Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2023, 11, 1459-1470
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

A Program Study of the Union of Semilattices on the Set of Subsets of Grids of Waterloo Language

Mikhail E. Abramyan[1,2], Boris F. Melnikov[1]

[1]Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU-BIT University, Shenzhen, China
[2]Algebra and Discrete Mathematics Department, Southern Federal University, Rostov-on-Don, Russian Federation
Email: m-abramyan@yandex.ru, bormel@mail.ru

How to cite this paper: 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

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

Abstract

The aim is to study the set of subsets of grids of the Waterloo language from the point of view of abstract algebra and graph theory. The study was conducted using the library for working with transition graphs of nondeterministic finite automata NFALib implemented by one of the authors in C#, as well as statistical methods for analyzing algorithms. The results are regularities obtained when considering semilattices on a set of subsets of grids of the Waterloo language. It follows from the results obtained that the minimum covering automaton equivalent to the Waterloo automaton can be obtained by adding one additional to the minimum covering set of grids.

Keywords

Nondeterministic Finite Automata, Universal Automaton, Basic Automaton, Grid, Covering Automaton, Equivalent Transformation Algorithms, Waterloo Automaton

1. Introduction and Preliminaries

It is well known that there exist different complete invariants for describing a regular language: not only well-known canonical automata [1] [2] [3] [4], but also basis automata [5] [6] [7] and universal automata [8] [9] [10] [11]. When constructing basis and universal automata, it is necessary to construct canonical automata both for a given regular language and for its mirror language. In the process of such construction, we can obtain, among other objects, a special binary relation # defined on pairs of states of these two canonical automata. This relation is also invariant (but incomplete) for the language in question.

Of course, at present, the most interesting language for research is the Waterloo language. The term "Waterloo automaton" is used in most of the articles, both in English and in Russian. However, it will be clear from what follows that the term "Waterloo language" is better.

A universal automaton constructed for this language has the following property: there exists a non-equivalent automaton among its corresponding covering automata [11] [12] [13]. It is still unknown whether a Waterloo language is minimal, but most likely it is.

And, of course, it is desirable to define "minimality" mentioned in the previous paragraph strictly; it can be considered according to different norms. The most natural norm is the number of states of an equivalent canonical automaton, but we are more interested in other variants:

- either the number of states of an equivalent basis automaton—for the Waterloo language this value is 20;
- or the product of the number of states of two equivalent canonical automata (for the given regular language and for the language mirroring the given one)—for Waterloo this value is 80.

Covering automata have already been mentioned above. Of course, for any automaton, they form a semilattice by union, but it can be shown that they do not, generally speaking, form a semilattice by intersection. More concretely, instead of one intersection semilattice, a union of several such semilattices is formed. This paper is devoted to consideration of such a construction for the Waterloo language.

Thus, the topic under consideration is adjacent to the study of the set of all possible arcs of a nondeterministic finite automaton defining a given regular language; such a study was started by one of the authors of this paper back in the late 1990s [14] [15] and was subsequently continued in [10] [11] [13] [16].

Thus, as it has already been mentioned, when we study the set of possible arcs of an automaton (in other words, the arcs of a COM automaton, the arcs of a universal automaton), we obtain a special binary relation # defined on pairs of states of these two canonical automata as an auxiliary construction. This relation is also invariant (but incomplete) for the language in question. For each such binary relation, there exists a whole subclass of the class of regular languages that possesses it. Hence, on the set of all regular languages, it is possible to define (another) binary relation; it is valid for some two languages if and only if they have the same binary relation #. Obviously, the binary relation defined in this way (in some of our previous works, [17] [18] [19], it was denoted by R) is an equivalence relation on the set of all regular languages. This raises the question of the "most typical" language, which is an element of the class of equivalence under the relation in question.

In some previous papers, we have studied in detail regular languages and their corresponding canonical automata, which can be considered as such "typical elements" of their class, and considered some of their properties [19] [20] [21]. Apparently, the most interesting of these properties is the following: using spe-

cially described transformations, we can obtain from such a "typical" automaton any canonical automaton whose language corresponds to the binary relation # under consideration.

Let us repeat that so far we know only one example of a Waterloo-like language (*i.e.* a language for which there exists a covering automaton which is not equivalent to it); the article [22] can be considered as a formulation of the problem of finding other such languages. Let us also note the possible connection of Waterloo-like languages with infinite iterations of finite languages [18] [23] [24] and others. We also note that many of the problems listed here can be considered as container packing problems (see [25] and many others); of course, it is not expedient to solve them in this way; they require other models similar to those considered in this paper.

