Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2025, 13(4), 1599-1613
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Combined Algorithm for Searching Waterloo-Like Automata and First Results of Its Application

Mikhail Abramyan[1,2], Boris 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, Russia
Email: m-abramyan@yandex.ru, bormel@mail.ru

How to cite this paper: Abramyan, M. and Melnikov, B. (2025) Combined Algorithm for Searching Waterloo-Like Automata and First Results of Its Application. *Journal of Applied Mathematics and Physics*, **13**, 1599-1613.
https://doi.org/10.4236/jamp.2025.134086

Received: March 9, 2025
Accepted: April 26, 2025
Published: April 29, 2025

Abstract

The paper continues the research on algorithms for generating nondeterministic finite automata possessing the following property (the so-called walibad property): among their covering automata, there exist automata that are not equivalent to the original automaton. The Waterloo automaton has this property; it plays an important role in vertex minimization algorithms. In this paper, we describe a combined algorithm for generating automata with the walibad property and present the first results of its numerical study.

Keywords

Nondeterministic Finite Automata, Vertex Minimization Algorithms for NFA, Universal Automaton, Covering Automaton, Complete Automaton, The Waterloo Automaton

1. Introduction

The paper continues the research related to the development of efficient algorithms for solving the NP-hard problem of vertex minimization of nondeterministic finite automata [1]-[4]. When developing such algorithms, it is necessary to take into account a special situation when there exists non-equivalent covering automata for the analyzed automaton. Such an automaton was first described in [5] and called the Waterloo automaton. The analysis of the Waterloo automaton and similar automata would allow us to study in more detail the problems arising in the minimization of nondeterministic finite automata. In [6] [7], an approach to the construction of automata possessing the same property was described. For brevity, this property was called *walibad* (from "Waterloo-like badness"), and

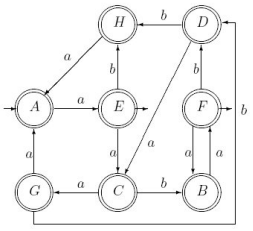
automata with this property were called *walibad automata*.

In [8], two algorithms for constructing walibad automata were described. They were based on the analysis of semilattices of covering automata for the Waterloo automaton [9]-[12], as well as on the results of [6] [7]. Also the results of numerical computations using these algorithms were given. The algorithms were implemented using the NFALib library developed by M. Abramyan.

The present paper continues the research started in [8]. Section 2 gives necessary definitions and facts related to the Waterloo automaton and its properties. Section 3 contains a brief description of algorithms 1 and 2 for generating walibad automata previously discussed in [8]. Section 4 gives a detailed description of an important step in Algorithm 2 related to label generation for the complete automaton (this step was not presented in [8]). Section 5 describes a combined algorithm for finding walibad automata based on the joint use of Algorithms 1 and 2. Section 6 presents the results of numerical computations performed using the simplest variant of the combined algorithm. Section 7 concludes with a brief discussion of directions for further research.

2. Preliminary Information: Waterloo Automaton and Related Objects

Two representations of the Waterloo automaton W are shown in **Figure 1(a)** and **Figure 1(b)** (the second variant of the representation is generated by means of the NFALib library). An important characteristic of an automaton K used in its minimization is the binary relation # [13], which is constructed over the set of states of two automata: the canonical automaton for the original automaton K and the canonical automaton for the mirror automaton K^R (the steps for constructing the relation # are described in more detail in Section 4). The matrix of the relation # for the automaton W is given in **Figure 1(c)**. It should be noted that the relation # is not invariant to the language defined by the automaton: there exist non-equivalent automata with the same relation #.



(a)

		a	b
==>	A	E	-
	B	F	-
	C	G	B
	D	C	H
<==	E	C	H
<==	F	B	D
	G	A	D
	H	A	-

(b)

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

(c)

Figure 1. The Waterloo automaton W (a), (b) and its matrix of the relation # (c).

Based on the matrix of the relation #, a set of grids is constructed. A *grid* $X_0 \times Y_0$ is defined by a pair of subsets $X_0 \in X$ and $Y_0 \in Y$, where X and Y are sets of rows and columns of the matrix of relation #. The subsets X_0 and Y_0 must satisfy two conditions: 1) for any states $x \in X_0$ and $y \in Y_0$, the relation $x \# y$ is satisfied; 2) the subsets X_0 and Y_0 cannot be expanded while preserving condition (1). A set

M of grids is called a *covering set* if for any elements $x \in X$, $y \in Y$ such that $x \# y$, there exists a grid $X_0 \times Y_0$ of M for which $x \in X_0$ and $y \in Y_0$. **Figure 2** shows the complete set of 14 grids for the relation # corresponding to the automaton W. Obviously, this set is a covering set.

