The complete finite automaton

B. F. Melnikov

***Abstract*—There is well-known, that for the description of a regular language, there are different complete invariants: not only well-known canonical automata, but also basis automata and universal automata. While constructing basis and universal automata, there is necessary to construct canonical automata for both a given regular language and its mirror image. In the process of such a construction, we get, among other objects, a special binary relation $\#$, defined on the pairs of states of these two canonical automata. This relation is also an invariant (the incomplete one) for the given regular language. For each such binary relation, there is an entire subclass of the class of regular languages, that possesses it. Therefore, on the set of all regular languages, there is possible to define an (another) binary relation; it holds for some two languages, if and only if they have the same binary the relation $\#$. It is obvious, that the binary relation defined in this way is the equivalence relation on the set of all regular languages. The question arises of the "most typical" language which is the element of such class equivalence with respect to the last relation. In this paper, we describe languages that can be considered as "typical elements", construct canonical finite automata for such languages, consider some of their properties. The main of these properties is the following: from such an automaton, using special transformations, we can obtain any canonical automaton whose language corresponds to the given binary relation $\#$.**

***Keywords*—regular languages, nondeterministic finite automata, invariants.**

I. Introduction

There is well-known, that for the description of a regular language, there are different invariants: not only well-known canonical automata, but also basis automata ([1], [2], [3] etc.) and universal automata ([4], [5] etc.). More precisely, each of these invariants could be called a *complete invariant*.

In considering both the basis and universal automata (in our terminology, the last automaton for a given regular language L is more accurately called the automaton $\mathcal{COM}(L)$), we need to construct canonical automata both for the given L and for its mirror image L^R. In the process of such a construction, we get, among other things, a special binary relation given on the state pairs of these two canonical automata; according to our terminology, this is the relation $\#$. Of course, this relation is also an invariant of the regular language (although, as elementary it is shown, it is *not* its complete invariant).

From the latter it follows that for every binary relation $\#$ (which is subject to some limitations, which, however, for this paper is unprincipled), there exists a whole subclass of the class of regular languages with this relation $\#$. In addition, *on the set of all regular languages* one can define a binary relation (let $\mathcal{R}$) that holds for some two languages if and only if (for some re-designation of the states of two finite automata for these languages) they get the same binary relation $\#$; it is obvious that the binary relation $\mathcal{R}$ defined in this way on the set of all regular languages is an equivalence relation. Thus, the question arises of the "most typical" language, which could be called an representative of the equivalence class with respect to $\mathcal{R}$.

In this paper, we describe languages that can be considered similar to the "typical representatives" of such equivalence classes. For this thing, we construct by it a canonical finite automaton depending on the given binary relation $\#$ (or according to a given defining table); in other words, we construct the regular language depending on it.

It is such an automaton (the *complete* one, denoted here by $K^{\#}$) is the main object of consideration in the paper. We consider detailed examples, some properties of the automaton $K^{\#}$ and the language $L^{\#}$ defined by it, and also show, how starting by $K^{\#}$, there is possible to obtain *any* canonical automaton whose language corresponds to a given table of the binary relation $\#$ by special transformations; these transformations include letter renaming, than they can not be called equivalent.

The structure of this paper is as follows. Section II briefly describes the notation and some facts from our previous publications on related topics. In Section III we consider in details, on a meaningful example, the work of algorithms for constructing auxiliary objects used in the paper. In Section IV we consider the formal definition of the complete automaton, and in Section V we describe a detailed example of its construction.

In the Sections VI and VII we present one of the main results of this paper, i.e., the possibility of constructing *any* finite automaton from the corresponding *a priory given* binary relation $\#$ by applying special operations to automaton $K^{\#}$. Namely, in Section VI the example of transformations is informally considered, and further in Section VII a formal description of the actions is given and the correctness of the described algorithm is shown. This description of the construction process shows the connection between canonical, basis and complete automata.

In Conclusion we formulate the main results of this article and the directions for further research.

II. Preliminaries

This section briefly describes the notation and some facts from our previous publications on related topics, see [2], [5], [6]. Let

$$K = (\,Q, \Sigma, \delta, S, F\,) \qquad (1)$$

be some finite automaton (nondeterministic Rabin-Scott's automaton), defining regular language $L = \mathcal{L}(K)$. Q is the set of states, $S \subseteq Q$ and $F \subseteq Q$ are sets of initial and final states respectively. We shall consider transition function δ of automaton (1) as

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

Received September 14, 2017.

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

but not as
$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)\,,$$
where the notation $\mathcal{P}(Q)$ denotes the superset (the power set) of the set Q; thus, we shall consider automaton without ε-transitions. We will usually write some edge $\delta(q,a) \ni r$ in the form $q \xrightarrow[\delta]{a} r$, or, if it does not cause discrepancies, simply in the form $q \xrightarrow{a} r$.

The mirror automaton for the automaton given in (1), i.e.,
$$(Q, \Sigma, \delta^R, F, S),$$
where
$$q' \xrightarrow[\delta^R]{a} q'' \quad \text{if and only if} \quad q'' \xrightarrow[\delta]{a} q',$$
will be denoted by K^R; note that K^R defines the language L^R.

Further in this section, we shall consider the regular language L to be given and use the notations defined in the cited papers for it. For the considered language L, its automaton of canonical form will be denoted by $\widetilde{L}$. Let automata $\widetilde{L}$ and $\widetilde{L^R}$ for the given language L be as follows:
$$\widetilde{L} = (Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi)$$
and
$$\widetilde{L^R} = (Q_\rho, \Sigma, \delta_\rho, \{s_\rho\}, F_\rho).$$
Moreover, we do not consider the language $L = \emptyset$, so both these automata *do have* initial states. [1]

Let us recall definitions of binary relation $\#$ and state-marking functions φ^{in} and φ^{out}, see for details [2]. Relation $\# \subseteq Q_\pi \times Q_\rho$ is defined for pairs of states of automata $\widetilde{L}$ and $\widetilde{L^R}$ in the following way: $A \# X$ if and only if
$$\big(\exists uv \in L\big)\,\big(u \in \mathcal{L}^{in}_{\widetilde{L}}(A)\,,\; v^R \in \mathcal{L}^{in}_{\widetilde{L^R}}(X)\big)\,.$$
Note that such a definition is non-constructive; however, for example, [2] contains also its equivalent constructive variant (i.e., the definition-algorithm).

