, e.g., [16].

[3] Hereinafter, the terminology associated with the grids is given according to [3]. In this case, the covering automata are constructed according to [13, Sect. 4].

This paper has the following structure. In this section, we briefly describe the used notation. In Sections III and IV, we consider the Waterloo automaton in detail, and from the point of view of the theory described in our previous works cited above. In Section V we define the concepts of proto-walibad, i.e., covering set of grids that do not coincide with the whole set of grids, and walibad, i.e. a proto-walibad, for which there exists a covering automaton that is not equivalent to the original automaton.

In Section VI, we show how one can obtain some given automaton having the walibad property (in our case, it is the Waterloo-automaton), using an exhaustive algorithm based on the complete automaton $K^{\#}$ constructed for the corresponding regular language. In addition, in this section we formulate an assertion relating to all regular languages (and in fact represents one of the possible variants of their classification): for each regular language, one can determine whether the walibad property has at least some language with a binary relation $\#$ coincides with this relation for the given language.

And in Section VII we show, how we can get *an automaton that has a nonequivalent covering automaton* using an algorithm of special *"incomplete"* combining the letters of automaton $K^{\#}$. Apparently, the result of this section is the most important *for developing practical minimization algorithms*: in other words, in Section VII we show that *in practice* the desired result (i.e., the presence of walibad) can often be obtained *much faster*, i.e., long before the end of the work of some exhaustive algorithm.

In Conclusion, we formulate some possible directions for further work.

II. Preliminaries

The notations used in our paper will be described very briefly: almost all the necessary notations and the results of our previous work needed for this paper were considered detailed in [1], and before in [11], [13], [15]. We shall consider nondeterministic finite automaton

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

without ε-edges, i.e., we shall consider the transition function δ of automaton (1) as

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

Some edge $\delta(q, a) \ni r$ will usually by written as

$$q \xrightarrow[\delta]{a} r,$$

or, if it will not cause misunderstandings, simply as $q \xrightarrow{a} r$. The language defined by automaton K will be defined as $\mathcal{L}(K)$.

The canonical automaton for some regular language L will be defined as $\widetilde{L}$, besides canonical automata for L and its mirror image L^R will be the following ones:

$$\widetilde{L} = (Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi)$$

and

$$\widetilde{L^R} = (Q_\rho, \Sigma, \delta_\rho, \{s_\rho\}, F_\rho).$$

Binary relation $\# \subseteq Q_\pi \times Q_\rho$ is defined for the pairs of states of automata $\widetilde{L}$ and $\widetilde{L^R}$ in the following way: $A \# X$ if and only if

$$\big(\exists uv \in L\big) \big(u \in \mathcal{L}^{in}_{\widetilde{L}}(A) , v^R \in \mathcal{L}^{in}_{\widetilde{L^R}}(X)\big) .$$

The state-marking function $\varphi^{in}_K : Q \to \mathcal{P}(Q_\pi)$ is defined in the following way:

$$\varphi^{in}_K(q) \ni \widetilde{q} \quad \text{if and only if} \quad \mathcal{L}^{in}_K(q) \cap \mathcal{L}^{in}_{\widetilde{L}}(\widetilde{q}) \neq \varnothing .$$

And the state-marking function

$$\varphi^{out}_K : Q \to \mathcal{P}(Q_\rho)$$

is defined similarly for automata K^R (the mirror automaton for K) and $\widetilde{L^R}$.

Here, we will not determine in detail the basis automaton $\mathcal{BA}(L)$ and the universal automaton $\mathcal{COM}(L)$, see [1], [11]. We note only the necessary condition of the existence of transition $\mathcal{B}_1 \xrightarrow[\delta_Q]{a} \mathcal{B}_2$ of automaton $\mathcal{COM}(L)$ (i.e., the transition from the state $\mathcal{B}_1$ in the state $\mathcal{B}_2$ of automaton $\mathcal{COM}(L)$ labeled by the letter $a \in \Sigma$; see also [16, Def. 2]):

$$\big(\forall p \in \alpha(\mathcal{B}_1)\big) \big(\delta_\pi(p, a) \in \alpha(\mathcal{B}_2)\big) \tag{2}$$

$$\& \quad \big(\forall r \in \beta(\mathcal{B}_2)\big) \big(\delta_\rho(r, a) \in \beta(\mathcal{B}_1)\big). \tag{3}$$

(Based on these conditions, we construct all the transitions of automaton $\mathcal{COM}(L)$. The conditions for input and output states are less important for this paper; if necessary, see them in our papers cited before.)

III. Consideration of automaton Waterloo according to our terminology

In this section we will consider a detailed example, i.e., Waterloo automaton. We can consider this example to be a continuation of the series of examples discussed in [11], but, more importantly, this example also demonstrates the possibility of the following situation.