Based on the complete set of grids for the automaton K, we can construct a *universal automaton* $COM(K)$ [14] [15], which defines the same regular language L as the original automaton K. In this case, each of the grids corresponds to some state of the $COM(K)$ automaton. Further, on the basis of the $COM(K)$ automaton, we can construct a family of *covering automata*, each of which is obtained by removing some states of the $COM(K)$ automaton provided that the remaining states correspond to the grids forming the covering set.

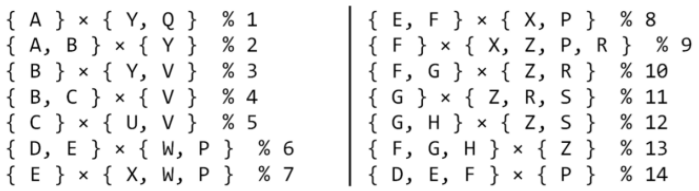


Figure 2. Complete set of grids for the automaton W.

The algorithm for minimizing the original automaton K consists in choosing a covering set of grids of minimal size for which the covering automaton is equivalent to the automaton K. Usually, all covering automata are equivalent to the original automaton K, so it is sufficient to choose a covering set of grids M_0 of minimal size to minimize it. However, this is not the case for the Waterloo automaton, since its set M_0, which has size 7 and consists of grids with numbers 1, 3, 5, 5, 6, 8, 10, 12, corresponds to a covering automaton that *is not equivalent to the automaton* W.

In [9]-[12], a more detailed numerical analysis of the properties of covering automata was performed for the Waterloo automaton. The following results were obtained:

- the number of covering sets of grids for the Waterloo automaton is 260; the minimal set contains 7 grids, and such a set is the only one;
- there are 8 pairwise non-equivalent sets of equivalent covering automata that are non-equivalent to the original Waterloo automaton, with 3 sets consisting of 16 elements and 5 sets consisting of 4 elements.

Table 1 summarizes the properties of the automaton W; in particular, column

Table 1. Properties of the Waterloo automaton.

Automaton (1)	Matrix # Size (2)	Grids (3)	Covering sets of grids (4)	Sets of covering automata non-equivalent to original one (5)
W	8 × 10	14	260 (7, 1)	16 (7), 16 (8), 16 (8), 4 (9), 4 (9), 4 (9), 4 (10), 4 (10)

(4) shows the number of covering sets of grids and, in parentheses, the size of the minimum covering set and the number of minimum covering sets, while column (5) gives information about each of the sets of equivalent covering automata that are non-equivalent to the original automaton: the size of the set and, in parentheses, the number of states of the minimal covering automaton included in the set.

3. Two Algorithms for Finding Waterloo-Like Automata

In this section, we briefly describe two basic heuristic algorithms that can be applied to find new walibad automata based on existing ones. These algorithms are described in more detail in [8].

The first algorithm is very simple. It seems quite natural that some of the covering automata, which are non-equivalent to the original walibad automaton, possess the walibad property. This hypothesis was confirmed by numerical computations and allows us to formulate the first of the heuristic algorithms for finding new walibad automata (see [8], Section 3).

Algorithm 1. On the basis of an existing walibad automaton, construct the complete set of its covering automata, select from it equivalent sets of automata that are not equivalent to the original one, and check the representative of each set for the presence of the walibad property. As representatives, choose from each set the automaton for which the size of the covering set of grids is minimal. If an automaton possessing the walibad property is found, then apply the same procedure to it.

It should be noted that this algorithm is recursive.

In order to describe the second heuristic algorithm, we need to introduce additional objects and operations. The second algorithm is based on the analysis and transformation of the so-called *complete automaton* [6] [7], which is built on the basis of the original walibad automaton. After the complete automaton is constructed, some symbols of its alphabet are provided with special labels (one or more), which are then used in performing the following (non-equivalent) transformations of this automaton [6] [7]:

1) *removing* symbols not associated with any labels; as a result, the size of the alphabet is reduced;

2) *splitting* symbols of the alphabet if they are associated not with one, but with several labels; as a result, the size of the alphabet increases, and exactly one label is associated with each symbol;

3) complete or partial *reduction* of alphabet symbols, in which symbols' labels are taken into account.

In [6], it is proved that as a result of successive application of these three transformations (including complete reduction) the complete automaton is transformed into the original automaton on the basis of which the complete automaton was created. However, if we apply partial reduction rather than complete reduction, new automata will be obtained. This is the basis of the second heuristic algorithm for finding walibad automata (see [8], Section 5).

Algorithm 2. A complete automaton is constructed on the basis of some walibad automaton. Then a set of transformations described above, including the partial reduction transformation, is applied to it. The resulting automata are checked for the existence of the walibad property.