State-marking function $\varphi^{in}_K : Q \to \mathcal{P}(Q_\pi)$ is defined in the following way:
$$\varphi^{in}_K(q) \ni \widetilde{q} \quad \text{if and only if} \quad \mathcal{L}^{in}_K(q) \cap \mathcal{L}^{in}_{\widetilde{L}}(\widetilde{q}) \neq \emptyset\,.$$
And 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 $\widetilde{L^R}$.

The definition for basis automaton for the given regular language L could be found also in [2]. In this paper, it will be defined by
$$\mathcal{BA}(L) = \big(\hat{Q}, \Sigma, \hat{\delta}, \hat{S}, \hat{F}\,\big).$$

Binary relation $\#$ defined in the above manner forms also the set of so-called *pseudo-grids* (see [5], etc.): namely, each of them is a pair (P,R) (where $P \subseteq Q_\pi$ and $R \subseteq Q_\rho$), such that for each pair of states $p \in P$ and $r \in R$ condition $p\#r$ holds. Each of such pseudo-grids corresponds to the state of any particular automaton for the given language. Moreover, the necessary condition for defining the given language by a finite automaton is that the subset of pseudo-grids corresponding to the set of states of considered automaton cover all the items of the relation $\#$.

And if for some pseudo-grid (P,R) we can not extend neither set P nor the set R in order to not violate the definition of a pseudo-grid, then we call such a pseudo-grid by a *grid*.

Examples of the objects considered here were considered in detail in the works cited above. And it is important to note that all the definitions are *constructive* [2], i.e., they forms *algorithms* for constructing such objects.

On the blocks considered by the states, we define automaton $\mathcal{COM}(L)$; its definition, examples and some related concepts can be found in [5]. [3]

For the future, there is very important the condition for the existence of an edge (a transition) $\mathcal{B}_1 \xrightarrow[\delta_Q]{a} \mathcal{B}_2$ (transition from $\mathcal{B}_1$ into $\mathcal{B}_2$ in automaton $\mathcal{COM}(L)$, labeled by letter $a \in \Sigma$; see also [7, Def. 2]):
$$\begin{array}{l}\big(\forall p \in \alpha(\mathcal{B}_1)\big)\,\big(\delta_\pi(p,a) \in \alpha(\mathcal{B}_2)\big) \quad \& \\ \big(\forall r \in \beta(\mathcal{B}_2)\big)\,\big(\delta_\rho(r,a) \in \beta(\mathcal{B}_1)\big).\end{array}$$
(On the basis of these conditions, all the transitions of the automaton $\mathcal{COM}(L)$ are made. Conditions for input and output states for this article are less important; if necessary, see them in the cited works.)

Thus, we can assume that by considering some given regular language L, we simultaneously introduce the notation for the related language:

- two canonical automata (i.e., $\widetilde{L}$ and $\widetilde{L^R}$), and also their states, their transition functions etc.;
- binary relation $\#$ defined on pairs of states of automata $\widetilde{L}$ and $\widetilde{L^R}$;
- state-marking functions φ^{in} and φ^{out};
- equivalent basis automaton $\mathcal{BA}(L)$;
- equivalent automaton $\mathcal{COM}(L)$.

III. The detailed consideration of an example of constructing auxiliary objects

Thus, we continue our constructions using the example of a language and the corresponding automata already considered in [2], [5]. [4] Let us briefly repeat the automata of these papers.

Let the given automaton (K, for regular language L) be given on Fig. 1, and the mirror automaton (K^R, for language L^R) be given on Fig. 2.

[1] Like [2], we call by canonical automaton a deterministic automaton, containing the minimum possible number of states. In this case, it is also like [2], we do *not* require the everywhere-defining of this automaton, and, therefore, do not consider the possible "dead state".

[2] Including, as we have already noted, there exists a constructive definition of binary relation $\#$.

[3] In fact, we actually described this automaton for the first time long before, in [7], but in that paper we did not use the notation given here. As we already noted, we subsequently proved that the automaton $\mathcal{COM}(L)$ coincides with the so-called Conway's universal automaton, [4], [5], [8] etc.

[4] This language was first obtained in 1996 using a *search algorithm*, see conference abstracts [9]. The algorithm looked for a language that is minimal in some parameters, which can be defined over a 2-letter alphabet using some automaton with at most 3 states. In this case, the edge-minimization algorithm of nondeterministic finite automata was used as an auxiliary, which we later described in [2]. But back in 1996 the program, implemented on the basis of this algorithm, received a solution on the *available at that time* computer technology.

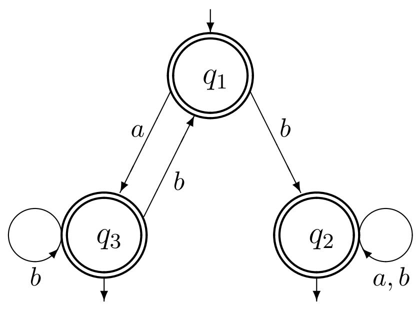

Fig. 1

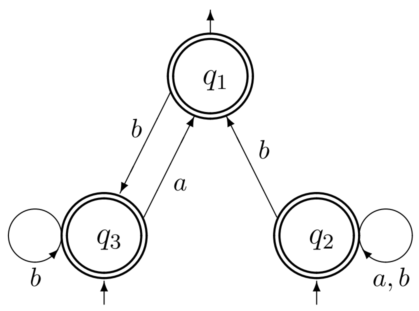

Fig. 2

Corresponding canonical automata $\widetilde{L}$ and $\widetilde{L^R}$ are given on Fig. 3 and 4:

