Some more on the equivalent transformation of nondeterministic finite automata. Part II. The “deleting” algorithm

B. F. Melnikov

***Abstract*—This paper is a continuation of our following previous papers, where we considered some simple algorithms for combining states of the given nondeterministic finite automaton, the reduction some problems related to the star-height to considering automata, and possible classification of the states and loops of the given automaton.**

In this part of the paper, we shall describe an algorithm which deletes the state of a given nondeterministic finite automaton. This algorithm preserves basic properties of automata, i.e. the languages of the given and the obtained automata are the same, and the value of star-height for the obtained automaton is no more than such value for the given automaton.

Like Part I, we consider two states having the same values of the state marking functions. Then we could apply the same algorithms, but, generally speaking, in the case of the initial conditions considered in this part, the application of combining algorithm of previous part increases the value of star-height of the automaton under consideration. Then we should apply another algorithm, we consider such algorithm in this part. We call it by deleting algorithm, because it deletes a state; however, we not only delete a state, but sometimes add some edges, inputs and outputs before deleting.

We also consider some examples of using the described deleting algorithm.

***Keywords*—nondeterministic finite automata, regular languages, equivalent transformations, deleting state, the star-height problem.**

VII. Introduction to Part II (Once more about the motivation)

This paper is the continuation of [1], i.e. Part I. Moreover, as we noted in that part, it can be considered also as a continuation of some our previous papers, i.e. [2], [3], [4], [5] etc. We continue the numeration of sections, equations, definitions, propositions, theorems, tables, and figures, but use new numbers of references and footnotes.

As we said in Part I and in [5], we already reformulated the star-height problem for regular languages in the following way: *for the given regular language, we have to construct the equivalent finite automaton having the minimum possible star-height*. After that, considering n! bijective “order” functions (n is the number of the states of the “minimum” automaton), we construct corresponding regular expressions and choose one having the minimum possible star-height. Thus, a possible solution of the star-height problem for regular language is constructing such “minimum” automaton.

To build such an automaton, we perform some auxiliary equivalent transformations. The description of such transformations is the main subject of this paper.

Received June 27, 2019.

Boris F. Melnikov, Shenzhen MSU – BIT University (email: bf-melnikov @yandex.ru).

In Part I, we combined two states having the same values of the state marking functions: i.e., for states q' and q'' under consideration we should have

$$\varphi^{in}_K(q') = \varphi^{in}_K(q'') \quad \text{and} \quad \varphi^{out}_K(q') = \varphi^{out}_K(q''). \qquad (4)$$

In this part of the paper, we also could apply the same algorithms (because below, both the equations of (4) also hold). However, we shall consider in this part the case, when both (2) and (3) [1] do not hold. Therefore, generally speaking, in the case of the initial conditions considered in this part, the application of combining algorithm of Part I (i.e. Theorem 1) *increases* the value of $\mathcal{SH}$ of the automaton under consideration. Then, also generally speaking, the combining algorithm is also applicable, but *we should apply another one*. We shall consider this algorithm below.

Thus, similarly to Part I, we shall consider an algorithm for equivalent transformation of nondeterministic finite automaton. Also similarly to Part I, we shall not fully describe *why* we are doing this [2]: we are going to give details in some subsequent publications. Let us repeat, that we *only describe algorithms* for equivalent transformation and *prove non-increasing the value of star-height* for the obtained automaton.

This paper has the following structure. In Section IX, we consider the first stage of the deleting algorithm: we add some new inputs and outputs (i.e., initial and final states) without changing the language and value $\mathcal{SH}$ of the automaton under considerstion. In Section X, we give the second stage of this algorithm: we add some new edges and fulfill the same condition at the same time. In Section XI, we consider the last stage of this algorithm: this is direct deleting the state. In Section XII, we consider some examples for the deleting algorithm.

VIII. Some more on preliminaries

All designations used below were already given in [1], i.e. Part I. However, in Part I, no specific reference to designations Ψ and $\hat{\Psi}$ was given (despite these designations were used). This notation can be found in [6] (cited there).