Most of the terminology related to nondeterministic finite automata corresponds to [17] (see also references from that paper). Here we add references to some terms directly related to automata minimization. The vertex minimization algorithm of nondeterministic finite automata considered in [4; Section 6] is based on the analysis of the binary relation # [4; Section 3.3] connecting the sets of states X and Y of two canonical automata which are constructed on the basis of the analyzed nondeterministic finite automaton K (defining some regular language L) and a mirror to it. A set of grids [4; Section 3.4] is associated with relation #. Each grid is defined by a pair of subsets $X_0 \subset X$ and $Y_0 \subset Y$ satisfying the following condition: for any states $x \in X_0$ and $y \in Y_0$, the relation $x \# y$ holds, and the subsets X_0 and Y_0 cannot be expanded while keeping the condition. We denote such a grid by $X_0 \times Y_0$. Note that grids can be considered as states of a universal automaton.

A set M of grids is called a covering grid if for any elements $x \in X$ and $y \in Y$ such that $x \# y$, there exists a grid $X_0 \times Y_0$ of M for which $x \in X_0$ and $y \in Y_0$. Obviously, the complete set of grids constructed by mapping # is a covering set.

In [11], we describe an algorithm for constructing from the complete set of grids a COM (L) automaton (actually isomorphic to a universal automaton) which defines the same regular language L as the original automaton K, and each grid corresponds to some state of the COM (L) automaton. Based on a COM (L) automaton, we can define a family of covering automata, each of which is obtained by removing some states of a COM (L) automaton, and the remaining states correspond to grids which form the covering set.

The minimization algorithm for the original automaton K consists in choosing a covering grid set M_0 of minimal size, for which the correspondent covering automaton is equivalent to the automaton K, *i.e.* defines the same regular language. Examples show that not every covering grid set leads to a covering automaton equivalent to the original one. A well-known example is the Waterloo automaton first given in [26] and analyzed in detail in [4; Sec. 6].

This paper presents the results of an additional study of covering automata based on the Waterloo automaton. The study was performed using the library

for working with nondeterministic automata NFALib implemented by one of the authors in C#, [27] *et al.*

3. The Program Study of Automata for Waterloo-Like Languages

Before describing the obtained results, let us describe the objects used and give the fragments of the program code that allow us to obtain them. An initial automaton (a canonical automaton for a regular Waterloo-like language) is defined by means of its text description contained in the Waterloo.txt file (the description corresponds to [4; **Table 13**]), **Figure 1**.

This is a deterministic automaton whose canonicalization leads to an automaton of the same kind. The canonical automaton for a mirror automaton, after renaming the states, takes the form given in [4; **Table 16**], **Figure 2**.

The States column gives the sets of states before their reassignment; each state also corresponds to the set of states of the original mirror automaton obtained as a result of its determinization. On the basis of the obtained canonical automata, a matrix of the relation # is constructed, the rows of which correspond to the states of the canonical automaton **w**, and the columns to the states of the automaton **w2**, which is canonical to the mirror automaton. This matrix corresponds to [4; **Table 17**], **Figure 3**.

Next, a complete set of 14 grids is defined for the found relation #. After a small change in the order of their order, we obtain a set that is completely identical with the one described in [4; p. 107], **Figure 4**.

```
NFA w = NFA.FromFile("Waterloo.txt");
%% Waterloo
        a   b
==> A   E   -
    B   F   -
    C   G   B
    D   C   H
<== E   C   H
<== F   B   D
    G   A   D
    H   A   -
```

Figure 1. A canonical automaton for a regular Waterloo-like language.

```
NFA w2 = w.CurrentMirrorCanonicalDFA
  .GetRenamedStates(i => names[i]);
        a   b   % States
==> x   Y   -   % E-F
<== Y   Z   U   % A-B
    Z   V   W   % F-G-H
    U   W   -   % C
    V   P   U   % B-C
    W   Q   R   % D-E
    P   Y   R   % D-E-F
<== Q   S   -   % A
    R   V   -   % F-G
    S   U   W   % G-H
```

Figure 2. The canonical automaton for a mirror automaton.