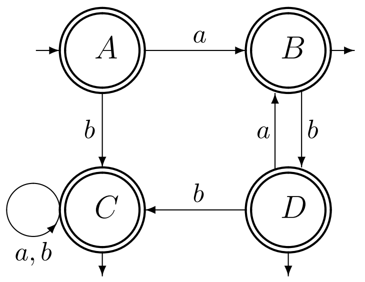

Fig. 3

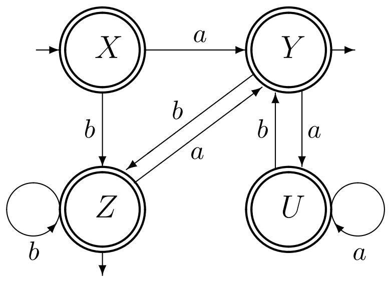

Fig. 4

Binary relation $\#$ for the given language, also obtained in [2], is given on Tab. 1:

Tab. 1

$\#$	X	Y	Z	U
A	–	$\#$	$\#$	–
B	$\#$	–	$\#$	–
C	$\#$	$\#$	$\#$	$\#$
D	$\#$	$\#$	$\#$	–

In [2], [5], we did not consider basis automaton $\mathcal{BA}(L)$ for this language; its transition function, which is also constructed by [2], is given on Tab. 2. (For convenience, we write down the state $\begin{smallmatrix} A \\ X \end{smallmatrix}$ by $A\#X$, etc. Besides, we do *not* write notation for sets, i.e., for example, instead of $\{ \begin{smallmatrix} A \\ X \end{smallmatrix}, \begin{smallmatrix} A \\ Y \end{smallmatrix} \}$, we write $A\#X, A\#Y$.)

Tab. 2

	$\mathcal{BA}(L)$	a	b
$\rightarrow$	$A\#Y$	$B\#Z, B\#X$	$C\#U$
$\rightarrow$	$A\#Z$	–	$C\#Y, C\#Z, C\#X$
$\leftarrow$	$B\#X$	–	–
	$B\#Z$	–	$D\#Y, D\#Z, D\#X$
	$D\#Y$	$B\#Z, B\#X$	$C\#U$
$\leftarrow$	$C\#X$	–	–
	$C\#Y$	$C\#Z, C\#X$	$C\#U$
	$C\#Z$	–	$C\#Y, C\#Z, C\#X$
	$C\#U$	$C\#Y, C\#U$	–
$\leftarrow$	$D\#X$	–	–
	$D\#Z$	–	$C\#Y, C\#Z, C\#X$

In our previous papers cited before [2], [3], [5], we also did not consider further actions with the basis automaton. As such further actions, we first select (all) 5 possible grids based on Tab. 1:

$$\begin{aligned} &(1)\ \ \{A, C, D\} \times \{Y, Z\}, \\ &(2)\ \ \{A, B, C, D\} \times \{Z\}, \\ &(3)\ \ \{B, C, D\} \times \{X, Z\}, \\ &(4)\ \ \{C\} \times \{X, Y, Z, U\}, \\ &(5)\ \ \{C, D\} \times \{X, Y, Z\}. \end{aligned}$$

We shall use *the same numbers* also in the future, in particular, to denote the states of new automata. Thus, using definitions of Section II and of automaton $\mathcal{BA}(L)$, we obtain automaton $\mathcal{COM}(L)$, see Tab. 3:

Tab. 3

	$\mathcal{COM}(L)$	a	b
$\rightarrow$	1	$2, 3$	$1, 2, 3, 4, 5$
$\rightarrow$	2	–	$1, 2, 3, 5$
$\leftarrow$	3	–	$1, 2, 3, 5$
$\leftarrow$	4	$1, 2, 3, 4, 5$	$1, 2, 3, 4, 5$
$\leftarrow$	5	$2, 3$	$1, 2, 3, 4, 5$

Terminology and constructions related to the so-called covering automata [5] will not be used in this paper; however, we propose to use them in the continuation paper, therefore, we describe briefly such constructions. Thus, it is easy to show that in the automaton $\mathcal{COM}(L)$, there is (and the only possible one) a covering subset consisting of exactly 3 grids (namely, blocks 1, 3 and 4), and covering subsets consisting of 2 grids does not exist. Selecting this subset (ie removing the blocks 2 and 5 that are not included in it), we obtain a covering automaton, given in Fig. 5; it is easy to verify that this automaton is equivalent to the original one.

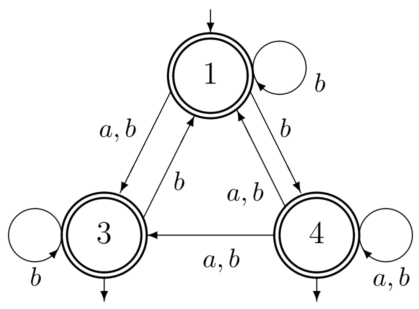

Fig. 5

Thus, the example above gives an equivalent automaton, which, as can be easily verified, has the minimum possible number of states. However, note that in the general case such a fact is incorrect (see [8]; for the first time the example was obtained in [10]), i.e., the covered automaton *is not necessarily equivalent* to the given one; we assume to consider in the continuation paper a detailed analysis of the corresponding example *in our terminology*, as well as its relation to the objects defined in this paper.

IV. The formal definition of the complete automaton

In [3], we defined an auxiliary object, designated there as an automaton $K^{\#}$; we shall use the same notation below. The possible use of this automaton in applied problems (specifically, in the problem of state-minimization of nondeterministic finite automata) was briefly shown in [11]. However, in both of these papers, there were no detailed definitions or examples (except either trivial, or, on the contrary, very complex); a detailed definition and a more informative explanatory example are given in this paper. So, as already noted in Introduction, it is the automaton $K^{\#}$ that is our main object of consideration.

For the following, the next statement is important, see [3].