There exists the language and a covered automaton for it[4]*, such that this automaton does not define the given language.*

Thus, let us consider automaton Waterloo, see Fig. 1.

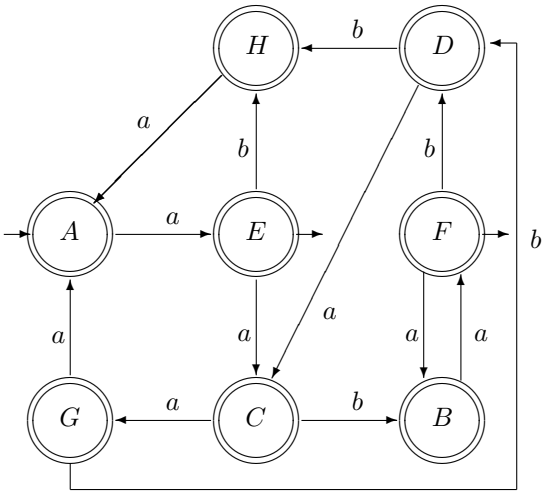

Fig. 1

This automaton is deterministic. Moreover, it does not contain equivalent states; then for its language, it is the *canonical* automaton; then *we can* denote this automaton by $\widetilde{L}$. It is convenient to consider it in the following Tab. 1. Let

[4] I.e., automaton which state-marking functions φ^{in} and φ^{out} covered all the possible pairs of corresponding states of two canonical automata $\widetilde{L}$ and $\widetilde{L^R}$. See below for details.

us remark that since its determinism, we can consider each cell of this table by both state and also by the *set* of states; such sets can contain more than 1 element.

Tab. 1

	$\widetilde{L}$	a	b
$\to$	A	E	–
	B	F	–
	C	G	B
	D	C	H
$\leftarrow$	E	C	H
$\leftarrow$	F	B	D
	G	A	D
	H	A	–

Tab. 2

	$(\widetilde{L})^R$	a	b
$\leftarrow$	A	G, H	–
	B	F	C
	C	D, E	–
	D	–	F, G
$\to$	E	A	–
$\to$	F	B	–
	G	C	–
	H	–	D, E

For obtaining automaton $\widetilde{L^R}$, it is convenient to use the procedure of determinization of automaton $(\widetilde{L})^R$. Automaton $(\widetilde{L})^R$ is shown on Table 2; it is not a deterministic one, then we have to consider each cell of this table be *the set* of states.

Let us determinate automaton $(\widetilde{L})^R$. We begin to consider the only its input state; it is convenient to denote it by E, F (similarly we will do below). This input state corresponds to the set $\{E, F\}$, and, therefore, can be obtained by combining states E and F of automaton $(\widetilde{L})^R$. Then the set (or the state) A, B appears, therefore we include the corresponding string in the built automaton; etc. The final states are ones "containing" the only final state A of automaton $(\widetilde{L})^R$;[5] they are $\{A\}$ and $\{A, B\}$, and we will denote them A and A, B respectively.

Tab. 3

		a	b
$\to$	E, F	A, B	–
$\leftarrow$	A, B	F, G, H	C
	F, G, H	B, C	D, E
	C	D, E	–
	B, C	D, E, F	C
	D, E	A	F, G
	D, E, F	A, B	F, G
$\leftarrow$	A	G, H	–
	F, G	B, C	–
	G, H	C	D, E

Tab. 4

	$\widetilde{L^R}$	a	b
$\to$	X	Y	–
$\leftarrow$	Y	Z	U
	Z	V	W
	U	W	–
	V	P	U
	W	Q	R
	P	Y	R
$\leftarrow$	Q	S	–
	R	V	–
	S	U	W

Thus, the complete Table 3 of corresponding deterministic automaton consists of 10 states, including the "old" states A and C (for convenience, we have regularized states in order of their "appearance"). Making the usual renaming states, we obtain Table 4. Evidently, the last automaton does not contain equivalent states, then we can consider it as the canonical automaton and denote it by $\widetilde{L^R}$. And using the renaming states we also obtain the following Table 5 for binary relation $\#$[6]:

Tab. 5

	X	Y	Z	U	V	W	P	Q	R	S
A		#						#		
B		#			#					
C				#	#					
D						#	#			
E	#					#	#			
F	#		#				#		#	
G			#						#	#
H			#							#

[5] I.e., when we consider them as *the sets* of states of the deterministic automaton. Note again, that we have in mind both the combining of the designations of different states, and (more importantly) the combining of each pair of cells of the corresponding string in the table.