On the basis of the obtained complete set of grids, a complete automaton is constructed, the representation of which coincides with the one given in [4; **Table 20**], **Figure 5**.

The constructed complete automaton is equivalent to the original Waterloo automaton, which can be shown by performing its determinization, see **Figure 6**. After renaming the states we get an automaton whose representation is the same as the original Waterloo automaton, **Figure 7**. Then the minimal covering set of grids is defined. This set contains grids 1, 3, 5, 6, 8, 10, 12 of the full set, see

```
SharpRelation sharpRelation = w.CurrentSharpRelation;
%% NFA: Waterloo
   x  Y  Z  U  V  W  P  Q  R  S
A  -  #  -  -  -  -  -  #  -  -
B  -  #  -  -  #  -  -  -  -  -
C  -  -  -  #  #  -  -  -  -  -
D  -  -  -  -  -  #  #  -  -  -
E  #  -  -  -  -  #  #  -  -  -
F  #  -  #  -  -  -  #  -  #  -
G  -  -  #  -  -  -  -  -  #  #
H  -  -  #  -  -  -  -  -  -  #
```

Figure 3. A matrix of the relation #.

```
List<Grid> completeGrids =
  sharpRelation.GetCompleteGrids().ToList();
Complete Grids
{ A } × { Y, Q }  % 1
{ A, B } × { Y }  % 2
{ B } × { Y, V }  % 3
{ B, C } × { V }  % 4
{ C } × { U, V }  % 5
{ D, E } × { W, P }  % 6
{ E } × { X, W, P }  % 7
{ E, F } × { X, P }  % 8
{ F } × { X, Z, P, R }  % 9
{ F, G } × { Z, R }  % 10
{ G } × { Z, R, S }  % 11
{ G, H } × { Z, S }  % 12
{ F, G, H } × { Z }  % 13
{ D, E, F } × { P }  % 14
```

Figure 4. A complete set of grids.

```
NFA com = w.GetCOM(completeGrids, "COM");
%% COM
          a              b        % Grids
==> 1     6,7,8,14       -        % { A } × { Y, Q }
==> 2     8,14           -        % { A, B } × { Y }
    3     8,9,10,13,14   -        % { B } × { Y, V }
    4     10,13          -        % { B, C } × { V }
    5     10,11,12,13    2,3,4    % { C } × { U, V }
    6     4,5            12,13    % { D, E } × { W, P }
<== 7     4,5            12,13    % { E } × { X, W, P }
<== 8     4              -        % { E, F } × { X, P }
<== 9     2,3,4          6,14     % { F } × { X, Z, P, R }
    10    2              6,14     % { F, G } × { Z, R }
    11    1,2            6,14     % { G } × { Z, R, S }
    12    1,2            -        % { G, H } × { Z, S }
    13    2              -        % { F, G, H } × { Z }
    14    4              -        % { D, E, F } × { P }
```

Figure 5. A complete automaton.

Figure 8. A covering automaton is constructed based on the minimal covering set of grids, **Figure 9**. After the canonization procedure, the resulting covering automaton takes the form shown in **Figure 10**. By renaming the states, similar to what was done for the deterministic representation of the full automaton, we obtain an automaton that is *not equivalent* to the original Waterloo automaton (the line with differences is underlined in the automaton representation), **Figure 11**. So, the covering automaton based on the minimal covering set of grids is not equivalent to the original one.

```
NFA w3 = com.GetCanonicalDFA();

                        a               b
  ==> 1-2               6-7-8-14        -
  <== 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
  <== 8-9-10-13-14      2-3-4           6-14
```

Figure 6. A canonical automaton for a complete automaton.

```
NFA w3a = w3.GetRenamedStates(i => names3[i])
  .GetOrderedStates();

          a   b   % States
  ==> A   E   -   % 1-2
      B   F   -   % 2-3-4
      C   G   B   % 4-5
      D   C   H   % 6-14
  <== E   C   H   % 6-7-8-14
  <== F   B   D   % 8-9-10-13-14
      G   A   D   % 10-11-12-13
      H   A   -   % 12-13
```

Figure 7. A canonical automaton with renamed states.

```
List<Grid> minGrids =
  sharpRelation.GetMinGridCovers().First();

Min Grids
{ A } × { Y, Q }  % 1
{ B } × { Y, V }  % 3
{ C } × { U, V }  % 5
{ D, E } × { W, P }  % 6
{ E, F } × { X, P }  % 8
{ F, G } × { Z, R }  % 10
{ G, H } × { Z, S }  % 12
```