In [7], we explained our use of single and double circles to denote states. In the current paper, only “ordinary” nondeterministic finite automata are used, therefore all states in the figures are represented by double circles.

Remark also, that the terminology related to the graph theory is agreed with [8].

[1] As we said before, we continue the numbering of the equations of Part I.

[2] “You’re far too keen on *where* and *how*, but not so hot on *why*”.

IX. The deleting algorithm: the first stage, adding the inputs and outputs

Thus, in this part of the paper, we also are working with the case when at least two states have the same values of the both state-marking functions, i.e. functions φ^{in} and φ^{out}. Of course, we remove one of these states, namely, one of them that has the greater value of the order function τ.

However, the algorithm we are describing is not simple: despite these states have the same values of the both state-marking functions, *the different sets of initial/final states may correspond to them.* Let us consider a simple example for this thing.

First, consider the "usual" language of the regular expression

$$(a + ab + ba)^*. \tag{5}$$

Its "usual" nondeterministic automaton is the following:

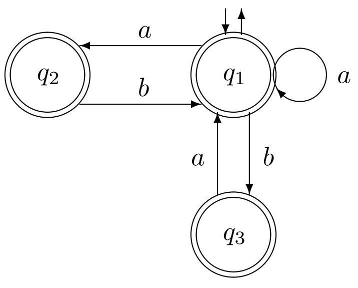

Figure 4. Automaton for the language of (5)

In the process of the most usual determinisation ([9] etc.), we obtained the automaton given by the following table:

Tab. 3. An equivalent deterministic automaton

		a	b
$\rightarrow$ $\leftarrow$	A	B	C
$\leftarrow$	B	B	D
	C	A	–
$\leftarrow$	D	B	C

(see [13] and [14, Footnote 1] about total (everywhere-defined) deterministic automata).

By [9] etc., we should combine states A and D for obtaining the canonical automaton. But for this paper, we need example of Tab. 3: we have two different states A and D having the same values of state-marking functions [3], but only A (not D) in an initial state.

Proposition 6: Let $L = \mathcal{L}(K)$ and for the state q of automaton (1) (where $q \notin S$), condition $\varphi^{in}_K(q) \ni s_\pi$ holds (by [13], [14], s_π is the only initial state of automaton $\widetilde{L}$). Then for automaton

$$K' = (\,Q, \Sigma, \delta, S \cup \{q\}, F\,)\,, \tag{6}$$

condition $\mathcal{L}(K') = \mathcal{L}(K)$ holds.

Proof. Evidently, $\mathcal{L}^{out}_{K'}(q) = \mathcal{L}^{out}_K(q)$. But, generally speaking,

$$\mathcal{L}^{in}_{K'}(q) \neq \mathcal{L}^{in}_K(q) \cup \{\varepsilon\},$$

(because, generally speaking, $q \notin S$). Despite this fact, the following equation is also evident:

$$\mathcal{L}^{in}_{K'}(q) \cdot \mathcal{L}^{out}_{K'}(q) = (\mathcal{L}^{in}_K(q) \cup \{\varepsilon\}) \cdot \mathcal{L}^{out}_K(q). \tag{7}$$

Since $\varphi^{in}_K(q) \ni s_\pi$, the following holds:

$$(\exists u \in \mathcal{L}^{in}_{\mathcal{BA}(L)}(s_\pi))\,(\forall v \in \mathcal{L}^{out}_K(q))\,(uv \in L).$$

Besides, because automaton $\widetilde{L}$ is deterministic, we obtain, that

$$(\forall u \in \mathcal{L}^{in}_{\mathcal{BA}(L)}(s_\pi))\,(\forall v \in \mathcal{L}^{out}_K(q))\,(uv \in L).$$

We have $s_\pi \in \varphi^{in}_K(q)$, then $\varepsilon \in \mathcal{L}^{in}_{\mathcal{BA}(L)}(s_\pi)$, therefore

$$(\forall v \in \mathcal{L}^{out}_K(q))\,(\varepsilon v \in L).$$

The last fact and (7) prove the proposition. $\square$

The following statement about outputs is proved similarly.