[6] We can obtain it in the following way. For example, in automaton of *both* Tables 3 and 4, we renamed state $\{E, F\}$ by X; therefore we obtain $E\#X$ and $F\#X$. The same procedure has to be used for all the states of the new automaton.

Using this table, we obtain the following 14 grids[7]:

(g1)	$\{A\} \times \{Y, Q\}$	(g2)	$\{A, B\} \times \{Y\}$
(g3)	$\{B\} \times \{Y, V\}$	(g4)	$\{B, C\} \times \{V\}$
(g5)	$\{C\} \times \{U, V\}$	(g6)	$\{D, E\} \times \{W, P\}$
(g7)	$\{E\} \times \{X, W, P\}$	(g8)	$\{E, F\} \times \{X, P\}$
(g9)	$\{F\} \times \{X, Z, P, R\}$	(g10)	$\{F, G\} \times \{Z, R\}$
(g11)	$\{G\} \times \{Z, R, S\}$	(g12)	$\{G, H\} \times \{Z, S\}$
(g13)	$\{F, G, H\} \times \{Z\}$	(g14)	$\{D, E, F\} \times \{P\}$

Below, we shall use these numbers of grids (states), sometimes without letter "g" and sometimes without round brackets; like [1], this will not cause discrepancies, from the context it will always be possible to distinguish the grid from the formula. We shall use them for both automaton $\mathcal{COM}(L)$ and the covering automaton (below, we shall consider the only such one).

Below, we shall use these numbers of grids (states), sometimes without letter "g" and sometimes without round brackets; like [1], this will not cause discrepancies, from the context it will always be possible to distinguish the grid from the formula. We use them for both automaton $\mathcal{COM}(L)$ and the covering automaton (we shall consider the only such one).

Thus, let us construct automaton $\mathcal{COM}(L)$ using obtained set of grids. For this constructing, let us make Tables 6 and 7 for both the letters of considering alphabet. In both following tables:

- strings (see column "q") are marked by the states of automaton $\mathcal{COM}(L)$;
- "δ_Q" (values of transition functions) is obtained using combining values $\delta_Q(q, a)$ (or $\delta_Q(q, b)$ for the second table) for the set of states marking the considered string;
- "α-candidates" are the states, which satisfy condition (2);[8] here, grid $\mathcal{B}_1$ is the label of corresponding string, and $\mathcal{B}_2$ is the label of corresponding column; for instance, we write 10 in the string labeled by 3 (on Table 6), because grid 3 (i.e., $\mathcal{B}_1 = \{B\} \times \{Y, V\}$) and grid 10 (i.e., $\mathcal{B}_2 = \{F, G\} \times \{Z, R\}$) satisfy condition (2);
- "transitions" are results, i.e., in the considered column of automaton $\mathcal{COM}(L)$, we write all the states satisfying both the conditions (2),(3).

Tab. 6 (for a)

q	$\alpha(q)$	$\beta(q)$	δ_Q	α-candidates	transitions
$\to 1$	A	Y, Q	E	6, 7, 8, 14	6, 7, 8, 14
$\to 2$	A, B	Y	E, F	8, 14	8, 14
3	B	Y, V	F	8,9,10,13,14	8,9,10,13,14
4	B, C	V	F, G	10, 13	10, 13
5	C	U, V	G	10,11,12,13	10,11,12,13
6	D, E	W, P	C	4, 5	4, 5
$\leftarrow 7$	E	X, W, P	C	4, 5	4, 5
$\leftarrow 8$	E, F	X, P	B, C	4	4
$\leftarrow 9$	F	X,Z,P,R	B	2, 3, 4	2, 3, 4
10	F, G	Z, R	A, B	2	2
11	G	Z, R, S	A	1, 2	1, 2
12	G, H	Z, S	A	1, 2	1, 2
13	F,G,H	Z	A, B	2	2
14	D,E,F	P	B, C	4	4

Let us remark, that two last columns on Table 6 coincide; however, for Table 7 (we construct it similarly for letter b) such a fact is false:

[7] Compare corresponding constructions in [1].

[8] Maybe, not both (2) and (3) simultaneously.

