INFORMATICA, 2002, Vol. 13, No. 3, 299–310

A New Algorithm of Constructing the Basis Finite Automaton

Boris MELNIKOV *
Department of Mechanics and Mathematics, Simbirsk State University
L. Tolstoy str. 42, 432700 Simbirsk, Russia
e-mail: bormel@mail.ru

Alexandra MELNIKOVA
Department of Economics, Dimitrovgrad Branch of Simbirsk State University
Dimitrov str. 4, 433510 Dimitrovgrad, Russia
e-mail: avahi@mail.ru

Received: January 2002

Abstract. In this paper we consider non-deterministic finite Rabin–Scott's automata. We obtain some properties for the basis automaton, which is, like automaton of canonical form, an invariant of a given regular language. We obtain also a new algorithm of constructing the basis automaton for a given regular language.

Key words: regular languages, basis finite automata.

1. Introduction

This paper can be regarded to continue the papers (Melnikov and Vakhitova, 1998; Melnikov, 1999; Vakhitova, 1999; Melnikov, 2000; Melnikov and Melnikova, 2001; Melnikov and Melnikova, 2002; Melnikov and Sciarini–Guryanova, 2002). We consider in this paper non-deterministic finite Rabin–Scott's automata and use some notations of the mentioned series of works, see them for more information and examples. Remark that we also use notation of Brauer (1984). At first, we consider this paper as a continuation of (Melnikov, 1999; Vakhitova, 1999), since we consider here a special binary relation and some properties of the basis finite automaton for a given regular language.

In opinion of the authors of this paper, applications of basis automata are possible and desirable in various areas of the formal languages theory. In (Melnikov and Melnikova, 2001; Melnikov and Sciarini–Guryanova, 2002), we have obtained two of such new applications, i.e., a special description of all the possible edges of a finite automaton defining the given regular language and the edge-minimization. But the authors are sure, that there are possible a lot of other applications.

*The first author was partially supported by the Russian Foundation of the Basic Research (project No. 00-15-99253).

Thus, the basis automata are desirable in various areas of the theory of regular languages. It is important to note, that there are such applications, which can be used for practical programming. For more details see, e.g., the conclusions of the papers (Melnikov, 1999; Melnikov and Melnikova, 2001); let us add to the tasks considered there, that the problems of economical representation of the finite automaton in computer memory is connected with the modelling the work of this automaton, and therefore can be also used, e.g., for some tasks of bottom-up parsing.

Therefore, it is important to obtain different algorithms for the same problem. Let us notice however, that in the previous papers of the authors (Melnikov and Vakhitova, 1998; Melnikov, 1999; Vakhitova, 1999; Melnikov, 2000; Melnikov and Melnikova, 2001; Melnikov and Melnikova, 2002; Melnikov and Sciarini–Guryanova, 2002) we did *not* consider questions, connected with the complexity of considered algorithms. Moreover, authors are sure that such questions are *not* interesting for the tasks of practical programming: in such tasks, we compare two or more algorithms for the "real" automata, and usually do not examine the specially constructed ones, where the complexity reaches extremum.[1]

Thus, various algorithms for solving the same problem are important for practical tasks, connected with finite automata. Therefore we describe a new algorithm of construction of the basis finite automaton (this algorithm was not considered in (Vakhitova, 1999)), and consider some connected problems.

The structure of this paper is the following. In Section 2, we consider some definitions and simple facts of the previous papers, where basis automata were considered. In Section 3, we consider some properties of basis automata used in this paper; such properties were not consider in (Vakhitova, 1999). The main result of the paper, i.e., a new algorithm of constructing the basis finite automaton for the given regular language, is obtained in Section 4. In Section 5, we consider an example of such constructing. And in Section 6, we consider possible connection betwen using basis and canonical automata in some tasks.