Proposition 1: Let binary relation $\#$ be defined on sets A_π and A_ρ (where $|A_\pi| = m > 0$, $|A_\rho| = n > 0$), i.e., $\# \subseteq A_\pi \times A_\rho$. Let also the following limitations hold:

- $(\forall p \in A_\pi)\,(\exists r \in A_\rho)\,(p\#r)$
  (i.e., the table of binary relation $\#$ has no empty rows);
- $(\forall r \in A_\rho)\,(\exists p \in A_\pi)\,(p\#r)$
  (i.e., the table has no empty columns);
- $(\forall p_1, p_2 \in A_\pi,\, p_1 \neq p_2)\,(\exists r \in A_\rho)\,\big((p_1\#r \,\&\, \overline{p_2\#r}) \vee (\overline{p_1\#r} \,\&\, p_2\#r)\big)$
  (i.e., the table has no identical rows);
- $(\forall r_1, r_2 \in A_\rho,\, r_1 \neq r_2)\,(\exists p \in A_\pi)\,\big((p\#r_1 \,\&\, \overline{p\#r_2}) \vee (\overline{p\#r_1} \,\&\, p\#r_2)\big)$
  (i.e., the table has no identical columns);

Then there exists a regular language for which the corresponding binary relation $\#$ coincides with the given one. $\square$

Thus, we can assume that for the regular language L chosen on the basis of the last statement, and for the standard notation of Section II associated with any regular language, the following holds: $Q_\pi = A_\pi$ and $Q_\rho = A_\rho$. Besides, Proposition 1 simply entails following Proposition 2.

Proposition 2: In the terms of Proposition 1, the following limitations hold:

- $m \leq 2^n - 1$;
- $n \leq 2^m - 1$. $\square$

Next we consider an alphabet whose number of letters equals to $|A_\pi| \cdot |A_\rho| = mn$. (We note at once that the number of states of the complete automaton considered below is equal to $|A_\pi| = m$. Therefore, in practical problems, for example, in the above-mentioned problems of minimizing nondeterministic finite automata, we can choose whether we consider a regular language corresponding to a given table of a binary relation $\#$ or a language corresponding to the transposed version of this table. [5]. In practical problems, we usually want a smaller number of vertices of the automaton in question, i.e., if necessary, we can assume that $m \leq n$.) Thus, let the considered alphabet be the following:

$$\Sigma_{\#} = \Big\{ a_{\substack{A \\ X}} \;\Big|\; A \in Q_\pi,\, X \in Q_\rho \Big\}.$$

We choose an arbitrary string (let s_π) and an arbitrary column (let s_ρ) for the given table of the binary relation $\#$. The meaning of this choice is the input states of two canonical automata, they are $\widetilde{L}$ and $\widetilde{L^R}$ in our previous notation. In advance, we note that the *arbitrarily chosen* starting state of $\widetilde{L^R}$ uniquely determines the *nonempty set* of final states $\widetilde{L}$, and vice versa.

Definition 1: For the given binary relation $\# \subseteq Q_\pi \times Q_\rho$ and the given states $s_\pi \in Q_\pi$ and $s_\rho \in Q_\rho$, let us consider automaton

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

(or, briefly, $K^{\#}$, if s_π are s_ρ are mentioned), where:

- $F_\pi = \{\, f_\pi \in Q_\pi \mid f_\pi \# s_\rho \,\}$; [6]
- transition function $\delta^{\#}$ is defined in the following way:

$$\delta^{\#}(A, a_{\substack{B \\ X}}) = \begin{cases} \{B\}, & \text{if A\#X;} \\ \emptyset, & \text{otherwise.} \end{cases}$$

(we allow here the possibility $A = B$). $\square$

For this automaton, we give the following statement, which can be considered as a simplified formulation of [3, Prop. 14].

Proposition 3: Let binary relation $\widetilde{\#} \subseteq A_\pi \times A_\rho$ be given. For it, let us consider automaton $K^{\#}$; we will use the considered in Section II designations and auxiliary objects for this automaton. For language of this automaton $\mathcal{L}(K^{\#})$, let us considered corresponding binary relation $\# \subseteq Q_\pi \times Q_\rho$. Then the built for its language $\mathcal{L}(K^{\#})$ binary relation $\#$ coincides with the given relation $\widetilde{\#}$ (up to re-designation of elements of sets A_π and A_ρ for elements of sets Q_π and Q_ρ). $\square$

Note that in fact in this section we have described automata that are not common for one given regular language (although the table of the binary relation $\#$ can be constructed from a given regular language, and this is usually done), but *for an entire class of languages*; the common objects for all these languages are the binary relation $\#$, and also the input and output states of the corresponding canonical automata (since, as follows from the foregoing, at the input state of automaton $\widetilde{L^R}$, the outputs of the automaton $\widetilde{L}$ are uniquely determined). Below in Section VI, it will be shown how to obtain *any automaton having a given table of binary relation* $\#$ (from the complete automaton, using special transformations and fixed input and output states).

V. The detailed example of the complete automaton

As we noted before, in [3] we considered the trivial example of the complete automaton only (and the title "complete" was not used), i.e., for $m = n = 2$ and $|\#| = 3$. In this section, we will consider a much more informative example, continuing to perform the constructions for the language and the corresponding automata already considered

[5] In [3], the automaton for this language was denoted by $K_{\# s_\pi s_\rho}$ (or, simply, $K_{\#}$), unlike considered here "more important" automaton $K^{\# s_\pi s_\rho}$. In this paper, we shall not use $K_{\#}$.

[6] The choice of such a *nonempty* set is possible due to the above limitations. Also note that by choosing different $s_\pi \in Q_\pi$ and $s_\rho \in Q_\rho$, we obtain a *set* of languages, each of which corresponds to the given relation $\#$.

in Section III above (i.e., for $m = n = 4$ and $|\#| = 11$). In fact, “the input” (for the construction of this section) are *only* binary relation $\#$ for the given language, i.e., Tab. 1 for our example.