Tab. 7 (for b)

q	$\alpha(q)$	$\beta(q)$	δ_Q	α-candidates	transitions
$\rightarrow 1$	A	Y, Q	–	–	–
$\rightarrow 2$	A, B	Y	–	–	–
3	B	Y, V	–	–	–
4	B, C	V	B	$2, 3, 4$	–
5	C	U, V	B	$2, 3, 4$	$2, 3, 4$
6	D, E	W, P	H	$12, 13$	$12, 13$
$\leftarrow 7$	E	X, W, P	H	$12, 13$	$12, 13$
$\leftarrow 8$	E, F	X, P	D, H	–	–
$\leftarrow 9$	F	X,Z,P,R	D	$6, 14$	$6, 14$
10	F, G	Z, R	D	$6, 14$	$6, 14$
11	G	Z, R, S	D	$6, 14$	$6, 14$
12	G, H	Z, S	D	$6, 14$	–
13	F,G,H	Z	D	$6, 14$	–
14	D,E,F	P	D, H	–	–

Combining two last tables, we obtain automaton $\mathcal{COM}(L)$, which is given on Table 8:

Tab. 8

$\mathcal{COM}(L)$		a	b
$\rightarrow$	1	$6, 7, 8, 14$	–
$\rightarrow$	2	$8, 14$	–
	3	$8, 9, 10, 13, 14$	–
	4	$10, 13$	–
	5	$10, 11, 12, 13$	$2, 3, 4$
	6	$4, 5$	$12, 13$
$\leftarrow$	7	$4, 5$	$12, 13$
$\leftarrow$	8	4	–
$\leftarrow$	9	$2, 3, 4$	$6, 14$
	10	2	$6, 14$
	11	$1, 2$	$6, 14$
	12	$1, 2$	–
	13	2	–
	14	4	–

And Table 9 demonstrates the process of determinization of automaton $\mathcal{COM}(L)$; it can be also used for obtaining functions φ^{in} and φ^{out} of the considered language.

Tab. 9

		a	b
$\rightarrow$	$1, 2$	$6, 7, 8, 14$	–
$\leftarrow$	$6, 7, 8, 14$	$4, 5$	$12, 13$
	$4, 5$	$10, 11, 12, 13$	$2, 3, 4$
	$12, 13$	$1, 2$	–
	$10, 11, 12, 13$	$1, 2$	$6, 14$
	$2, 3, 4$	$8, 9, 10, 13, 14$	–
	$6, 14$	$4, 5$	$12, 13$
$\leftarrow$	$8, 9, 10, 13, 14$	$2, 3, 4$	$6, 14$

The last automaton is equivalent to the given one; we can obtain the given automaton (Fig. 1), renaming states [9] in the following way:

$$\{1, 2\} = A, \quad \{2, 3, 4\} = B,$$
$$\{4, 5\} = C, \quad \{6, 14\} = D,$$
$$\{6, 7, 8, 14\} = E, \ \{8, 9, 10, 13, 14\} = F,$$
$$\{10, 11, 12, 13\} = G, \ \{12, 13\} = H.$$

Below, we shall also use the basis automaton for this language; we consider only the table of its transition function, without detailed algorithm of its construction. Thus, $\mathcal{BA}(L)$ can be described in the following way, see Tab. 10. [10]

[9] Or *sets* of states, if we consider cells of Table 9 as the sets.

[10] Similarly to [1], we denote 20 its states like $A\#X$ etc., instead of the usual notation $\begin{smallmatrix} A \\ X \end{smallmatrix}$.

Tab. 10

$\mathcal{BA}(L)$		a	b
$\rightarrow$	$A\#Y$	$E\#X, E\#P$	–
$\rightarrow$	$A\#Q$	$E\#W$	–
	$B\#Y$	$F\#X, F\#P$	–
	$B\#V$	$F\#Z, F\#R$	–
	$C\#U$	$G\#S$	$B\#Y, B\#V$
	$C\#V$	$G\#Z, G\#R$	–
	$D\#W$	$C\#U$	$H\#Z, H\#S$
	$D\#P$	$C\#V$	–
$\leftarrow$	$E\#X$	–	–
	$E\#W$	$C\#U$	$H\#Z, H\#S$
	$E\#P$	$C\#V$	–
$\leftarrow$	$F\#X$	–	–
	$F\#Z$	$B\#Y$	–
	$F\#P$	$B\#V$	–
	$F\#R$	–	$D\#W, D\#P$
	$G\#Z$	$A\#Y$	–
	$G\#R$	–	$D\#W, D\#P$
	$G\#S$	$A\#Q$	–
	$H\#Z$	$A\#Y$	–
	$H\#S$	$A\#Q$	–

Remark that we obtain the loop

$$B\#Y \xrightarrow{a} F\#P \xrightarrow{a} B\#V \xrightarrow{a} F\#Z \xrightarrow{a} B\#Y\,; \quad (4)$$

we shall use this fact below.

IV. Automaton Waterloo and an nonequivalent covering automaton

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 $\#$; [11]
- there exist all the possible transitions; in other words, we choose all the transitions of automaton $\mathcal{COM}(L)$ (i.e., the transitions satisfying both the conditions (2),(3)); [12]
- however, its language is not equal to L (i.e., language of the given automaton).