2. Preliminaries

We use the well-known notation of Brauer (1984). In addition, we use the following designations. Let

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

be some non-deterministic finite automaton. Then:

- The language defined by K is designated by $\mathcal{L}(K)$.
- Input and output languages of the state q (i.e. the languages defined by automata

  $$(\, Q, \Sigma, \delta, \{q\}, F \,) \qquad \text{and} \qquad (\, Q, \Sigma, \delta, S, \{q\} \,)$$

  respectively) are denoted by $\mathcal{L}_K^{in}(q)$ and $\mathcal{L}_K^{out}(q)$.

[1] For this thing, see also Section 4.

- In the figures of transition graphs, initial and final states will be marked by small "input" and "output" arrows respectively.

Let us consider some definitions and simple facts of the revious papers (at first, of (Melnikov, 1999; Vakhitova, 1999)) about the binary relation $\#$ and basis automata.

Let a regular language L be given. Let

$$\widetilde{K} = (\, Q, \Sigma, \delta_Q, \{s_Q\}, F_Q)$$

be its automaton of canonical form, and

$$\widetilde{K^R} = (\, R, \Sigma, \delta_R, \{s_R\}, F_R)$$

be the automaton of canonical form defining its mirror language, i.e., L^R. Then for the states of these automata consider the following binary relation $\#$ ($\# \subseteq Q \times R$):

$$A \# X \quad \text{if and only if} \quad \Big(\exists uv \in L\Big)\Big(u \in \mathcal{L}^{in}_{\widetilde{K}}(A)\,,\, v \in \mathcal{L}^{in}_{\widetilde{K^R}}(X)\Big)\,.$$

Remark that this definition can also be considered as an *algorithm* of making the relation $\#$. Remark also, that considering the only useless state of the canonical automaton (let us define such states by N for all such automata), we can have neither $N \# X$ nor $A \# N$. See (Melnikov, 1999) for details.

Define the automaton

$$\mathcal{BA}(L) = (\, \mathcal{T}, \Sigma, \delta_T, S_T, F_T\,)$$

(the basis finite automaton for L) in the following way.

- $\mathcal{T}$ is the set of pairs $T = (A, X)$, such that $A \in \mathcal{Q}$, $X \in \mathcal{R}$, and $A \# X$. (We shall write in such case $A = \alpha(T)$ and $X = \beta(T)$.)
- Define δ_T in the following way. For each $T_1, T_2 \in \mathcal{T}$ and $a \in \Sigma$ define $\delta_T(T_1, a) \ni T_2$ if and only if

$$\delta_Q(\alpha(T_1), a) = \{\alpha(T_2)\} \quad \text{and} \quad \delta_R(\beta(T_2), a) = \{\beta(T_1)\}. \tag{1}$$

- Define $T \in S_T$ if and only if $\alpha(T) = s_Q$ (i.e., $\alpha(T)$ is the only initial state of the automaton $\widetilde{K}$, it was already denoted by s_Q).
- Similarly, $T \in F_T$ holds, if and only if $\beta(T) = s_R$ (i.e., $\beta(T)$ is the only initial state of the automaton $\widetilde{K^R}$).

Remark that both the canonical automata (i.e., automata for the languages L and L^R) are defined in the only way (we could re-define the labels of the states only), therefore we can say that the automaton $\mathcal{BA}(L)$ is defined by the given L also in the only way.

For an example, consider the regular language L, which can be defined by the regular expression

$$(a + ab + ba)^*. \tag{2}$$

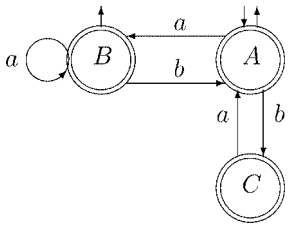

Fig. 1. Transition graph of automaton $\widetilde{K}$.

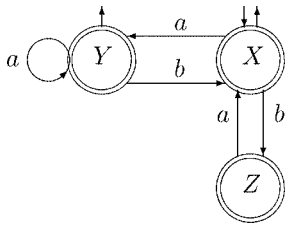

Fig. 2. Transition graph of automaton $\widetilde{K^R}$.

Transition graphs of the automata $\widetilde{K}$ and $\widetilde{K^R}$ are drawn on the Figs. 1 and 2 respectively.