According to the definitions given above, the corresponding alphabet $\Sigma_\#$ is the following:

$$\Sigma_\# = \Big\{ a_{\overset{A}{X}}, a_{\overset{B}{X}}, a_{\overset{C}{X}}, a_{\overset{D}{X}}, a_{\overset{A}{Y}}, a_{\overset{B}{Y}}, a_{\overset{C}{Y}}, a_{\overset{D}{Y}},$$

$$a_{\overset{A}{Z}}, a_{\overset{B}{Z}}, a_{\overset{C}{Z}}, a_{\overset{D}{Z}}, a_{\overset{A}{U}}, a_{\overset{B}{U}}, a_{\overset{C}{U}}, a_{\overset{D}{U}} \Big\}$$

(it contains 16 letters, because there exist 4 states of automaton $\widetilde{L}$ and 4 states of automaton $\widetilde{L^R}$, $4 \cdot 4 = 16$). Below, we shall write for simplicity

$$\overset{A}{X} \quad \text{instead of} \quad a_{\overset{A}{X}}, \quad \text{etc.}$$

Let $s_\pi = A$, $s_\rho = X$; then $F_\pi = \{B, C, D\}$.[7] And, according to the above definition, we obtain for the language $L^\#$ the following canonical automaton $K^\#$ (Tab. 4 and 4′):

Tab. 4

	$K^\#$	$\overset{A}{X}$	$\overset{B}{X}$	$\overset{C}{X}$	$\overset{D}{X}$	$\overset{A}{Y}$	$\overset{B}{Y}$	$\overset{C}{Y}$	$\overset{D}{Y}$
$\rightarrow$	A	–	–	–	–	A	B	C	D
$\leftarrow$	B	A	B	C	D	–	–	–	–
$\leftarrow$	C	A	B	C	D	A	B	C	D
$\leftarrow$	D	A	B	C	D	A	B	C	D

Tab. 4′

	$K^\#$	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{C}{Z}$	$\overset{D}{Z}$	$\overset{A}{U}$	$\overset{B}{U}$	$\overset{C}{U}$	$\overset{D}{U}$
$\rightarrow$	A	A	B	C	D	–	–	–	–
$\leftarrow$	B	A	B	C	D	–	–	–	–
$\leftarrow$	C	A	B	C	D	A	B	C	D
$\leftarrow$	D	A	B	C	D	–	–	–	–

For the following, we consider the construction of the binary relation $\#$ for the language defined by the last automaton; we will show by an example that this relation coincides with the given (up to re-designation of the elements of the set). The process of construction is similar to the one considered in [2]. For this thing, let us firstly consider the mirror automaton $(K^\#)^R$; this automaton[8] is given on following Tab. 5–5‴:

Tab. 5

	$(K^\#)^R$	$\overset{A}{X}$	$\overset{B}{X}$	$\overset{C}{X}$	$\overset{D}{X}$
$\leftarrow$	A	B,C,D	–	–	–
$\rightarrow$	B	–	B,C,D	–	–
$\rightarrow$	C	–	–	B,C,D	–
$\rightarrow$	D	–	–	–	B,C,D

(like similar situations before, we omit the signs of sets, i.e., we write, for instance, A, B, C, D instead of $\{A, B, C, D\}$).

Tab. 5′

	$(K^\#)^R$	$\overset{A}{Y}$	$\overset{B}{Y}$	$\overset{C}{Y}$	$\overset{D}{Y}$
$\leftarrow$	A	A,C,D	–	–	–
$\rightarrow$	B	–	A,C,D	–	–
$\rightarrow$	C	–	–	A,C,D	–
$\rightarrow$	D	–	–	–	A,C,D

Tab. 5″

	$(K^\#)^R$	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{C}{Z}$	$\overset{D}{Z}$
$\leftarrow$	A	A,B,C,D	–	–	–
$\rightarrow$	B	–	A,B,C,D	–	–
$\rightarrow$	C	–	–	A,B,C,D	–
$\rightarrow$	D	–	–	–	A,B,C,D

Tab. 5‴

	$(K^\#)^R$	$\overset{A}{U}$	$\overset{B}{U}$	$\overset{C}{U}$	$\overset{D}{U}$
$\leftarrow$	A	C	–	–	–
$\rightarrow$	B	–	C	–	–
$\rightarrow$	C	–	–	C	–
$\rightarrow$	D	–	–	–	C

The process of determinization of the last automaton is described by the following table (Table 6–6‴; we write so called “aggregate states” in the order of their appearance in the build process):

Tab. 6

	$(K^\#)^R$	$\overset{A}{X}$	$\overset{B}{X}$	$\overset{C}{X}$	$\overset{D}{X}$
$\rightarrow$	B,C,D	–	B,C,D	B,C,D	B,C,D
$\leftarrow$	A,C,D	B,C,D	–	B,C,D	B,C,D
$\leftarrow$	A,B,C,D	B,C,D	B,C,D	B,C,D	B,C,D
	C	–	–	B,C,D	–

Tab. 6′

	$(K^\#)^R$	$\overset{A}{Y}$	$\overset{B}{Y}$	$\overset{C}{Y}$	$\overset{D}{Y}$
$\rightarrow$	B,C,D	–	A,C,D	A,C,D	A,C,D
$\leftarrow$	A,C,D	A,C,D	–	A,C,D	A,C,D
$\leftarrow$	A,B,C,D	A,C,D	A,C,D	A,C,D	A,C,D
	C	–	–	A,C,D	–

Tab. 6″

	$(K^\#)^R$	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{C}{Z}$	$\overset{D}{Z}$
$\rightarrow$	B,C,D	–	A,B,C,D	A,B,C,D	A,B,C,D
$\leftarrow$	A,C,D	A,B,C,D	–	A,B,C,D	A,B,C,D
$\leftarrow$	A,B,C,D	A,B,C,D	A,B,C,D	A,B,C,D	A,B,C,D
	C	–	–	A,B,C,D	–

Tab. 6‴

	$(K^\#)^R$	$\overset{A}{U}$	$\overset{B}{U}$	$\overset{C}{U}$	$\overset{D}{U}$
$\rightarrow$	B,C,D	–	C	C	C
$\leftarrow$	A,C,D	C	–	C	C
$\leftarrow$	A,B,C,D	C	C	C	C
	C	–	–	C	–

[7] We have already noted that, according to the definitions introduced, F_π can be determined on the basis of an arbitrarily chosen s_ρ; here, we have chosen s_ρ so that the automaton constructed by us corresponds to the original one.