4. Complete Automaton: Defining, Labeling and Performing Various Transformations

The paper [8] did not describe an important step in the implementation of Algorithm 2, namely, the step of adding special labels to the complete automaton (more precisely, to some symbols of the alphabet of this automaton), which are then used in subsequent transformations of the complete automaton. It was only noted that these labels are constructed using the *basis automaton BA*, and reference was made to [6], from which an algorithm for generating these labels can be extracted. However, due to the importance of the label generation step, we now fully describe the algorithm for their generation, using as an example the same original automaton K that was used for similar purposes in [6]. In this description, we give the representations of all necessary automata generated by the NFALib library.

First of all, we give a view of the original automaton K and the automaton K^R, which is a mirror automaton for K (**Figure 3(a)** and **Figure 3(b)**). In these automata, the states are denoted by q1, q2, q3, and the alphabet consists of the symbols a and b.

	a	b
==> q1	q3	q2
<== q2	q2	q2
<== q3	-	q1,q3

(a)

	a	b
<== q1	-	q3
==> q2	q2	q1,q2
==> q3	q1	q3

(b)

Figure 3. Automata K (a) and K^R (b).

	a	b	% States
==> A	B	C	% q1
<== B	-	D	% q3
<== C	C	C	% q1-q2-q3
<== D	B	C	% q1-q3

(a)

	a	b	% States
==> X	Y	Z	% q2-q3
<== Y	U	Z	% q1-q2
<== Z	Y	Z	% q1-q2-q3
U	U	Y	% q2

(b)

	X	Y	Z	U
A	-	#	#	-
B	#	-	#	-
C	#	#	#	#
D	#	#	#	-

(c)

Figure 4. Canonical automata $K^{\sim}$ (a) and $(K^R)^{\sim}$ (b) and the matrix # (c).

In **Figure 4(a)** and **Figure 4(b)**, the *canonical automata* to the original and mirror automata are given. They are denoted by $K^{\sim}$ and $(K^R)^{\sim}$, respectively. These are deterministic automata whose states are related to the *sets* of states of the original automata (these sets are given in the additional column named "States"; the symbol "-" is used as a separator when describing these sets). It is convenient to denote the states of the automaton $K^{\sim}$ by the initial letters of the alphabet, and the states of the automaton $(K^R)^{\sim}$ by the final letters. The letter U and the ordering of the

states for the automaton $(K^R)^\sim$ are chosen in this way because this is the representation of this automaton used in [6].

On the basis of canonical automata, the relation # matrix is constructed (**Figure 4(c)**).

The rule of the construction if the relation # matrix is easy to understand if we analyze the representations of canonical automata. The states for the canonical automata $K^\sim$ and $(K^R)^\sim$ are in relation # if and only if the same state of the original automaton K is present in the States column for these states. For example, state A of automaton $K^\sim$ is related to the single state q1 of automaton K, so in the first row of the matrix, the symbol # is located in those columns which correspond to the states of automaton $(K^R)^\sim$ related to state q1 (these are states Y and Z). The state D of the automaton $K^\sim$ is related to the states q1 and q3 of automaton K, so in the last row of the matrix, the symbol # is placed in the columns that correspond to the states of the automaton $(K^R)^\sim$ related to the states q1 or q3 (these are the states X, Y and Z).

As it was mentioned in [8], a helper automaton, the *basis automaton* $BA(K)$, is used to construct labels. The states of this automaton are pairs of states $K^\sim$ and $(K^R)^\sim$, which are in the relation #. Such states in the representation of the automaton $BA(K)$ are denoted with double underline (**Figure 5**). The basis automata and, in particular, the algorithm of their construction are considered in [16]. The NFALib library contains the function GetBasisNFA, which constructs a basis automaton for an automaton K taking into account the current form of the canonical automata $K^\sim$ and $(K^R)^\sim$ (and, correspondingly, the relation # matrix).

		a	b
==>	A__Y	B__X,B__Z	C__U
==>	A__Z	-	C__X,C__Y,C__Z
<==	B__X	-	-
	B__Z	-	D__X,D__Y,D__Z
<==	C__X	-	-
	C__Y	C__X,C__Z	C__U
	C__Z	-	C__X,C__Y,C__Z
	C__U	C__U,C__Y	-
<==	D__X	-	-
	D__Y	B__X,B__Z	C__U
	D__Z	-	C__X,C__Y,C__Z

Figure 5. Basis automaton $BA(K)$.

Figure 6 shows a view of the *complete automaton* $K^\#$ for automaton K. The transition table of this automaton can be easily obtained from the representation of the matrix of the relation # by augmenting it with the notations of input and output states corresponding to the input and output states of the automaton $K^\sim$. Each symbol of the alphabet of the automaton $K^\#$ is associated with a characteristic defined by a *pair of states*, in which the first element of the pair is a state of the automaton $(K^R)^\sim$, and the second element of the pair is a state of the automaton $K^\sim$. These characteristics are given in the last line of the definition of the automaton $K^\#$, with the elements of the pair separated by the symbol "^". The rule of

formation of characteristics is easy to determine from the representation of the automaton $K^{\#}$.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
==> A	-	-	-	-	A	B	C	D	A	B	C	D	-	-	-	-
<== B	A	B	C	D	-	-	-	-	A	B	C	D	-	-	-	-
<== C	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
<== D	A	B	C	D	A	B	C	D	A	B	C	D	-	-	-	-
%	X^A	X^B	X^C	X^D	Y^A	Y^B	Y^C	Y^D	Z^A	Z^B	Z^C	Z^D	U^A	U^B	U^C	U^D

Figure 6. Complete automaton K#.