Remark that we have $L = L^R$ in the considered example, therefore the automata $\widetilde{K}$ and $\widetilde{K^R}$ differ by the labels of the states only. However, we need the figures for transition graphs of both automata to continue the example.

By the definition, we obtain the following basis automaton.

		a	b
$\rightarrow \quad \leftarrow$	(A,X)	(B,Z)	(C,Y)
$\rightarrow$	(A,Y)	$(B,X),(B,Y)$	
$\leftarrow$	(B,X)	(B,Z)	(A,Y)
	(B,Y)	$(B,X),(B,Y)$	
	(B,Z)		(A,X)
	(C,Y)	$(A,X),(A,Y)$	

For more details, see (Vakhitova, 1999). Some properties of the basis automaton are considered also in (Vakhitova, 1999) and in the remained part of this paper.

3. Some Properties of the Basis Automaton

Let us consider some properties of the basis automaton, which also can be titled as properties of the table of corresponding states. We already considered this table in some propositions in (Melnikov, 1999; Vakhitova, 1999; Melnikov, 2000; Melnikov and Melnikova, 2001; Melnikov and Melnikova, 2002; Melnikov and Sciarini–Guryanova, 2002). [2] However, we consider this table as a binary relation only, and did not consider its properties. But there exist some binary relations, such that we cannot form such table by them. Theorems 3.5 and 3.6, which are considered in this section, formulate some necessary conditions for such binary relations.

But at first consider some simple facts. The following Propositions 3.1 and 3.2 hold simply by the definition of the basis automaton.

[2] Certainly, this table correlates to the basis automaton. For example, Proposition 2.1 of (Melnikov and Sciarini–Guryanova, 2002) can be considered either as a property of basis automata or as a property of tables of corresponding states.

Proposition 3.1. *Let the regular language L be given, $\widetilde{K}$ be automaton of canonical form defining L, A be some state of $\widetilde{K}$. Then for each state X of the automaton $\widetilde{K^R}$, we have*

$$\mathcal{L}^{in}_{\widetilde{K}}(A) = \mathcal{L}^{in}_{\mathcal{BA}(L)}\big((A, X)\big).$$

Below, we shall often use automata $\widetilde{K^R}$ and $\left(\widetilde{K^R}\right)^R$; therefore let us use the following notation:

$$\left(\widetilde{K^R}\right)^R = <K> .$$

Proposition 3.2. *Let the regular language L be given, $\widetilde{K^R}$ be automaton of canonical form defining L^R, X be some state of $\widetilde{K^R}$.*[3] *Then for each state A of the automaton $\widetilde{K}$, we have*

$$\mathcal{L}^{out}_{<K>}(X) = \left(\mathcal{L}^{in}_{\widetilde{K^R}}(X)\right)^R = \mathcal{L}^{out}_{\mathcal{BA}(L)}\big((A, X)\big).$$

And the following Propositions 3.3 and 3.4 can be considered either as a corollary of (Vakhitova, 1999, Section 6), or simply as a consequence of the definition of the basis automaton.

Proposition 3.3. *Let the regular language L be given, $\widetilde{K}$ be automaton of canonical form defining L. Then for some state A of $\widetilde{K}$, we have*

$$\mathcal{L}^{out}_{\widetilde{K}}(A) = \bigcup_{X \in \widetilde{K^R}} \mathcal{L}^{out}_{\mathcal{BA}(L)}\big((A, X)\big)$$

(where $X \in \widetilde{K^R}$ means that X is some state of the automaton $\widetilde{K^R}$).

Proposition 3.4. *Let the regular language L be given, $\widetilde{K^R}$ be automaton of canonical form defining L^R. Then for some state X of $\widetilde{K^R}$, we have*

$$\mathcal{L}^{in}_{\widetilde{K^R}}(X) = \mathcal{L}^{out}_{<K>}(X) = \bigcup_{A \in \widetilde{K}} \mathcal{L}^{in}_{\mathcal{BA}(L)}\big((A, X)\big)$$

(where $A \in \widetilde{K}$ means that A is some state of the automaton $\widetilde{K}$).