Proposition 7: Let $L = \mathcal{L}(K)$, and for the state q of automaton (1) (where $q \notin F$), condition $\varphi^{out}_{\widetilde{K}}(q) \ni s_\rho$ holds (s_ρ is the only initial state of automaton $\widetilde{L^R}$). Then for automaton

$$K' = (\,Q, \Sigma, \delta, S, F \cup \{q\}\,)\,, \tag{8}$$

condition $\mathcal{L}(K') = \mathcal{L}(K)$ holds. $\square$

X. The deleting algorithm: the second stage, adding the edges

Like previous section, despite two states may have the same values of the both state-marking functions, *the different sets of edges may correspond to them.* Let us consider two simple examples for this thing.

First, let us continue to consider the automaton of Figure 4 and its language (5). We can also consider the universal automaton (automaton $\mathcal{COM}$) for the same language (about the universal automaton, see details in [10], [11], [12]):

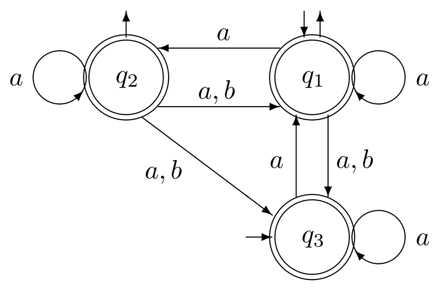

Figure 5. The second automaton for the language of (5)

Based on calculations similar to those in [13], we find that the values of functions of the same name (φ^{in} and φ^{out}) are the same for two states labelled q_1.

Secondly, let us consider the other language of [13], [14].

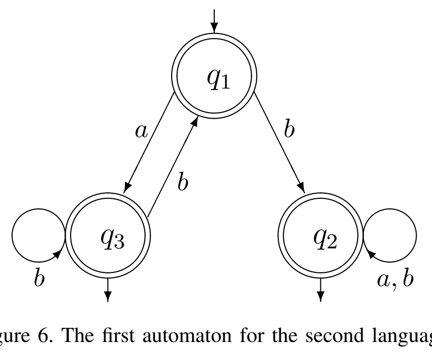

Figure 6. The first automaton for the second language

[3] In the usual designations of states of canonical automata $\widetilde{L}$ and $\widetilde{L^R}$ for such language L,

$$\varphi^{in}_K(A) = \varphi^{in}_K(D) = \{A\}\,, \quad \varphi^{out}_K(A) = \varphi^{out}_K(D) = \{X, Y\}.$$

Its first automaton is given on Fig. 6. The second automaton for the same language is the following:

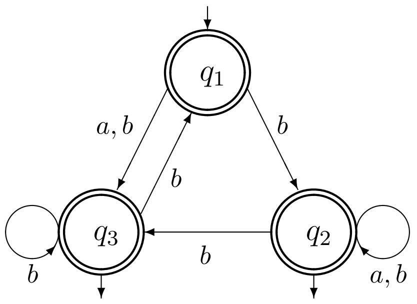

Figure 7. The second automaton for the second language

Like previous example, the identical states of two automata have the same values of functions φ^{in} and φ^{out}, but the set of edges of the first automaton is an own subset for the second one.

As we noted above, the algorithm described in this paper is based on the following auxiliary algorithms.

- First, we add the "missing" edges for the state having a smaller value of the order function τ (see [5] and Part I of this paper for details of τ).
- Secondly, we delete the the state having a greater value of τ.

Proposition 8: Let the following objects be given:

- the regular language L;
- some automaton (1) defining L;
- some states $q_1, q_2 \in Q$, belonging to two different highly connected components of automaton (1).