Now we have all the necessary objects to describe the algorithm for generating additional labels associated with some symbols of the alphabet of the automaton $K^{\#}$ (or, in other words, with some columns of its representation).

The labels correspond to non-empty transitions for the automaton $K^{\sim}$, and thus are uniquely determined by the initial state and the alphabet symbol. In our case, there are 7 labels: A:a, C:a, D:a, A:b, B:b, C:b, D:b (label B:a is absent, since this transition is empty for the automaton $K^{\sim}$). Note that in [6], instead of such label designations, the order numbers were introduced for the edges of the graph of the automaton $K^{\sim}$ (the edges were numbered from (1) to (7); their order corresponds to the order of enumeration of labels given above).

Each of these labels is associated with one or more symbols of the automaton $K^{\#}$ alphabet on the basis of an algorithm, which is easiest to describe on the following example.

The first label A:a corresponds to the transition A –(a)→ B for the automaton $K^{\sim}$. We analyze the transitions for the automaton $BA(K)$ corresponding to symbol a, for which the initial state contains symbol A and the final state contains symbol B. This is the only transition A__Y –(a)→ B__X, B__Z. This transition corresponds to the only transition for the complete automaton $K^{\#}$ of the following form: A –(Y^B)→ B. When constructing this transition, the symbols Y and B located to the left and right of the arrow in the transition A__Y –(a)→ B__X, B__Z are moved "inside" the arrow, i. e. they define the edge Y^B of the $K^{\#}$ automaton, along which the transition is performed; in addition, only the symbol B is left to the right of the arrow in the resulting transition A –(Y^B)→ B. The resulting transition means that the label A:a must be associated with the column f that corresponds to the pair Y^B (see **Figure 7**).

It should be noted that not the whole column is actually labeled, but the column element corresponding to the first letter of the label, i. e., in this case, the element of the first row.

Let us describe similar actions for the second label C:a. This label corresponds to the transition C –(a)→ C for the automaton $K^{\sim}$. We analyze the transitions for the automaton $BA(K)$ corresponding to the symbol a, for which both initial and final states contain the symbol C. These are the transitions C__Y –(a)→ C__X,C__Z and C__U –(a)→ C__U,C__Y. Transforming these transitions according to the rules described above for the first label A:a, we obtain two

transitions for the complete automaton $K^{\#}$: C –(Y^C)→ C and C –(U^C)→ C. These transitions mean that the label C:a should be associated with columns g and o, which correspond to pairs Y^C and U^C respectively (see **Figure 7**).

```
      a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p
==> A -   -   -   -   A   B   C   D   A   B   C   D   -   -   -   -
<== B A   B   C   D   -   -   -   -   A   B   C   D   -   -   -   -
<== C A   B   C   D   A   B   C   D   A   B   C   D   A   B   C   D
<== D A   B   C   D   A   B   C   D   A   B   C   D   -   -   -   -
%     X^A X^B X^C X^D Y^A Y^B Y^C Y^D Z^A Z^B Z^C Z^D U^A U^B U^C U^D
%                     A:a C:a                                   C:a
```

Figure 7. Complete automaton $K^{\#}$ with two first labels (A:a and C:a).

Repeating the described actions for all 7 labels, we obtain the representation of the automaton $K^{\#}$ shown in **Figure 8**. In this representation, the labels are placed in the penultimate line. The NFALib library has a special function SetCompleteInfo2, which adds this additional set of labels to the complete automaton. As noted above, labels with order numbers (1)-(7) were used for similar purposes in [6, Table 8]; these labels are shown in **Figure 8** in the last line. It should be noted that in [6] one of the labels was omitted, namely, label (4) for column k.

```
      a   b   c   d   e   f       g               h   i   j   k           l   m   n   o   p
==> A -   -   -   -   A   B       C               D   A   B   C           D   -   -   -   -
<== B A   B   C   D   -   -       -               -   A   B   C           D   -   -   -   -
<== C A   B   C   D   A   B       C               D   A   B   C           D   A   B   C   D
<== D A   B   C   D   A   B       C               D   A   B   C           D   -   -   -   -
%     X^A X^B X^C X^D Y^A Y^B     Y^C             Y^D Z^A Z^B Z^C         Z^D U^A U^B U^C U^D
%                     A:a,D:a C:a,A:b,C:b,D:b             A:b,C:b,D:b B:b         C:a
%                     (1),(3) (2),(4),(6),(7)             (4),(6),(7) (5)         (2)
```

Figure 8. Complete automaton K# with all labels.