[8] According to the terminology and notation of [3], the canonical automaton for its language is an automaton $L_\#$ ($K_\#$), but we do not use the last notation in this paper. We only note that the automaton $\widetilde{(K^\#)^R}$ constructed below has all the properties of the automaton $K^\#$.

And replacing:

- $\{B, C, D\}$ for X (it corresponds to the following elements of binary relation $\#$: $B\#X$, $C\#X$ and $D\#X$);
- $\{A, C, D\}$ for Y (corresponds to elements $A\#Y$, $C\#Y$ and $D\#Y$);
- $\{A, B, C, D\}$ for Z (corresponds to elements $A\#Z$, $B\#Z$, $C\#Z$ and $D\#Z$);
- $\{C\}$ for U (corresponds to element $C\#U$),

we firstly form binary relation $\#$, *corresponding to considered language $\mathcal{L}(K^{\#})$*, and secondly, obtain the following automaton (Tab. 7, 7′); we should designate it, according to our notation system, as $\widetilde{(K\#)}^R$:

Tab. 7

$\widetilde{(K\#)}^R$	$^{A}_{X}$	$^{B}_{X}$	$^{C}_{X}$	$^{D}_{X}$	$^{A}_{Y}$	$^{B}_{Y}$	$^{C}_{Y}$	$^{D}_{Y}$
→ X	–	X	X	X	–	Y	Y	Y
← Y	X	–	X	X	Y	–	Y	Y
← Z	X	X	X	X	Y	Y	Y	Y
U	–	–	X	–	–	–	Y	–

Tab. 7′

$\widetilde{(K\#)}^R$	$^{A}_{Z}$	$^{B}_{Z}$	$^{C}_{Z}$	$^{D}_{Z}$	$^{A}_{U}$	$^{B}_{U}$	$^{C}_{U}$	$^{D}_{U}$
→ X	–	Z	Z	Z	–	U	U	U
← Y	Z	–	Z	Z	U	–	U	U
← Z	Z	Z	Z	Z	U	U	U	U
U	–	–	Z	–	–	–	U	–

Obviously, the formed binary relation $\#$ coincides with the originally defined one.

VI. An example of the use of complete automaton

As we already remarked in Introduction, one of the main results of this paper is the possibility of constructing a *any* finite automaton from the corresponding *pre-defined* binary relation $\#$ by applying special operations to the automaton $K^{\#}$.

We note that the process of constructing an automaton with a given binary relation, although it has general constructions with the process of constructing any automaton by a given basic automaton [12], is an entirely different problem: in our case we are dealing with *nonequivalent* (generally speaking) transformations of nondeterministic finite automata (for example, we change the alphabet in the process of construction), while in [12], as it follows from the title of that article, all the transformations were equivalent.

Such nonequivalent transformations of nondeterministic automata, which we have not considered in previous publications, are the following:

- "duplication" letters of the language of the automaton;
- forming the set of so-called "selected" edges (performed on the base of the basic automaton);
- deleting the letter of the language (with the removal of all arcs marked in the automaton);
- a special version of the combination of letters, taking into account the available "selecting";
- renaming letters of a language in the automaton.

Next, we will use the words "selected edges", "selecting" *as the terms* and write them without quotes. For a detailed description of these actions, see below: an example will be considered in this section (which is a continuation of the examples discussed above), and a formal description of the actions to be performed will be given in the next section. In the example below, we show how to obtain the automaton $\widetilde{L}$ (Tab. 4) using such several steps starting by automaton $K^{\#}$ (Fig. 3). From the example under consideration, it is clear that similar actions are possible for any language (and the corresponding canonical automaton) that has the same table of the binary relation $\#$.

Let us note in advance the following fact: we can assume that all the subsequent ones are produced *by means of a nondeterministic search algorithm, and the sequence of actions can be considered an "oracle"* (see [13] etc.). Thus, in the development of specific algorithms, it is convenient to implement the actions described below, for example, using the branch and bound method, and also, perhaps, to apply parallel programming technologies, which we described, for example, in [14] and some recent publications in Russian.

So, let us continue our consideration of the examples that have been started before. Let us repear Fig. 3, where we add the following things:

- firstly, we write down all possible states of the automaton $\widetilde{L^R}$ corresponding to the states of the automaton $\widetilde{L}$ (i.e., for some state $A \in Q_\pi$ we write all $X \in Q_\rho$, such that $A\#X$), we write them below, under the labels of the corresponding vertices of the automaton $\widetilde{L}$;
- secondly, we somehow number the edges (and write numbers from 1 to 7 next to the arcs in brackets).

In doing so, we get the following Fig. 6.

Now, let us consistently consider all the 7 edges of automaton $\widetilde{L}$. For edge (1), i.e.,

$$A \xrightarrow[\delta_\pi]{a} B\,,$$

we obtain corresponding edges for the basis automaton (Tab. 2)

$$^{A}_{Y} \xrightarrow[\hat\delta]{a} {}^{B}_{Z} \quad \text{and} \quad ^{A}_{Y} \xrightarrow[\hat\delta]{a} {}^{B}_{X}\,.$$

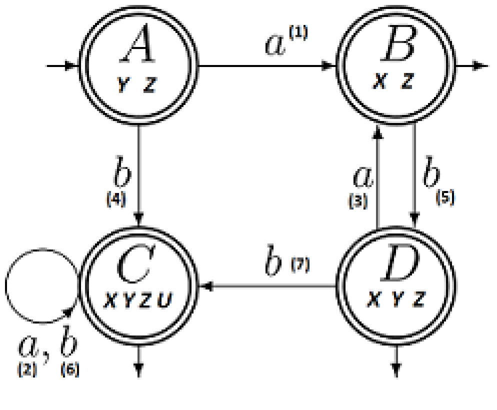

Fig. 6

By definition of automaton $K^{\#}$, both these edges of the basic automaton correspond to the edge

$$A \xrightarrow[\delta^{\#}]{{}^{B}_{Y}} B$$

(we recall that by writing ${}_{Y}^{B}$, or, which is the same thing, $a_{{}^{B}_{Y}}$, we have designated one of the letters of the alphabet over which the automaton is defined $K^{\#}$).