Figure 8. The minimal covering set of grids.

```
NFA w4 = com.GetCovering(minGrids);

          a       b     % Grids
  ==> 1   6,8     -     % { A } × { Y, Q }
      3   8,10    -     % { B } × { Y, V }
      5   10,12   3     % { C } × { U, V }
      6   5       12    % { D, E } × { W, P }
  <== 8   -       -     % { E, F } × { X, P }
      10  -       6     % { F, G } × { Z, R }
      12  1       -     % { G, H } × { Z, S }
```

Figure 9. A covering automaton.

4. Computation Results for Covering Automata

In the final part of the article we describe results obtained when analyzing other covering automata that can be constructed on the basis of the minimal covering grid set complemented with some other grids from the initial complete grid set.

As will be seen later, it would be reasonable to change a little the order of additional grids, moving grid 9 of the complete set of grids (see **Figure 4**) to the beginning of the set of additional grids, **Figure 12**.

Each covering automaton will be denoted by a 7-digit binary number in which the digit 1 marks the additional grids included in the covering set of grids on which it is based. In particular, automaton 1111111 corresponds to the complete automaton (see **Figure 5**) based on the complete set of grids, and automaton 0000000 corresponds to the covering automaton based on the minimal covering set (see **Figures 9-11**). The order of digits corresponds to the order of additional grids from **Figure 12**.

Based on the NFALib library tools described above, a set of 128 covering automata was constructed and their equivalence was investigated. To check the

```
NFA w4a = w4.GetCanonicalDFA();

                a       b
==> 1           6-8     -
<== 6-8         5       12
    5           10-12   3
    12          1       -
    10-12       1       6
    3           8-10    -
    6           5       12
<== 8-10        -       6
```

Figure 10. A covering automaton after canonization.

```
NFA w4b = w4a.GetRenamedStates(i => names1[i])
  .GetOrderedStates();

        a   b   % States
==> A   E   -   % 1
    B   F   -   % 3
    C   G   B   % 5
    D   C   H   % 6
<== E   C   H   % 6-8
<== F   -   D   % 8-10
    G   A   D   % 10-12
    H   A   -   % 12
```

Figure 11. A covering automaton after canonization and renaming the states.

```
Additional Grids
{ F } × { X, Z, P, R }  % 9
{ A, B } × { Y }  % 2
{ B, C } × { V }  % 4
{ E } × { X, W, P }  % 7
{ G } × { Z, R, S }  % 11
{ F, G, H } × { Z }  % 13
{ D, E, F } × { P }  % 14
```

Figure 12. The set of additional grids.

equivalence of two automata we used an algorithm in which

- initially, the automata are reduced to a canonical form,
- then, for each state of automata, some hash characteristic is constructed,
- and, if the hash-characteristics coincide, comparison of transition functions for different combinations of new names of the second automaton is performed using the back-tracking algorithm.

As a result, we obtained four sets of pairwise equivalent automata whose properties are shown in **Table 1**.

In addition, situations were investigated in which the addition of some additional grid results in a transition from an automaton that is not equivalent to the original Waterloo automaton to an equivalent automaton. In total, there are 80 transitions, and for each of the 48 non-equivalent automata there are one or two such transitions:

- one transition for automata from set A,
- two transitions for automata from sets B and C.

The results can be represented graphically: depict a 7-dimensional hypercube, highlighting in light red the vertices that correspond to covering automata which are not equivalent to the original Waterloo automaton, and also the edges that correspond to the transition from non-equivalent to equivalent automata (**Figure 13**). In this Figure,

- the outer "circle" contains the first half of vertices (0000000 to 0111111) traversed clockwise from the rightmost vertex,
- and inner "circle" contains the second half of vertices (from 1000000 to 1111111) traversed in the same order.

Even more illustrative representation of the sets of automata and transitions between them can be obtained if we represent each of the sets A, B, C as a 4-dimensional hypercube and set D as five 4-dimensional hypercubes, and then combine these hypercubes into one 7-dimensional hypercube (**Figure 14**).

Figure 14 shows that the vertices of 7-dimensional hypercube are decomposed into equivalence classes of 16 elements each, the first 3 binary digits of which coincide. The properties of automata from each of these equivalence classes are the same, so to fully illustrate our results it is sufficient to use a three-dimensional

Table 1. Four sets of pairwise equivalent automata.