The obtained labels allow us to reconstruct the canonical automaton $K^{\sim}$ for the original automaton K by the canonical automaton $K^{\#}$. For this purpose, the transformations (1)-(3) described in Section 3 should be successively applied to the labeled automaton $K^{\#}$. Each of these transformations is associated with a corresponding method of the NFALib library. **Figure 9** shows the automaton K_1 obtained from the automaton $K^{\#}$ using transformation (1) (removing unlabeled columns). **Figure 10** shows automaton K_2 obtained from automaton K_1 using transformation (2) (splitting labeled columns). If we apply to automaton K_2 the variant of transformation (3), which consists in performing a complete reduction, we obtain automaton K_3 (**Figure 11**), which coincides with automaton $K^{\sim}$ (see **Figure 4(a)**).

```
      f         g                 k           l     o
==> A B         C                 C           D     -
<== B -         -                 C           D     -
<== C B         C                 C           D     C
<== D B         C                 C           D     -
%     Y^B       Y^C               Z^C         Z^D   U^C
%     A:a,D:a   C:a,A:b,C:b,D:b   A:b,C:b,D:b B:b   C:a
```

Figure 9. Automaton K_1 obtained from automaton $K^{\#}$ using transformation (1).

	a	b	c	d	e	f	g	h	i	j	k
==> A	B	B	C	C	C	C	C	C	C	D	-
<== B	-	-	-	-	-	-	C	C	C	D	-
<== C	B	B	C	C	C	C	C	C	C	D	C
<== D	B	B	C	C	C	C	C	C	C	D	-
%	Y^B	Y^B	Y^C	Y^C	Y^C	Y^C	Z^C	Z^C	Z^C	Z^D	U^C
%	A:a	D:a	C:a	A:b	C:b	D:b	A:b	C:b	D:b	B:b	C:a

Figure 10. Automaton K_2 obtained from automaton K_1 using transformation (2).

	a	b
==> A	B	C
<== B	-	D
<== C	C	C
<== D	B	C

Figure 11. Automaton K_3 obtained from automaton K_2 using transformation (3).

Only the reduction stage (which was described in detail in [8]) needs additional explanations. Using the example of transition from automaton K_2 (**Figure 10**) to automaton K_3 (**Figure 11**), this stage can be described as follows. Some label (e. g., A:a) is selected. It is used to construct a transition for automaton K_3, for which the initial state and the alphabet symbol coincide with the state and the alphabet symbol of the label (in this case, A and a), and the final state is taken from automaton K_2, namely, from the table cell located at the intersection of the row with the initial state and the column containing the label being analyzed (in this case, it is row A and column a, at the intersection of which is the state B). For the label D:a, which is located in column b, similar actions construct a transition for the automaton K_3, which starts from the state D, uses the symbol a of the alphabet and ends in the state B (since the intersection of row D and column b of the automaton K_2 is the state B).

5. Description of the Combined Algorithm for Finding Walibad Automata

Based on the algorithms 1 and 2 described above (see Section 3), a combined algorithm using NFALib library tools was developed.

The input of the algorithm is an original automaton K possessing the walibad-property (the Waterloo automaton W was used as such an automaton in numerical experiments).

Step 1. Automaton K is added to the queue Q of detected walibad automata.

Step 2. For automaton K, a complete analysis of its semilattices is performed, on the basis of which all its covering automata are constructed, sets of pairwise equivalent covering automata, which non-equivalent to automaton K, are selected, and for each such a set, a minimal representative (an automaton corresponding to the covering set of grids of minimal size) is chosen. For this representative, checking the walibad property is performed. If this representative has the walibad property, then it is also added to the queue Q of found walibad automata, and its semilattices are analyzed for it immediately. Such recursive actions are performed until all covering automata possessing the walibad property have

been processed. The names of covering automata are constructed on the basis of the automaton K by adding the symbol "-" and the order number of the set of covering automata for which this automaton is the minimal representative.

As an example of the implementation of step 2, let us give the results of its application to automaton W. In a recursive analysis of its covering automata, the set of covering walibad automata shown in **Figure 12** was found.

```
W[8] W-1[3] W-2[2] W-2-2[1] W-3[2] W-3-2[1] W-7[3] W-7-3[1] W-8[3] W-8-3[1]
```

Figure 12. Automaton W and a set of its covering automata possessing walibad property.