Let us do the same with all the remaining 6 edges of the automaton $\widetilde{L}$, and write the results in the form of such an addition (Tab. $8, 8'$) to Table 4 considered before:

Tab. 8

	$\overset{A}{X}$	$\overset{B}{X}$	$\overset{C}{X}$	$\overset{D}{X}$	$\overset{A}{Y}$	$\overset{B}{Y}$	$\overset{C}{Y}$	$\overset{D}{Y}$
→ A	–	–	–	–	A	B	C	D
← B	A	B	C	D	–	–	–	–
← C	A	B	C	D	A	B	C	D
← D	A	B	C	D	A	B	C	D
						(1) (3)	(2) (4) (6) (7)	

Tab. 8′

	$\overset{A}{Z}$	$\overset{B}{Z}$	$\overset{C}{Z}$	$\overset{D}{Z}$	$\overset{A}{U}$	$\overset{B}{U}$	$\overset{C}{U}$	$\overset{D}{U}$
→ A	A	B	C	D	–	–	–	–
← B	A	B	C	D	–	–	–	–
← C	A	B	C	D	A	B	C	D
← D	A	B	C	D	–	–	–	–
			(6) (7)	(5)			(2)	

The new elements (the addition) of the last table are:

- markings of columns in the new last line, showing which edges of automaton $K^{\#}$ correspond to this letter;
- marking (with a gray background) the cells of the table (edges of the automaton $K^{\#}$), showing which edges of the automaton $K^{\#}$ correspond to this edge.

We note the following two things.

- The correspondence between the edges of the given canonical automaton (or of the given basis automaton) and the edges of automaton $K^{\#}$ is *not* single-valued, as seen in the last table.
- When using the basic automaton, we *did not pay attention* to the values of the function φ^{out} for the output states. (Because the edges we need are determined without using them.)

Next, let us remove all the letters from the alphabet under consideration, for which the cells (i.e., edges) marked with a gray background, i.e., those corresponding to the edges of the given canonical (or basic) automaton do not exist in the corresponding columns of the last automaton (Tab. 8). In addition, every remaining letter *duplicate* in as many as it corresponds to the arcs. We obtain the automaton given on the following Table 9:

Tab. 9

	$\overset{B}{Y}$	$\overset{B}{Y}$	$\overset{C}{Y}$	$\overset{C}{Y}$	$\overset{C}{Y}$	$\overset{C}{Y}$	$\overset{C}{Z}$	$\overset{C}{Z}$	$\overset{D}{Z}$	$\overset{C}{U}$
→ A	B	B	C	C	C	C	–	–	–	C
← B	–	–	–	–	–	–	C	C	D	–
← C	B	B	C	C	C	C	C	C	D	C
← D	B	B	C	C	C	C	C	C	D	–
	(1)	(3)	(2)	(4)	(6)	(7)	(6)	(7)	(5)	(2)

(The columns are written in the same order as in the previous table 8. And, of course, each letter in the new alphabet, with duplicated letters, corresponds now to only one edge marked with a gray background. Among other things, we note in the obtained automaton the absence of transitions to the input state A, which was one of the signs of the given automaton and it was not fulfilled for the corresponding $K^{\#}$ and further automata constructed on its base.)

Next, we combine letters that are identically marked in the last line. It is necessary to fulfill this additional condition: *in the same row, the merged columns cannot have different selected transitions* (i.e., there is impossible that both cells of the same row of the merged columns are selected, and different transitions are defined); we note that in our example this situation does not arise[9]. After combining the letters, we do not need their previous notation, so we do not use the letters $\overset{B}{Y} \ldots \overset{C}{U}$ in the new table, but denote 7 letters in the same way as we denoted the edges of the automaton on Fig. 6: (1) … (7).

So, pointing "new letters" (1) … (7), as usual, in the header cell of each column and "sorting them in ascending order", we get the following automaton (Tab. 10):

Tab. 10

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
→ A	B	C	B	C	–	C	–
← B	–	–	–	–	D	C	C
← C	B	C	B	C	D	C	C
← D	B	C	B	C	D	C	C

And, as is easy to be convinced, considering in it only the selected arcs and denoting (in a natural way):

- letters (1), (2) and (3) by a;
- letters (4), (5), (6) and (7) by b,

we obtain the desired automaton.

Note that we can, *applying the same nondeterministic algorithm, get an entire class of automata* (languages), for example, using one of the following options:

- either selecting some other edges of the given automaton $K^{\#}$;
- or leaving in the last automaton (Tab. 10 in our example) some other edges (not only selected ones).

All these automata (all these languages) can be obtained as a result of the transformations described here; all of them are characterized by the same (the given) table of binary relation $\#$.

VII. Constructing arbitrary finite automaton on the base of the complete automaton

This section formally describes the nondeterministic algorithm, which in the previous section was described informally.

Definition 2: Let regular language L defines binary relation $\#$. Then we shall say, that edge

$$A \xrightarrow[\delta_\pi]{a} B \tag{2}$$

of automaton $\widetilde{L}$ corresponds to edge

$$A \xrightarrow[\delta\#]{\overset{B}{X}} B, \tag{3}$$

of automaton $K^{\#}$, if for some $Y \in Q_\rho$, the following condition holds:

$$\overset{A}{X} \xrightarrow[\hat{\delta}]{a} \overset{B}{Y}\,. \tag{4}$$

[9] We assume in the next publication to consider, among other things, a similar example.

Three edges (2), (3) and (4) (or some pair of them) will be called *corresponding*. □

Of course, according to this definition, a certain edge of canonical automaton $\widetilde{L}$ corresponds to an only edge of automaton $K^{\#}$; but the converse, in general, is not true. The same fact is true for the basic automaton (instead of the canonical one). That is, in both cases, the correspondence can be considered as a morphism acting from the set of transitions of the canonical automaton (transitions of the basic automaton) to the set of transitions of a complete automaton. Examples, when some (including 0) edges of canonical automaton correspond an only edge, were considered in the last section; in the tables, we marked them in a gray background.