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

$$\mathrm{g1},\ \mathrm{g3},\ \mathrm{g5},\ \mathrm{g6},\ \mathrm{g8},\ \mathrm{g10},\ \mathrm{g12},$$

or, simply,

$$1,\ 3,\ 5,\ 6,\ 8,\ 10,\ 12; \quad (5)$$

evidently, that all the 20 elements of relation $\#$ (Tab. 5) are covered by these states. [13]

Constructing all the possible transitions (i.e., transitions satisfying (2),(3)), we obtain automaton of Table 11. [14]

[11] Let us consider this thing detailed. State 10 of automaton of Tables 6 and 7 has the following values of marking function: $\varphi^{in}_K(10) = \{F, G\}$ and $\varphi^{out}_K(10) = \{Z, R\}$. Then *this* state covers the following elements of relation $\#$: $F\#Z$, $F\#R$, $G\#Z$ and $G\#R$. There exist some *possible* sets of such states, where corresponding pairs (like $F\#Z$) cover all the elements of considered relation $\#$.

[12] Let us remark, that using [1], we can define *some such* automaton by its state-marking functions or by corresponding relation $\#$. In other words, using the theory given in [16], we can define *all possible* transitions for a finite automaton defining a given regular language, knowing the state-markings function of of its states.

[13] Remark that the *algorithms* of constructing such a set of grids is a hard problem, see [2] etc. Some possible approaches for making heuristic algorithms for this problem were considered in the papers cited in Introduction, including some our papers; see also [18], [19].

[14] By [1], [15], another way of constructing such covering automaton is the sequent elimination of states of automaton $\mathcal{COM}(L)$. More detailed, we remove states which do not belong to set (5), and also corresponding edges.

The determinization of this automaton gives automaton of Table 12.

Tab. 11

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

Tab. 12

		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

After that, we rename states in the following "natural" way:

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

This renaming gives automaton of Table 13:

Tab. 13

		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	–

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 (Tab. 1).

V. Proto-walibads and walibads

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

Definition 1: Any covering subset of a set of grids that includes not all grids (and also the corresponding covering automaton) will be called having the property of proto-walibad (or simply *proto-walibad*). □

Example 1: Consider the following table of relation $\#$:

Tab. 14

	X	Y	Z
A	$\#$	$\#$	
B	$\#$	$\#$	$\#$
C		$\#$	$\#$

For this relation, we have the following grids:

(g1) $\{A,B\} \times \{X,Y\}$ (g2) $\{B,C\} \times \{Y,Z\}$
(g3) $\{B\} \times \{X,Y,Z\}$ (g4) $\{A,B,C\} \times \{Y\}$

A proto-walibad is the set consisting of grids (g1) and (g2). [15] □

Definition 2: If for the given regular language (or for the given finite automaton) there exists a proto-walibad, for which the corresponding covering automaton is not equivalent to the given automaton, then the given language (the given automaton) is called walibad. [16] □

Remark that automaton Waterloo satisfies this definition (see Sections III and IV). However we can show, that considering language of Example 1, we can show that there exists *no* walibads having the table of relation $\#$ (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 shall not consider this example in more details. And, of course, there are regular languages that do not have the property proto-walibad; the simplest example is the language which can be defined by regular expression

$$(a + ab + ba)^*.$$

VI. Construction of walibads using relation $\#$

In this section we shall show, how we can get a walibad (in particular, the Waterloo automaton) in an *exhaustive* way based on the complete automaton $K^{\#}$ and the theory given in [1] (that is, actually, based on the binary relation $\#$ only). It is important to note that in this section, the constructions are carried out for *some already known* walibad.

Remark that the language of its automaton $L^{\#}$ includes 80 letters (i.e., the table of transition of automaton $K^{\#}$ for language Waterloo includes 80 columns), this is a lot for detailed consideration. Therefore, unlike [1], we do not give the entire table in this paper, we shall consider *the used* letters only. Moreover, unlike [1], the edges and letters of the alphabet $\Sigma^{\#}$, corresponding to the different letters of the source language (i.e., to a and b), we shall give more details in different tables.

Thus, consider automaton Waterloo once again. Let us write out from the table of the automaton $K^{\#}$ all the columns with marks corresponding to the letter a (they are painted in Tab. 15, 15′, like [1]):

Tab. 15

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

Tab. 15′

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

[15] Remark once again, that by [1], we can construct finite automaton corresponding the given table of relation $\#$. It can be also shown, that if automaton $Ł^{\#}$ is chosen as such automaton, then, according to Definition 1, it can also be called a proto-walibad.

[16] Remind that covering automata are constructed according to [13, Sect. 4].

of the obtained table by the brute force algorithm while the process of obtaining a walibad, "absorbing" unmarked cells by marked ones; see Tab. 16 for letter a:

Tab. 16

	$K^{\#}$	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. 17 and Tab. 18 below). Thus, combining tables for letters a and b, we obtain transition function for (canonical) automaton Waterloo (compare Tab. 1).