The automata are listed in **Figure 12** in the order of their analysis; after the name of the covering automaton, the total number of sets of covering automata non-equivalent to the original one is given in square brackets. In particular, the notation W [8] means that 8 sets of pairwise equivalent covering automata, non-equivalent to the original one, were found for the automaton W (see also **Table 1** in Section 2), and the automata from the sets 1, 2, 3, 7 and 8 turned out to be walibad automata. For the minimal representatives W-1, W-2, W-3, W-7 and W-8 chosen from these sets, in turn, their covering automata were analyzed, which revealed additional walibad automata of the "second level": W-2-2, W-3-2, W-7-3 and W-8-3. W-2-2 denotes the minimal representative of the second set of covering automata for automaton W-2, and W-8-3 denotes the minimal representative of the third set of covering automata for automaton W-8. At the third level of recursion (when analyzing covering automata for W-2-2, W-3-2, W-7-3 and W-8-3), no new walibad automata were found; thus, step 2 for automaton W was completed.

Note that step 2 corresponds to Algorithm 1 described above, and the given example of its implementation corresponds to the results of its application to the Waterloo automaton (see [8, Tables 1 and 2]).

Step 3. If the queue Q is non-empty, then the first walibad automaton K is extracted from Q, the complete automaton $K^{\#}$ is constructed for automaton K, and the transformations (1)-(3) described above, including the operation of partial reduction, are applied to the automaton $K^{\#}$. Additional parameters of these transformations are defined in the settings of the combined algorithm. All reduced variants of complete automaton are analyzed for the presence of walibad property, and if such property is detected, steps 1 and 2 are executed for these variants immediately. The names of the reduced variants of complete automaton are constructed based on the name of the original walibad automaton K by adding to it the character "-" and a string of characters corresponding to the complete automaton columns on which the reduction is performed. The columns use lowercase and uppercase Latin and Russian letters in alphabetical order (if the size of the alphabet exceeds 116 elements, an error message is displayed and this automaton is not analyzed). After the analysis of all reduced variants of complete-automaton is completed, step 3 for automaton K is finished, and the next walibad automaton is extracted from queue Q, for which step 3 is performed.

As an example of the implementation of step 3, we give the results of this step for the automaton W in the case when the maximum number of columns to be reduced is 3. In **Figure 13**, only reduced automata, for which the walibad property was found, are given. The first line contains the alphabet symbols of the automaton to which the partial reduction operation is applied (this is the transformed complete automaton for automaton W).

These results correspond to the data given in [8, Table 4]; however, **Figure 13** additionally shows the results of recursive analysis (step 2) of covering automata.

The combined algorithm terminates either when the automata contained in the queue Q are exhausted or when a predetermined number of automata is analyzed for the presence of the walibad property. During the algorithm's operation, information about all analyzed automata is saved, so when the algorithm is run again, it does not re-process the automata that were analyzed earlier.

```
W: abcdefghijklmno
ace: W-ace[8] W-ace-4[5] W-ace-4-5[3] W-ace-7[5] W-ace-7-5[3] W-ace-8[3]
acf: W-acf[8] W-acf-5[5] W-acf-5-5[3] W-acf-7[5] W-acf-7-5[3] W-acf-8[3]
ade: W-ade[8] W-ade-4[5] W-ade-4-5[3] W-ade-5[5] W-ade-5-5[3] W-ade-8[3]
adf: W-adf[8] W-adf-4[5] W-adf-4-5[3] W-adf-5[5] W-adf-5-5[3] W-adf-8[3]
bce: W-bce[8] W-bce-4[5] W-bce-4-5[3] W-bce-7[5] W-bce-7-5[3] W-bce-8[3]
bcf: W-bcf[8] W-bcf-5[5] W-bcf-5-5[3] W-bcf-7[5] W-bcf-7-5[3] W-bcf-8[3]
bde: W-bde[8] W-bde-4[5] W-bde-4-5[3] W-bde-7[5] W-bde-7-5[3] W-bde-8[3]
bdf: W-bdf[8] W-bdf-4[5] W-bdf-4-5[3] W-bdf-7[5] W-bdf-7-5[3] W-bdf-8[3]
gik: W-gik[3]
gil: W-gil[3]
gjk: W-gjk[3]
gjl: W-gjl[3]
hik: W-hik[3]
hil: W-hil[3]
hjk: W-hjk[3]
hjl: W-hjl[3]
```

Figure 13. Result of applying step 3 of the combined algorithm to automaton W.

6. Some Results of Applying the Simplest Variant of the Combined Algorithm

When describing step 3 of the combined algorithm, it was noted that different variants can be used for it. The results described below are obtained for the variant with the following features.

1) The transformation (2) associated with column splitting is *excluded* from the transformations of the complete automaton; this means that the columns of the automaton to which partial reduction is applied can contain *several* labels.

2) Partial reduction is applied to only one set of columns; all labels associated with these columns must contain the same alphabet character specified after the ":" character.

3) The maximum number of columns for which partial reduction is applied is assumed to be 3.