Definition 3: Let edges of automaton $\widetilde{L}$ are numbered from 1 to n; we shall denote them by (1) …(n). Let edge (i), written as (2), corresponds to edge (3). Then for *the letter* (i), we shall call

$$A \xrightarrow[\delta_{(-)}]{(i)} B$$

by *modified (i)-edge*. (The constructed transition function will be denoted by $\delta_{(-)}$.)

In this case, we consider modified edges for the same states of the canonical automaton (set Q_π) over alphabet

$$\Sigma^{(-)} = \{\, (1), (2), \dots, (n) \,\} \,. \,\square$$

Thus, we obtain a *one-to-one* correspondence between the edges of the given canonical automaton and the modified edges constructed by us; all the necessary examples were also considered in the last section. Therefore, considering a given regular language L and the corresponding relation $\#$, starting with the automaton $K^{\#}$, we delete edges, each of which *does not* correspond to some edge of automaton $\widetilde{L}$, after which we transform each of the remaining letters into several new ones (each of which corresponds to the edge of canonical automaton), and, in the end, rename the letters, perhaps, by calling several letters by the same new letter. Let us formulate the described process in the form of the following proposition.

Proposition 4: For a given regular language L and the corresponding binary relation $\#$, there exists a sequence of transformations consisting of:

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,

resulting automaton $\widetilde{L}$. □

The correctness of the statement follows from the above material.

VIII. Conclusion

So, in the present paper we described languages that can be considered as "the most typical representatives" of subclasses of the class of regular languages – such subclasses, that each element of them is characterized by the same table of the binary relation $\#$. With the help of several nonequivalent transformations of canonical automata for the "typical" languages we have identified, we obtain a canonical automaton for an arbitrary regular language corresponding to a given table of the binary relation $\#$.

As already noted above, a complete automaton does not define the regular language under consideration; however, it defines a language that possesses *many important properties* of the considered one. Similarly to the above cited article [12], where we "collected" the edges of an arbitrary automaton (for a predefined regular language) from the edges of its *basic* automaton – in this paper, we roughly "collect" edges of an arbitrary automaton from the edges of its complete automaton. (In the interpretation of the algorithm above, we actually removed the edges from the complete automaton, i.e., we performed the "analytical", not the "synthetic" algorithm; however, of course, this difference is in this case unprincipled.)

The material of this article is expected to continue in the following two ways. *First*, we continue to consider the connection of complete automata with the problems of vertex minimization of nondeterministic finite automata, see [8], [14], [15] and others. And this direction includes both the theoretical part (description of new variants of minimization algorithms, proof of their correctness, etc.), and the practical part (description and implementation of heuristics for minimizing automata with a large number of states). Some *algorithmic* questions related to the implementation of the corresponding heuristic algorithms have already been considered in [11]; but, of course, this work requires the continuation, in particular, of new variants of parallel implementation of the corresponding algorithms, which could be the development of algorithms considered in [14], [15]. An indirect argument that the implementation of these algorithms will yield very good results is the following fact, briefly mentioned above: in 1996, the program for finding the minimum language by some criteria, implemented on the basis of such an algorithm, on computer technology available at that time received the necessary decision.

The problem of edge-minimization of nondeterministic finite automata adjoins the problems described here; and the description of algorithms for these problems that are more effective than the algorithms now available should become the development of both the material from [2] and the technology of the nonequivalent transformation of the automaton $K^{\#}$ described in this paper. In the author's opinion, this problem is even more important for practice than the much more studied problem of state-minimization: in practical problems, a nondeterministic automaton in the memory of computer is usually represented as a set of transition edges.

And *the second direction of the continuation of the work* on the topic is completely different. We already noted in Introduction that for every binary relation $\#$ satisfying the necessary limitations, there exists a subclass of the class of regular languages, each of which has such a relation $\#$. The binary relation on the set of all regular languages that is satisfied if and only if the two binary relations have the same binary relation $\#$, was named in Introduction by binary relation $\mathcal{R}$. In this case, we can consider *subclasses of the self-binary relation* $\mathcal{R}$ (depending, for example, on the operations applied to the automaton $K^{\#}$ described in Sections VI and VII to get the desired regular language) – and we will show in one of the following publications that the subclasses of binary relations defined by us form a lattice.

References

[1] Melnikov B., Melnikova A., *Edge-minimization of non-deterministic finite automata*. Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). 2001, vol. 8, no. 3, pp. 469–479.

[2] 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.

[3] Melnikov B., Melnikova A., *Some more on the basis finite automaton*. Acta Univ. Sapientiae, Informatica. 2013, vol. 5, no. 2, pp. 227–244.

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

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

[6] Melnikov B., Vakhitova A., *Some more on the finite automata*. Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). 1998, vol. 5, no. 3, pp. 495–505.

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

[8] Polák L., *Minimalizations of NFA using the universal automaton*. International Journal of Foundation of Compututer Sciences. 2005, vol. 16, no. 5, pp. 999–1010.

[9] Melnikov B., *Once more on the combining states of nondeterministic finite automaton*. Proceedings of XI International Scientific Conference on the Problems of Theoretical Cybernetics. M., Russian State University for the Humanities Ed. 1996. P. 139–141. (in Russian)

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

[11] Krivolapova A., Melnikova E., Sofonova N., *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)

[12] Melnikov B., Sayfullina M., *On some algorithms of equivalent transformations of nondeterministic finite automata*. Izvestiya of universities. Mathematics. 2009, no. 4, pp. 67–72. (in Russian) (English translation: Mel'nikov B., Saifullina M., *Some algorithms for equivalent transformations of nondeterministic finite automata*. Russian Mathematics (Izv. VUZ). 2009, no. 4, pp. 54–56.)

[13] Hromkovič J., *Theoretical Computer Science. An Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography*. Springer, 2003. 321 p.

[14] Melnikov B., Tsyganov A., *The state minimizaton 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.

[15] Melnikov B., Radionov A., Moseev A., 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.