Let for some states $\begin{smallmatrix}A\\X\end{smallmatrix}$ and $\begin{smallmatrix}B\\Y\end{smallmatrix}$ of the equivalent basis automaton $\mathcal{BA}(L)$ (we allow possibilities $A = B$ and $X = Y$) and for some letter $a \in \Sigma$, the following hold:

$$\left[q_1 \ni \begin{smallmatrix}A\\X\end{smallmatrix}\right], \quad \left[q_2 \ni \begin{smallmatrix}B\\Y\end{smallmatrix}\right] \quad \text{and} \quad \begin{smallmatrix}A\\X\end{smallmatrix} \xrightarrow[\mathcal{BA}(L)]{a} \begin{smallmatrix}B\\Y\end{smallmatrix} \qquad (9)$$

(for notation $[q \ni \hat{q}]$ and corresponding examples, see [6]).

Then automaton

$$K' = K_{+(q_1 \xrightarrow{a} q_2)}$$

is equivalent to the given one, i.e., the equality

$$\mathcal{L}(K') = \mathcal{L}(K)$$

holds. [4]

Besides, for each state $q \in Q$ the following conditions hold:

$$\varphi^{in}_K(q) = \varphi^{in}_{K'}(q) \quad \text{and} \quad \varphi^{out}_K(q) = \varphi^{out}_{K'}(q). \qquad (10)$$

Proof. It is sufficient to prove, that $\mathcal{L}(K') \subseteq \mathcal{L}(K)$.

Let us consider any state

$$Y' \in \varphi^{out}_K(q_2)$$

and any corresponding word v', such that

$$(v')^R \in \mathcal{L}^{in}_{\widetilde{L^R}}(Y').$$

Let u be some word of the language $\mathcal{L}^{in}_{\widetilde{L}}(A)$. We have

$$ua \in \mathcal{L}^{in}_{\widetilde{L}}(B).$$

(This fact is proved simple; for example, it could be considered as the consequence of [4, Prop. 2.1].)

Besides, the condition

$$B \in \varphi^{in}_K(q_2)$$

holds, then we obtain, that

$$uav' \in L.$$

From the last fact, we obtain, that for automaton $\widetilde{L^R}$, the condition

$$(v')^R a \in \mathcal{L}^{in}_{\widetilde{L^R}}(X)$$

holds. The same fact is true for each state $X' \in Q_\rho$, such that $A \# X'$.

Thus,

$$\mathcal{L}^{out}_K(q_1) \ni av',$$

then for each state

$$A' \in \varphi^{in}_K(q_1),$$

there exists some word

$$u' \in \mathcal{L}^{in}_{\widetilde{L}}(A'),$$

such that $u'av' \in L$; therefore *for each* word

$$u' \in \mathcal{L}^{in}_{\widetilde{L}}(A')$$

the condition $u'av' \in L$ holds. [5] The last condition proves the equality $\mathcal{L}(K') = \mathcal{L}(K)$, which is also explained by the following two facts:

- for each pair of states, i.e.

$$A' \in \varphi^{in}_K(q_1) \quad \text{and} \quad Y' \in \varphi^{out}_K(q_2),$$

and for each pair of corresponding input words, i.e.

$$u' \in \mathcal{L}^{in}_{\widetilde{L}}(A') \quad \text{and} \quad (v')^R \in \mathcal{L}^{in}_{\widetilde{L^R}}(Y'),$$

condition $u'av' \in L$ holds;
- in automaton, the *new* loops (i.e. the loops, that were absent in the transition graph of automaton K) cannot appear. [6]

Thus, $\mathcal{L}(K') = L$. Then for automaton K', the equivalent canonical automaton is $\widetilde{L}$. Therefore we can use notation, associated to $\widetilde{L}$, also for automaton K'; we mean, at first, the use of subsets of their states as values of function $\varphi^{in}_{K'}$ (not only for function φ^{in}_K). Similarly, automaton $\widetilde{L^R}$ is equivalent to $(K')^R$, and we can use sets of their states for values of function $\varphi^{out}_{K'}$.