When running this simplest variant of the combined algorithm, 30003 automata were generated and analyzed for the walibad property, of which 1636 automata (5.5% of the total number of analyzed automata) possessed this property. At this point, the queue Q contained 859 automata waiting to be analyzed; thus, the

algorithm was not terminated "naturally" (due to the exhaustion of automata in the queue Q), but was explicitly terminated when the specified number of processed automata was reached.

None of the automata obtained by the reduction of one or two columns turned out to be an automaton with the walibad property, so the labels of the walibad automata found always contain sets of *triples* of the letter names of the reduced columns.

Among the 1636 walibad automata obtained, there are 846 groups of pairwise equivalent automata, with 423 groups containing one automaton and 343 groups containing two or three automata. Thus, the number of non-equivalent automata is quite large.

At the same time, if we perform grouping of all found walibad automata, including in each group automata with equivalent matrices of the relation #, then only 21 groups will be obtained. This result is quite consistent with the results of [6] [7], in which it is noted that when performing non-equivalent transformations (1)-(3) of the complete automaton, as a rule, automata with the same relation # matrix as the original complete automaton are obtained.

If we group all found walibad automata according to another characteristic, namely, coincidence of properties (2)-(5) given in **Table 1** from Section 2, then the number of such groups will be 19. **Table 2** contains descriptions of all these

Table 2. Groups of walibad-automata derived from the Waterloo automaton W using the simplest variant of the combined algorithm.

No.	Group description (1)	Matrix # Size (2)	Grids (3)	Covering sets of grids (4)	Sets of covering automata non-equivalent to original one (5)
1	W (1)	8 × 10	14	260 (7, 1)	16 (7), 16 (8), 16 (8), 4 (9), 4 (9), 4 (9), 4 (10), 4 (10)
2	W-1 (25)	8 × 8	12	65 (7, 1)	1(9), 2(9), 2(9)
3	W-2 (2)	10 × 9	13	60 (8, 2)	2(9), 2(10)
4	W-2-2 (4)	10 × 9	12	18 (9, 4)	2(9)
5	W-7 (2)	8 × 10	13	80 (8, 3)	12(8), 4(9), 4(9)
6	W-ace (26)	8 × 10	14	260 (7, 1)	1(9), 4(9), 4(9), 1(10), 1(10), 4(10), 4(10), 1(11)
7	W-ace-4 (164)	9 ×10	15	400 (8, 3)	5(9), 20(9), 5(10), 5(10), 5(11)
8	W-ace-4-5 (244)	9 × 10	15	325 (8, 1)	13(9), 13(10), 13(10)
9	W-gik (24)	8 × 10	14	260 (7, 1)	16(7), 16(8), 16(8)
10	W-2-adh (88)	10 × 10	14	100 (8, 1)	2(10), 2(11)
11	W-2-adh-2 (256)	11 × 10	15	150 (9, 2)	10(10)
12	W-2-dhk (32)	10 × 10	14	120 (8, 2)	4(9), 4(10)

Continued

13	W-2-dhk-2 (88)	11 × 10	15	180 (9, 4)	20(9)
14	W-2-2-adh (176)	10 × 10	13	30 (9, 2)	2(10)
15	W-2-2-dhk (64)	10 × 10	13	36 (9, 4)	4(9)
16	W-7-aei (48)	8 × 10	13	80 (8, 3)	1(9), 4(9), 1(10), 1(10), 1(11)
17	W-7-aei-5 (144)	9 × 10	14	125 (8, 1)	5(9), 5(10), 5(10)
18	W-7-cgk (48)	8 × 10	13	80 (8, 3)	4(8), 4(9), 4(9)
19	W-ace-4-bde-5 (200)	10 × 10	16	625 (8, 1)	25(9), 25(10), 25(10)

groups, with column (1) containing the name of the first walibad automaton with this characteristic and, in parentheses, the total number of automata belonging to this group. The representatives of each group are listed in the order in which they were found. It should be noted that the first representative of the last, 19th group (automaton *W*-ace-4-bde-5), is a walibad automaton with the order number 226; thus, all subsequent walibad automata found (with order numbers from 227 to 1636) belong to one of the already existing groups.

The relations between groups of automata with equivalent matrices of relation # and groups of automata with the same characteristics from **Table 2** are rather complicated. Some groups from **Table 2** correspond to groups of automata with equivalent matrices; these are groups 2, 3, 4, 8, 10, 11, 12, 13. Some groups from **Table 2** include two groups of automata with equivalent matrices; these are groups 7, 14, 15, 17, 19. On the other hand, groups 1, 6, and 9 from **Table 2** form one group of automata with equivalent matrices; note that this group includes those and only those automata whose names do not have numerical components. Finally, groups 5, 16, and 18 from **Table 2** include automata belonging to *two* groups with equivalent matrices of the relation #.