Tab. 17

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

Tab. 18

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

Evidently, the same automaton can be obtained *by the brute force algorithm*, even *not knowing corresponding basis automaton*. To do this, we "only" need:

- to sort through all the possible sets of markers of cells (i.e., edges) of automaton $K^{\#}$;
- for each of these sets, to construct the combined columns of the transition matrix;
- to obtain the original automaton Waterloo after such combining.

Thus, using the example of the automaton Waterloo, we demonstrated the possibility of a simple proof of the following statement.

Proposition 1: If for some given table of a binary relation $\#$, there exists an automaton for which the some its covering automaton is not equivalent to the given one, then this given automaton can be obtained in an exhaustive way on the basis of automaton $K^{\#}$ and some transformation of this automaton, performed on the basis of the sequence of transformations described in [1, Prop. 4]:

1) building "starting" automaton $K^{\#}$;
2) deleting some its edges;
3) renaming some edges (i.e., changing their marking letters) – perhaps, marking some edges by *different* new letters;
4) renaming marks of some edges by possible marking some *existing different* edges by one new letter. □

A strict proof is constructed in an obvious way similarly to the process described in this section, with the replacement of a specific automaton (language) by an arbitrary one. [17] It can also be said that this statement follows from the material of [1].

[17] Of course, when speaking of a specific automaton (regular language) we mean both Waterloo and some other automaton (language) that has the same properties. However, other examples (except Waterloo) are not known to us (except for the language we obtained below on the basis of the same automaton Waterloo).

And the following statement (the corollary of the previous one) can be considered as belonging to *all the regular languages*: for each of them, we can determine whether *at least one regular language* possessing the given binary relation $\#$ has the walibad property.

Proposition 2: For some given binary relation $\#$, which has the proto-walibad property, there is an exhaustive algorithm, described on the basis of combining the arcs of automaton $K^{\#}$, determining whether there is a walibad having the given binary relation. □

VII. An example of incomplete combining transitions of a complete automaton

In this section, we show by example that to answer the question whether some proto-walibad is walibad, we can not even combine the states of a complete automaton before [18] the given automaton is received, i.e., before we obtain the automaton, about whom *it is known in advance* that it is a walibad. We already noted in Introduction that we consider the result of this section to be the most important *for the development of practical minimization algorithms*, because it is very useful in the process of vertex minimization of the automaton to know *in advance* whether the minimized language (automaton) is walibad. Moreover, in the example we will consider the same language: Waterloo.

Consider Tab. 5 once again. There was already mentioned, that we show in the table only a subset of letters for automaton $K^{\#}$. (The full table consists of 80 letters.) It is very important to remark the following thing: although the considered table of $\#$ is also such a table for the language Waterloo, but it is possible to show, *that automaton $K^{\#}$ has no walibad property*. [19] (However, it certainly has the proto-walibad property.)

Thus, let us consider the following Tab. 19, 19′, which can be obtained combining Tab. 15, 15′ and 17:

Tab. 19

	a	a	a	a	b	a	a	a
	$\overset{E}{Y}$	$\overset{F}{Y}$	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{B}{U}$	$\overset{G}{U}$	$\overset{F}{V}$	$\overset{G}{V}$
$\rightarrow A$	E	F						
B	E	F					F	G
C					B	G	F	G
D								
$\leftarrow E$								
$\leftarrow F$			A	B				
G			A	B				
H			A	B				

Tab. 19′

	a	b	a	a	a	b	a
	$\overset{C}{W}$	$\overset{H}{W}$	$\overset{B}{P}$	$\overset{C}{P}$	$\overset{E}{Q}$	$\overset{D}{R}$	$\overset{A}{S}$
$\rightarrow A$					E		
B							
C							
D	C	H	B	C			
$\leftarrow E$	C	H	B	C			
$\leftarrow F$			B	C		D	
G						D	A
H							A

Thus, in Table 19, 19′, we have given only the letters of the automaton $K^{\#}$ constructed for the Waterloo language, which are necessary for the following construction. We gave these

[18] In practice: *long before*.

[19] The detailed construction, proving this fact, is very cumbersome, and we omit it.

letters (according to definition of [1, Sect.V], these letters, but not only they, belong to alphabet $\Sigma_{\#}$) in the convenient order, i.e., in the order given of [1, Sect.V].