Set ID	Set characteristics		
	Numbers of automata included in the set	The total amount of automata in the set	Are automata equivalent to the original Waterloo automaton?
A	0000000 - 0001111	16	no
B	0010000 - 0011111	16	no
C	0100000 - 0101111	16	no
D	0110000 - 1111111	80	yes

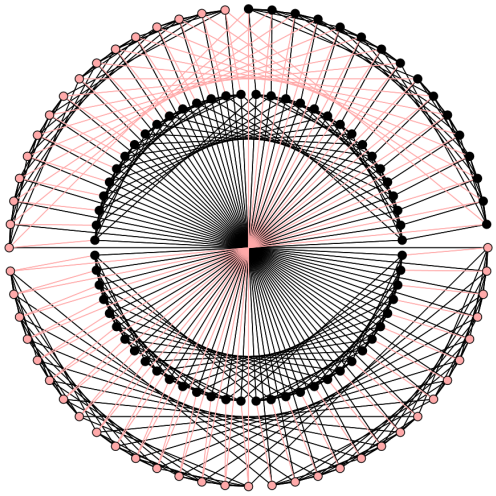

Figure 13. A set of 128 covering automata (first representation).

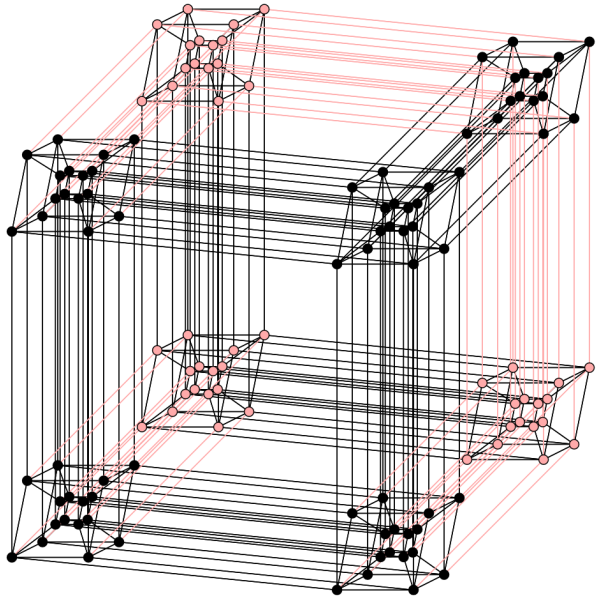

Figure 14. A set of 128 covering automata (second representation).

"factor cube", in which vertices are determined by the first three binary digits or, which is the same, by the presence or absence of grids 9, 2 and 4 in the corresponding covering automaton (**Figure 15**).

It follows from these results that a minimal covering automaton equivalent to the Waterloo automaton can be obtained by adding an additional grid 9 to the minimal covering grid set, *i.e.* by passing from automaton 0000000 to automaton 1,000,000.

Additional calculations show that, besides the minimal covering automaton 1,000,000, it is possible to obtain 4 more minimal covering automata equivalent to the original Waterloo automaton (of 11 different covering automata with 8 states), but to obtain each of them one or two grids from the minimal covering

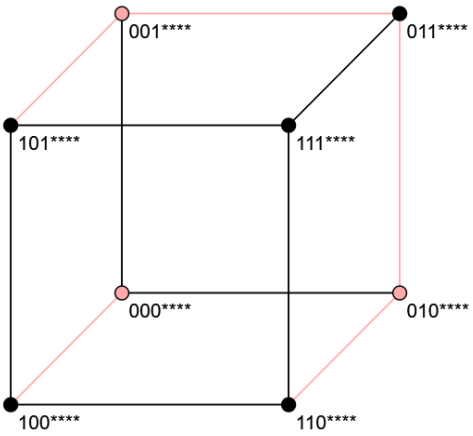

Figure 15. A set of 128 covering automata represented as a factor cube.

set must be replaced by two or, respectively, three grids from the additional set:

- grid 3 must be replaced by grids 2 and 4,
- grid 8 must be replaced by grids 7 and 9,
- grid 10 must be replaced by grids 9 and 11,
- grids 8 and 10 must be replaced by grids 7, 9, and 11.