Thus, consider the main facts of this section.

Theorem 3.5. *Let the canonical automaton for the given regular language L have at least 2 different states,*[4] *and (A, B) is some pair of such states. Then there exists a state*

[3] And, therefore, of the automaton $<K>$.

[4] As in previous papers, we do not consider the only useless state of automaton of canonical form. For details, see (Melnikov, 1999).

X of the canonical automaton defining L^R, such that the basis automaton for L contains exactly 1 *state of the following:* (A, X) *and* (B, X). [5]

Proof. We shall use here the usual notation for automata $\widetilde{K}$ and $\widetilde{K^R}$, not defining K (which can be considered as some automaton for the given L).

Using the rule of contraries, suppose that we have 2 following states, A and B, such that for each state X of the canonical automaton defining L^R, the basis automaton for L contains both states (A, X) and (B, X). (There are the states having the same strings in the table of corresponding states.)

By Proposition 3.2, for such state X (of the automaton $\widetilde{K^R}$), we have

$$\mathcal{L}^{out}_{\mathcal{BA}(L)}\big((A, X)\big) = \mathcal{L}^{out}_{\mathcal{BA}(L)}\big((B, X)\big). \tag{3}$$

Therefore, since Proposition 3.3,

$$\mathcal{L}^{out}_{\widetilde{K}}(A) = \bigcup_{X \in \widetilde{K^R}} \mathcal{L}^{out}_{\mathcal{BA}(L)}\big((A, X)\big)$$

(similarly for B instead of A), and, since (3),

$$\mathcal{L}^{out}_{\widetilde{K}}(A) = \mathcal{L}^{out}_{\widetilde{K}}(B).$$

And the last equation conflicts to the property of automata of canonical form (or to the method of constructing such automata, see, e.g., (Brauer, 1984; Aho and Ullman, 1972), since each of them cannot have different states having the same output language.

Similarly, the following fact holds.

Theorem 3.6. *Let the canonical automaton for the mirror image of the given regular language L have at least* 2 *different states, and (X, Y) is some pair of such states. Then there exists a state A of the canonical automaton defining L, such that the basis automaton for L contains exactly* 1 *state of the following:* (A, X) *and* (A, Y).

Proposition 3.7. *The basis automaton for a given regular language is unambiguous one.* [6]

Proof. This proposition is a corollary of (Vakhitova, 1999, Section 2).

[5] We can also formulate this fact in other words. Let the table of corresponding states for L contains at least 2 strings. Then there exists a column of this table, such that the basis automaton for L contains exactly 1 square of 2 corresponding to considered strings and column.

[6] Authors do not define unambiguous and ambiguous finite automata by special definition. Similarly to the other cases of formal languages theory for the word "ambiguous", an ambiguous finite automaton can define some word of its language by 2 or more sequences of states. Certainly, there is no difference between sets of languages of unambiguous and ambiguous finite automata, because both these sets are simply sets of regular languages.

However, unambiguous finite automata can be useful for various problems. For example, there exists an analogy between sequence of subsets of context-free languages (and corresponding push-down automata) and the following sequence of subsets of the set of finite automata: arbitrary automata, unambiguous automata, deterministic automata, automata of canonical form.

We are going to consider some facts, connected with the Proposition 3.7, in a following paper.

4. A New Algorithm of Constructing the Basis Finite Automaton

In this section we obtain an algorithm of constructing the basis finite automaton. This algorithm was not considered in (Vakhitova, 1999), where basis automata were defined and the simple algorithm of their constructing [7] was considered. Remark in advance, that we cannot unambiguously answer the questions, which of these two algorithms is better, because of the following.

- The usual mathematical analisys of complexity of the algorithms, like (Aho *et al.*, 1974; Goodma and Hedetniemi, 1977; Wirth, 1985) etc., [8] gives almost the same results. However, we can obtain results the worst complexity only, because in practice, we cannot consider all the finite automata, even having 4 or more states for the regular languages on alphabets having 2 or more letters.