Let us consider some more comments to Table 19, 19′. As we already said, we continue to consider automaton $K^{\#}$ for binary relation $\#$ of automaton Waterloo, and 30 edges of its basis automaton (Tab.,10) corresponds to 15 edges of automaton (columns of of Tab. 19, 19′), 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 $K^{\#}$, see some details of such construction in [1, Sect.V]. Initial state (A) and final states (E and F) of automaton of this table also correspond to initial and final states of the canonical automaton for language Waterloo. In 15 chosen columns of automaton $K^{\#}$, 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

$$A\#Y \xrightarrow{a} E\#X \quad \text{and} \quad A\#Y \xrightarrow{a} E\#P.$$

Now, we shall make *one of possible* variant of combining some columns (as we said in [1, Sect.VII], we use a special version of the combination of letters (i.e., 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 $K^{\#}$):

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

In such a way, we obtain the following automaton:

Tab. 20

	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-denoted 11 letters of used alphabet by a, b, …, k.) Thus, since now, *we shall 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 we did for Tab. 4) coincides with the original relation $\#$ considered before.

After constructing next objects (which are similar to objects constructed in Section III; see also [13], [20]) we obtain the following universal automaton $\mathcal{COM}(L)$, see Tab. 21, 21′ below. [20] 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) also coincides with numbers used in Section III. As before, we omit the symbols of sets (the braces): for instance, we write YQ 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.

[20] To reduce the size of the table, we did not specify corresponding pairs of states of automata $\widetilde{L}$ and $\widetilde{L^R}$ in it. We indicated this correspondence above, and also shall give it below once again.

Tab. 21

$COM(L)$	a	b	c	d	e	f
$\to$ (1)	6 7 8 14	8 9 10 13 14				
$\to$ (2)	6 7 8 14	8 9 10 13 14				
(3)	6 7 8 14	8 9 10 13 14				
(4)						
(5)					2 3 4	10 11 12 13
(6)				4 5		
$\leftarrow$ (7)				4 5		
$\leftarrow$ (8)				4		
$\leftarrow$ (9)			1 2	2 3 4		
(10)			1 2	2		
(11)			1 2	1 2		
(12)			1 2	1 2		
(13)			1 2	2		
(14)				4		

Tab. 21′

$COM(L)$	g	h	i	j	k
$\to$ (1)				6 7 8 14	
$\to$ (2)					
(3)	8 9 10 13 14	10 13			
(4)	8 9 10 13 14	10 13			
(5)	8 9 10 13 14	10 11 12 13			
(6)			12 13		
$\leftarrow$ (7)			12 13		
$\leftarrow$ (8)					
$\leftarrow$ (9)					
(10)					6 14
(11)					6
(12)					
(13)					
(14)					

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

Tab. 22

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

Tab. 22′

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

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

$$B\#Y \xrightarrow{a} F\#P \xrightarrow{d} B\#V \xrightarrow{g} F\#Z \xrightarrow{c} B\#Y \quad (6)$$

(compare (4)). Remark that edges of this "missing" loop correspond to edges of the "missing" loop of the covered automaton considered in Section IV. It is also important to remark, that the last automaton does contain loop marked $adgc$; however, this loop does not correspond the above loop (6). Anyway, the *nonequivalence* of the two these automata (on Tab. 22, 22′ and Tab. 20) can be simple shown by constructing two equivalent canonical automata.

VIII. Conclusion

Thus, in the present paper we have considered algorithms for *automatic* construction of automata, where the covering finite automaton turns out to be nonequivalent to the original one. The continuation of this work is supposed in two directions:

1) to build *a special classification of regular languages* based on the existence / non-existence of similar structures;
2) to describe in detail the application for auxiliary algorithms recognizing the existence of these constructions in algorithms for minimization of nondeterministic finite automata. [21]

We have already realized some corresponding computer programs, [9]. Using described in this paper auxiliary heuristic algorithms and having the table of language Waterloo as *the only input*, we obtained the desired result (i.e., automaton on Tab. 22, 22′, which is walibad) in about 200 hours of computer time (CPU clock speed was about 3 GHz). Therefore, as we said before, we consider as the main result the possibility of obtaining in the near future answer of the question, whether or not Waterloo is the "minimal" automaton having walibad property over the alphabet of 2 letters. (The minimality can be defined in some natural ways, for example, by the number of states of corresponding basis automaton, of by product of number of states of automata $\widetilde{L}$ and $\widetilde{L^R}$.)

Also, further practical work can be carried out in the directions of our papers cited above and [21], [22].

[21] Above we have already noted that to create effective computer programs for NFA-minimization, it is desirable to be able to determine *in advance* (i.e., long before the full processing of the given regular language) the presence of walibad. Here it is possible to consider an analogy with the branch-and-bound method, including incomplete one: for the majority of problems solved by it, it, generally speaking, does not guarantee obtaining the optimal solution in an acceptable time, but almost always gives very good results.