Founding

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

Conflicts of Interest

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

References

[1] Ginsburg, S. (1966) The Mathematical Theory of Context Free Languages. McGraw-Hill Book Company, New York.

[2] Aho, A. and Ullman, J. (1973) The Theory of Parsing, Translation, and Compiling. Vol. 1, Prentice Hall, Upper Saddle River.

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

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

[5] Melnikov, B. and Melnikova, A. (2002) Some Properties of the Basis Finite Automaton. *Informatica* (*Lithuanian Academy of Sciences*), **13**, 299-310. https://doi.org/10.1007/BF03012345

[6] Melnikov, B. and Melnikova, A. (2002) Some Properties of the Basis Finite Automaton. *The Korean Journal of Computational and Applied Mathematics*, **9**, 135-150. https://doi.org/10.1007/BF03012345

[7] Dolgov, V., Melnikov, B. and Melnikova, A. (2016) The Loops of the Basis Finite Automaton and the Connected Questions. *Bulletin of the Voronezh State University. Series*: *Physics. Mathematics,* **4**, 95-111. (In Russian)

[8] Polák, L. (2005) Minimalizations of NFA Using the Universal Automaton. *International Journal of Foundation of Computer Science*, **16**, 999-1010. https://doi.org/10.1142/S0129054105003431

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

[10] Zubova, M.A. and Melnikov, B.F. (2012) On Algorithm of Constructing Conway's Universal Automaton. *Bulletin of the Voronezh State University. Series*: *Physics*. *Mathematics*, **1**, 135-137. (In Russian)

[11] Dolgov, V.N. and Melnikov, B.F. (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)

[12] Melnikov, B. and Melnikova, A. (2001) Edge-Minimization of Nondeterministic Finite Automata. *The Korean Journal of Computational and Applied Mathematics*, **8**, 469-479. https://doi.org/10.1007/BF02941980

[13] Melnikov, B. and Melnikova, A. (2011) Multi Aspect Minimization of Nondeterministic Finite Automata (Part I. Auxiliary Facts and Algorithms). *University Proceedings*. *Volga Region. Physical and Mathematical Sciences*, **4**, 59-69. (In Russian)

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

[15] Melnikov, B. and Sciarini-Guryanova, N. (2002) Possible Edges of a Finite Automaton Defining a Given Regular Language. *The Korean Journal of Computational and Applied Mathematics*, **9**, 475-485. https://doi.org/10.1007/BF03021555

[16] Dolgov, V.N. and Melnikov, B.F. (2014) On Algorithms of Automatic Construction of Waterloo-Like Finite Automata on the Basis of Full Automata. *Heuristic Algorithms and Distributed Calculations*, **1**, 24-45. (In Russian)

[17] Melnikov, B. (2022) Petal (Semi-Flower) Finite Automata: Basic Definitions, Examples and Their Relation to Complete Automata. Part I. *International Journal of Open Information Technologies*, **10**, 1-11. (In Russian)

[18] Melnikov, B. (2022) Petal (Semi-Flower) Finite Automata: Basic Definitions, Examples and Their Relation to Complete Automata. Part II. *International Journal of Open Information Technologies*, **10**, 1-10. (In Russian)

[19] Melnikov, B. (2017) The Complete Finite Automaton. *International Journal of Open Information Technologies*, **5**, 9-17. https://doi.org/10.12816/0048700

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

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

[22] Abramyan, M., Melnikov, B. and Melnikova, E. (2019) Finite Automata Table: A Science Project for High School Students. *Computer Tools in Education Journal*, **2**, 87-107. (In Russian) https://doi.org/10.32603/2071-2340-2019-2-87-107

[23] Melnikov, B. (1995) Subclasses of the Class of Context-Free Languages. MSU Publ., Moscow. (In Russian)

[24] Brosalina, A. and Melnikov, B. (2000) Commutation in Global Supermonoid of Free Monoid. *Informatica* (*Lithuanian Academy of Sciences*), **11**, 353-370.

[25] Hromkovič, J. (2004) Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Springer, Berlin.

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

[27] Abramyan, M. (2021) Computing the Weight of Subtasks in State Minimization of Nondeterministic Finite Automata by the Branch and Bound Method. *University Proceedings*, *Volga Region*. *Physical and Mathematical Sciences*, **2**, 46-52. (In Russian) https://doi.org/10.21685/2072-3040-2021-2-4