- Results of practical programing show, that algorithm of Vakhitova (1999) performs computations quickly for almost all examples of regular languages having a small number of the states in their canonical automata.[9] But considering automata, obtained in real programming tasks, we have as a rule, a large number of states.[10] For some tasks (at first, related to the description of programming languages, for example, to the problems considered in (Melnikov and Kashlakova, 2000)) the algorithm of Vakhitova (1999) is preferable, and for some other (related to the theory of expected utility, see, e.g., (Fishburn, 1982)) the new one is. Authors are sure, that the detailed research of two algorithms *can be obtained by some statistical technologies only*, moreover, for different tasks the different technologies have to be used.

- The most of computations, which are used in this section as the supported subalgorithms, can be also used for some other problems dealing with the basis automata. (An example of such problems is edge-minimization of non-deterministic automata, see (Melnikov and Melnikova, 2001).) Vice versa, when we have both automata of canonical form (i.e., automata of canonical form for the given regular language and for its mirror image) *in advance*, then the algorithm obtained in this section is much more quick than one of Vakhitova (1999).

Thus, both these algoriths of constructing the basis automaton for a given regular language (i.e., algorithms of Vakhitova (1999) and of this section) can be useful for special sets of problems.

Thus, let a regular language L be given. Let also

$$\widetilde{K} = (\mathcal{Q}, \Sigma, \delta_Q, \{s_Q\}, F_Q) \quad \text{and} \quad \widetilde{K^R} = (\mathcal{R}, \Sigma, \delta_R, \{s_R\}, F_R)$$

[7] I.e., the algorithm which is simply the corollary of their definition. See also Section 2.

[8] This analisys exceed the bounds of this paper. See about this thing, e.g., in Introduction.

[9] However, this fact holds, if we consider this task as an *isolated* one. See for this thing also the next item.

[10] To say, 25 states and more. And, therefore, we have to use some heuristical algorithms (genetic ones etc.) for constructing the relation $\#$ and basis automata. Authors hope to publish some approaches to constructing such algorithms in the future.

be automata of canonical form, defining languages L and L^R respectively. Remark that we shall consider transition functions δ_Q and δ_R as the sets of elements of corresponding binary relations. Therefore in examples given below, edges of automata $\widetilde{K}$ and $\widetilde{K^R}$ are denoted by capital Greek letters, which do not coincide with Latin ones. As a rule, edges of automaton $\widetilde{K}$ are denoted by the letters, which are ordered in Greek alphabet till the letter Ξ; and edges of automaton $\widetilde{K^R}$ (and also $<K>$) are denoted by the letters, which are ordered in Greek alphabet since the letter Π. [11]

For some edge Γ from the state A into the state B labelled by $a \in \Sigma$, denote $\alpha^a(\Gamma) = A$ and $\beta^a(\Gamma) = B$. Let us remark, that in this section the considered letter a has to be fixed a priori.

Thus, let us fix $a \in \Sigma$; remark that the procedure considered below should be made for each $a \in \Sigma$. Let δ_Q^a be edges of automaton $\widetilde{K}$ (i.e., elements of the set δ_Q), labelled by the fixed a. Similarly, let δ_R^a be edges of automaton $\widetilde{K^R}$ (i.e., elements of the set δ_R), also labelled by a. The same notation will be used for the set of edges of automaton $<K>$; to prevent variant reading, we shall *not* use definitions α^a and β^a for the edges of automaton $\widetilde{K^R}$ (because, as we said before, they are used for the edges of automaton $<K>$ having the same names). There is important, that for each given regular language L, the automaton $<K>$ constructed by this section is unambiguous; in general, each automaton, which is the mirror automaton for a deterministic one, is also unambiguous.