References

[1] Melnikov B. *The complete finite automaton*. International Journal of Open Information Technologies. 2017, vol. 5, no. 10, pp. 9–17.

[2] Jiang T. and Ravikumar B. *Minimal NFA problems are hard*. SIAM J. Comput. 1993, vol. 22, no. 6, pp. 1117–1141.

[3] Melnikov B. *Once more about the state-minimization of the nondeterministic finite automata*. The Korean Journal of Computational and Applied Mathematics. 2000, vol. 7, no. 3, pp. 655–662.

[4] Polák L. *Minimalizations of NFA using the universal automaton*. International Journal of Foundations of Computer Science. 2005, vol. 16, no. 5, pp. 999–1010.

[5] Melnikov B., Radionov A., Moseev A. and Melnikova E. *Some specific heuristics for situation clustering problems*. Proceedings of the 1st International Conference on Software and Data Technologies, ICSOFT-2006. Setubal, Portugal. 2006, pp. 272–279.

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

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

[8] Yo-Sub Han. *State elimination heuristics for short regular expressions*. Fundamenta Informaticae. 2013, vol. 128, pp. 445–462.

[9] Krivolapova A., Melnikova E. and Sofonova N. *Nekotoryye vspomogatel'nyye algoritmy ... [Some auxiliary algorithms for construction of Waterloo-like automata]*. Vestnik of Voronezh State University. Series: System analysis and information technologies. 2016, no. 4, pp. 20–28. (in Russian)

[10] Gera R., Hedetniemi S., and Larson C., editors. *Graph Theory: Favorite Conjectures and Open Problems – 1*. Springer, 2016, 291 p.

[11] Melnikov B. *Once more on the edge-minimization of nondeterministic finite automata and the connected problems*. Fundamenta Informaticae. 2010, vol. 104, no. 3. pp. 267–283.

[12] Melnikov B. and Zubova M. *Postroenie avtomata COM ... [Construction of automaton COM on the base of the basis automaton]*. Vektor Nauki of Togliatti State University, 2010, no. 4, pp. 30–32. (in Russian)

[13] Dolgov V. and Melnikov B. *Postroenie universalnogo konechnogo avtomata. I. Ot teorii ... [Construction of the universal finite automaton. I. From the theory to the practical algorithms]*. Vestnik of Voronezh State University. Series: Physics. Mathematics, 2013, no. 2, pp. 173–181. (in Russian)

[14] Dolgov V. and Melnikov B. *Postroyeniye universal'nogo konechnogo avtomata. II. Primery ... [Construction of universal finite automaton. II. Examples of work of algorithms]*. Vestnik of Voronezh State University. Series: Physics. Mathematics, 2014, no. 1, pp. 78–85. (in Russian)

[15] Melnikov B. and Dolgov V. *Some more algorithms for Conway's universal automaton*. Acta Univ. Sapientiae, Informatica. 2014, vol. 6, no. 1, pp. 5–20.

[16] Melnikov B. and Sciarini-Guryanova N. *Possible edges of a finite automaton defining a given regular language*. The Korean Journal of Com. and Applied Mathematics. 2002, vol. 9, no. 2., pp. 475–485.

[17] Kameda T. and Weiner P. *On the state minimization of nondeterministic finite automata*. IEEE Trans. on Comp. 1970, vol. C-19, no. 7, pp. 617–627.

[18] Baumgärtner S. and Melnikov B. *Multievristicheskiy podkhod k probleme ... [Multiheuristic approach to the problem of star-height minimization of nondeterministic finite automata]*. Vestnik of Voronezh State University. Series: System analysis and information technologies. 2010, no. 1, pp. 5–7. (in Russian)

[19] Melnikov B., Pivneva S., Melnikova E. and Rudnitskiy V. *Parallelnaya realizatsiya zadach ... [The parallel implementation of the optimization problems on the base of multiheuristic approach]*. Samara, ASGARD Ed., 2017. 70 p. (in Russian)

[20] Lombardy S. and Sakarovitch J. *The Universal Automaton*. in: Logic and Automata, Texts in Logic and Games Amsterdam Univ. Press. 2008, vol. 2, pp. 457–504.

[21] Melnikov B., Tsyganov A. and Bulychov O. *A multi-heuristic algorithmic skeleton for hard combinatorial optimization problems*. Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, CSO. Sanya, Hainan. 2009, pp. 33–36.

[22] Melnikov B., Pivneva S. and Rogova O. *Reprezentativnost' sluchayno sgenerirovannykh ... [Representation of randomly generated nondeterministic finite automata from the view of the basis automata]*. Stochastic Optimization in Informatics, 2010, vol. 6, no. 1-1, pp. 74–82. (in Russian)