These results indicate that when using reduction on a small number of columns (even if we do not perform splitting of columns containing several labels beforehand) it is not reasonable to generate and process a large number of automata, since, starting from some point, all new walibad automata will have the basic characteristics that the previously found automata had.

7. Conclusions

The variant of the combined algorithm described in this paper allows a number of modifications that can make its application more efficient and allow us to obtain a more representative set of different types of walibad automata. In particular, it is possible to use additional actions to cut off walibad automata that have properties that coincide with the previously processed automata. The cutoff in this case means that such automata will not be added to the queue Q for the subsequent application of step 3 to them. Such a variant with cutoff, unlike the simplest variant considered in this paper, will not repeatedly analyze automata with similar

properties and, therefore, can quickly enough terminate "naturally", due to the exhaustion of automata in the queue Q.

Another direction in which the basic combined algorithm can be modified is to change the way of performing partial reduction in step 3. In particular, it seems justified to perform the reduction over a sufficiently large number of columns, since in this case a greater variety of found automata with the walibad property can be achieved.

Of course, a more detailed analysis of the obtained sets of walibad automata and selection of automata with particularly interesting properties from them is an urgent task. As an example of such automata, we can point out the automata of groups 6 and 9 from **Table 2**. These automata are interesting because their sets of pairwise equivalent covering automata, which are not equivalent to the original automaton, almost exactly correspond to one of the two parts of the similar sets for the Waterloo automaton (the view of automata *W*-ace and *W*-gik was given in [8], Figs. 8 and 9).

Conflicts of Interest

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

References

[1] Jiang, T. and Ravikumar, B. (1993) Minimal NFA Problems Are Hard. *SIAM Journal on Computing*, **22**, 1117-1141. https://doi.org/10.1137/0222067

[2] Melnikov, B.F. and Melnikova, A.A. (2001) Edge-Minimization of Non-Deterministic Finite Automata. *Korean Journal of Computational & Applied Mathematics*, **8**, 469-479. https://doi.org/10.1007/bf02941980

[3] Polak, L. (2005) Minimalizations of NFA Using the Universal Automaton. *International Journal of Foundations of Computer Science*, **16**, 999-1010.

[4] Melnikov, B. (2010) Once More on the Edge-Minimization of Nondeterministic Finite Automata and the Connected Problems. *Fundamenta Informaticae*, **104**, 267-283. https://doi.org/10.3233/fi-2010-349

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

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

[7] Melnikov, B.F. and Melnikova, E.A. (2017) Waterloo-Like Finite Automata and Algorithms for Their Automatic Construction. *International Journal of Open Information Technologies*, **5**, 8-15.

[8] Abramyan, M. and Melnikov, B. (2024) On Some Approaches to Automatic Generation of Waterloo-Like Automata. *Proceedings of the* 2024 8*th International Conference on Computer Science and Artificial Intelligence*, Beijing, 6-8 December 2024, 544-550. https://doi.org/10.1145/3709026.3709091

[9] Zyablitceva, L.V., Korabelshchikova, S.Yu. and Abramyan, M.E. (2023) Semilattice Graphs, Matrix Representations of Idempotent Semigroups and Waterloo Automaton Semilattices. *International Journal of Open Information Technologies*, **11**, 69-76.

(In Russian)

[10] Abramyan, M.E. and Melnikov, B.F. (2023) A Program Study of the Union of Semilattices on the Set of Subsets of Grids of Waterloo Language. *Journal of Applied Mathematics and Physics*, **11**, 1459-1470. https://doi.org/10.4236/jamp.2023.115095

[11] Abramyan, M.E. and Melnikov, B.F. (2023) On the Study of All Semilattices on the Set of Covering Automata for the Waterloo Automaton. *Proceedings of the* 2023 7*th International Conference on Computer Science and Artificial Intelligence*, Beijing, 8-10 December 2023, 486-494. https://doi.org/10.1145/3638584.3638588

[12] Abramyan, M. (2025) On the Representation of Classes of Semilattices on the Set of Covering Automata for the Waterloo Automaton. In: *Proceedings of the* 6*th International Conference on Informatics Engineering and Information Science* (*ICIEIS* 2024), Springer, 63-71. https://doi.org/10.1007/978-981-96-1108-9_6

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

[14] Lombardy, S. and Sakarovitch, J. (2008) The Universal Automaton. In: *Logic and Automata* (*Texts in Logic and Games*), Amsterdam Univ. Press, 457-504.

[15] Dolgov, V. and Melnikov, B. (2014) Construction of Universal Finite Automaton. II. Examples of Algorithms Functioning. *Bulletin of the Voronezh State University. Series: Physics. Mathematics*, **1**, 78-85. (In Russian)

[16] Melnikov, B. and Melnikova, A. (2013) Some More on the Basis Finite Automaton. *Acta Universitatis Sapientiae*, *Informatica*, **5**, 227-244. https://doi.org/10.2478/ausi-2014-0012