Consider the binary relation

$$\#^a \subseteq \delta_Q^a \times \delta_R^a,$$

definied in the following way. For some $\Gamma \in \delta_Q^a$ and $\Omega \in \delta_R^a$, define $\Gamma \#^a \Omega$ if and only if for some $w \in L$ the following representation holds: $w = uav$, and also:

$$u \in \mathcal{L}_K^{in}(\alpha^a(\Gamma)), \tag{4}$$

$$u \in \mathcal{L}_{<K>}^{in}(\alpha^a(\Omega)), \tag{5}$$

$$v \in \mathcal{L}_K^{out}(\beta^a(\Gamma)), \tag{6}$$

$$v \in \mathcal{L}_{<K>}^{out}(\beta^a(\Omega)). \tag{7}$$

As before in such cases (e.g., defining relation $\#$ in (Melnikov, 1999), defining the basis automaton in (Vakhitova, 1999)), we can consider the definition of the relation $\#^a$ by (4)–(7) not only as the definition, but also as an *algorithm* of constructing this relation. [12] And,

[11] However, the leter Σ, as before, denotes the given alphabet.

[12] We can simply explain this thing in the following way. For each state q of any finite automaton K, languages $\mathcal{L}_K^{in}(q)$ and $\mathcal{L}_K^{out}(q)$ are regular, therefore their intersection is also a regular language. Besides, we can re-formulate the conditions (4) and (5) in the following way:

$$\mathcal{L}_K^{in}(\alpha^a(\Gamma)) \cap \mathcal{L}_{<K>}^{in}(\alpha^a(\Omega)) \neq \emptyset \ .$$

Similarly for the conditions (6) and (7), although the formulation by the conditions (4)–(7) is some more comfortable for the following work.

also like those cases, we shall *not* consider questions of constructing effective methods for this relation.

Proposition 4.1. *Let $\Gamma \in \delta_Q$ and $\Omega \in \delta_R$ be some edges of the canonical automata defining the languages L and L^R respectively. Then the basis automaton $\mathcal{BA}(L)$ has the edge*

$$\delta_T\Big(\big(\alpha^a(\Gamma), \beta^a(\Gamma)\big), a\Big) \ni \big(\alpha^a(\Omega), \beta^a(\Omega)\big), \tag{8}$$

if and only if $\Gamma \#^a \Omega$.

Proof. Like formulating the relation $\#^a$, we suppose that the letter $a \in \Sigma$ is fixed.

Let us re-formulate the condition $\Gamma \#^a \Omega$ (i.e., (4)–(7)) in another way. After fixing the words u and v, which are used in (4)–(7),[13] we can re-formulate both (4) and (6) (i.e., the first and third sub-conditions of the condition $\Gamma \#^a \Omega$) as follows:

$$\delta_Q(\alpha^a(\Gamma), a) = \{\beta^a(\Gamma)\}. \tag{9}$$

Similarly, both (5) and (7) can be re-formulated in the following way:

$$\delta_R(\beta^a(\Omega), a) = \{\alpha^a(\Omega)\}. \tag{10}$$

Let us remark for (10), that we apply the functions α^a and β^a to the edge Γ, considering its as the edge of automaton $<K>$, not $\widetilde{K^R}$.

Joining the conditions (9) and (10), we obtain (8).

5. An Example

Let us consider an example, i.e., let us continue to consider the regular language (2). At first, consider transition graphs of the automata $\widetilde{K}$ and $<K>$. Remark that the graphs of $\widetilde{K}$ and $\widetilde{K^R}$ (the last one is the mirror automaton of $<K>$) were already drawn in Section 2, however, we need here the designations (not labels) of the edges. Thus, the automata $\widetilde{K}$ and $<K>$, including designations of the edges by capital Greek letters, are drawn on Fig. 3 and Fig. 4, respectively.

Now, let us fix the letter $a \in \Sigma$ and construct the relation $\#^a$. There exist 8 elements of this relation, this fact can be explained by the following table:

$\#^a$	Π	Υ	Φ
Γ	a	$\underline{a}a$	ab
Δ	$a\underline{a}$	$a\underline{a}a$	$a\underline{a}b$
Θ	ba	$b\underline{a}a$	

[13] Remark that such words u and v do exist. This fact is simply the consequence of the method of constructing automata $\widetilde{K}$ and $<K>$.

In this table, each pair of edges of automata $\widetilde{K}$ and $<K>$ defines the square. In this square, there is written a word; if it contains 2 or more letters a, then a letter is underlined. The word written in this square is the word of the considered language, having the following property. Reading this word, both the considered automata (i.e., $\widetilde{K}$ and $<K>$, let us remind of their unambiguity) run by these edges for underlined letter a, or for the only such letter. (Certainly, all the squares of the table can be marked by other words of the given language. We wrote the words which have the minimum possible length.)

The pair of edges Θ and Φ does not have corresponding word. The absence of such word for these edge could be explained by the examination of the deterministic automata (e.g., automata of canonical form), corresponding to the output languages of states C and X, i.e., to the regular languages

$$\mathcal{L}^{out}_{\widetilde{K}}(C) \quad \text{and} \quad \mathcal{L}^{out}_{<K>}(X)\,.$$

(Having construct such automata, we can simply obtain, that the itersection of their languages is the empty set.)

However in particular cases, including the considered one, we can solve such particular problem much more simple, without constructing automata defining the output languages of the pair of states. In the considered example, the automaton $\widetilde{K}$ runs the edge Θ only for two following cases: either having read b as the first letter of the defined word, or having read bb (see for details Fig. 3). In both these cases, the automaton $<K>$ can be only in the state Y (not X, see for details Fig. 4), and, therefore, cannot run the edge Φ.

Thus, we have obtained the 8 elements of the binary relation $\#^a$. Let us consider, how we can use these elements for obtaining the edges of the basis automaton for the same language. [14] For this thing consider the following table, where the square, corresponding to a pair of edges of the automata $\widetilde{K}$ and $<K>$, contains the edge of the basis automata, which is constructed by (8):

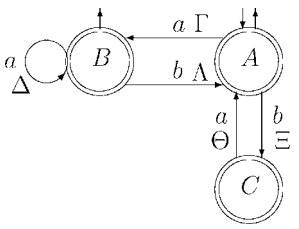

Fig. 3. Edges of automaton $\widetilde{K}$.

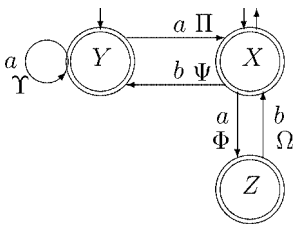

Fig. 4. Edges of automaton $<K>$.

[14] Remark once more, that we have already constructed such automaton in another way in (Vakhitova, 1999).

$\#^a$	Π	Υ	Φ
Γ	$(A,Y) \to (B,X)$	$(A,Y) \to (B,Y)$	$(A,X) \to (B,Z)$
Δ	$(B,Y) \to (B,X)$	$(B,Y) \to (B,Y)$	$(B,X) \to (B,Z)$
Θ	$(C,Y) \to (A,X)$	$(C,Y) \to (A,Y)$	

This table shows, that the edges are the same as in (Vakhitova, 1999, Section 3).

Now consider the letter $b \in \Sigma$. Let us describe constructing relation $\#^b$ some less detailed, than we did for the relation $\#^a$; e.g., let us write two tables together. Thus, the table 2×2 of the binary relation $\#^b$ contains the following 3 elements:

$\#^b$	Ψ		Ω	
Λ	aab	$(B,X) \to (A,Y)$	ab	$(B,Z) \to (A,X)$
Ξ	b	$(A,X) \to (C,Y)$		

As for the letter a, the edges are the same as in (Vakhitova, 1999).

6. Conclusion

We can formulate the problem, correlated to ones considered in this paper, for the solution in the future. This problem is the following one: to describe by a special way the set of finite automata (or the set of regular languages), such that we can obtain from them *any* regular language using special morphism and concatenation, but not using iteration. [15] Authors have some interesting facts for such problem, e.g., they have examples of *different* automata (i.e., automata which cannot be obtained by special operations, corresponding to morphism and concatenation), having the same table of binary relation of their basis automata. Moreover, they have obtained *all such languages*, having the following limitation: automaton of canoncal form, defining either such language or its mirror image, conains no more than 2 states (as before, we do not consider the only useless state of canonical automaton).

Thus, in this paper we have continued to consider basis finite automata, which are, like automata of canonical form, invariants of regular languages. Sometimes there is more suitable to consider basis automata (not canonical, and, in general, deterministic ones). E.g., such approach is more suitable for the tasks, where we have to consider the language L^R or the reaction (behavior) of some automaton defining L^R (not only the given language L). Some of such tasks were considered in this paper and previous papers of the authors cited before.

References

Aho, A., J. Hopcroft, J. Ullman (1974). *The Design and Analysis of Computer Algorithms*. N.J., Addison–Wesley.

[15] Corresponding operations for automata were considered in (Melnikov and Vakhitova, 1998).

Aho, A., J. Ullman (1972). *The Theory of Parsing, Translation and Compiling. V. 1: Parsing*. N. J. , Prentice–Hall, Inc., Englewod Cliffs.

Brauer, W. (1984). *Automatentheorie. Eine Einführung in die Theorie Endlicher Automaten*, B. G. Teubner Stuttgart.

Fishburn, P. (1982). *The Foundations of Ecpected Utility*. Holland, D. Reidel Publ. Co.

Goodman, S., S. Hedetniemi (1977). *Introduction to Design and Analysis of Algorithms*. N.Y., McGraw–Hill Book Company.

Melnikov, B. (1999). A new algorithm of the state-minimization for the nondeterministic finite automata. *The Korean Journal of Computational and Applied Mathematics*, **6**(2), 277–290.

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

Melnikov, B., E. Kashlakova (2000). Some grammatical structures of programming languages as simple bracketed languages, *Informatica*, **11**(4), 441–454.

Melnikov, B., A. Melnikova (2001). Edge-minimization of non-deterministic finite automata. *The Korean Journal of Computational and Applied Mathematics*, **8**(3), 469–479.

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

Melnikov, B., N. Sciarini–Guryanova (2002). Possible edges of a finite automaton defining the given regular language. *The Korean Journal of Computational and Applied Mathematics*, **9**(2), 475–485.

Melnikov, B., A. Vakhitova (1998). Some more on the finite automata. *The Korean Journal of Computational and Applied Mathematics*, **5**(3), 495–506.

Vakhitova, A. (1999). The basis automaton for the given regular language. *The Korean Journal of Computational and Applied Mathematics*, **6**(3), 617–624.

Wirth, N. (1985). *Algorithms and Data Structure*. N.J., Prentice–Hall Inc.

B. Melnikov received his B.S. from Moscow State University in 1984 and Cand.Sci. Degree (Ph.D.) at the same University in 1990. Since 1991 he has been at Simbirsk (Ulyanovsk) Branch of Moscow University and Simbirsk State University. He received his Doct.Sci. Degree at Moscow State University in 1997. Since 1999 he has been a full professor of Simbirsk State University, and since May 2001 he has been head of Chair of Foundation of Computer Sciences. Main research interests: finite automata, regular languages, theory of semigroups; heuristical algorithms (genetic, branch-and-bounds etc.); artificial intelligence; mathematical aspects of the object-oriented programming.

A. Melnikova (Vakhitova) received her B.S. in 1993. Since 1996 she was an Assistant Professor of Dimitrovgrad Branch of Simbirsk Polytechnical University. Since 2000 she has been an Associated Professor of Dimitrovgrad Branch of Simbirsk State University. Main research interests: finite automata, regular languages, mathematical linguistics.

Naujas algoritmas baziniams baigtiniams automatams konstruoti

Boris MELNIKOV, Alexandra MELNIKOVA

Straipsnyje nagrinėjami nedeterminuoti baigtiniai Rabin–Scott automatai. Parodyta, kad bazinis automatas, panašiai kaip kanoninis automatas, turi savybių nepriklausančių nuo juo aprašomos reguliariosios kalbos. Straipsnyje pasiūlytas naujas algoritmas baziniam duotosios reguliariosios kalbos automatui konstruoti.