We proved in fact, that

$$\bigl(\forall A' \in \varphi^{in}_K(q_1)\bigr) \bigl(\exists B' \in \varphi^{in}_K(q_2)\bigr) \bigl(A' \xrightarrow[\widetilde{L}]{a} B'\bigr);$$

[4] Let us remark, that the simultaneous fulfillment of equalities $A = B$ and $X = Y$ is impossible, because we consider two *different* strongly connected components of (the transition graph of) the basis automaton $\mathcal{BA}(L)$.

Let us also remark, that we can reformulate the condition about the difference of the strongly connected components, in the following simpler way: the transition graph of automaton $\mathcal{BA}(L)$ has no path from $\begin{smallmatrix}B\\Y\end{smallmatrix}$ to $\begin{smallmatrix}A\\X\end{smallmatrix}$.

[5] This fact (i.e. the possibility of replacement in this situation the word "exists" for "for each") can be considered as a property of the canonical automaton $\widetilde{L}$.

[6] The impossibility of new loops is a consequence of the fact that q_1 and q_2 belongs to the *different* strongly connected components.

Let us also remark, that if we did *not* impose such a requirement, then the condition $\mathcal{L}(K') = \mathcal{L}(K)$, generally speaking, was false. However, the more detailed consideration of such examples is not included in the scope of this paper.

then
$$\varphi^{in}_{K'}(q_2) = \varphi^{in}_{K}(q_2).$$
Similarly
$$\varphi^{out}_{K'}(q_1) = \varphi^{out}_{K}(q_1).$$
The conditions
$$\varphi^{in}_{K'}(q_1) = \varphi^{in}_{K}(q_1) \quad \text{and} \quad \varphi^{out}_{K'}(q_2) = \varphi^{out}_{K}(q_2),$$
as well as (10) for $q \notin \{q_1, q_2\}$, are evident. $\square$

Definition 6: We shall call such edges $q_1 \xrightarrow{a} q_2$ by *inter-component* edges.

We shall use this notation in the following papers.

Proposition 9: For the conditions of Proposition 8, the equality
$$\mathcal{SH}(K') = \mathcal{SH}(K)$$
holds.

Proof. In the transition graph of automaton K', there *cannot* be paths from the state q_2 in the state q_1 because of the following. If such a path would exist, then we obtain existing of corresponding path of automaton $\mathcal{BA}(L)$, i.e. a path from state $\genfrac{}{}{0pt}{}{B}{Y}$ to state $\genfrac{}{}{0pt}{}{A}{X}$. And the last existing contradicts to the supposition we make before, i.e., that $\genfrac{}{}{0pt}{}{A}{X}$ and $\genfrac{}{}{0pt}{}{B}{Y}$ belongs to different strongly connected components of automaton $\mathcal{BA}(L)$.

Thus considering the same order function τ (it was given for automaton K) for the new automaton K', we obtain, that $\mathcal{SH}(K') = \mathcal{SH}(K)$. $\square$

XI. The deleting algorithm: the third stage, direct deleting the state

We shall not explain the initial conditions necessary for the following proposition (as we noted in Introduction). An explanation of the need to fulfill these conditions for the deleting algorithm will be given in one of the following publications.

Proposition 10: Let the following condition *hold*:
$$\Big(\exists i \in \{1, \dots, m{-}1\}\Big) \Big(\Psi(q_i) = \Psi(q_m)\Big), \tag{11}$$
and the following condition *does not hold*:
$$\Big(\exists (A,X) \in \bigcup_{q \in \{q_m\} \cup \hat{T}_m} \hat{\Psi}(q_i)\Big) \Big((A,X) \notin \hat{\Psi}_m\Big). \tag{12}$$

Then there exists automaton
$$K' = \big(\, Q \setminus \{q_m\}, \Sigma, \delta', S \setminus \{q_m\}, F \setminus \{q_m\}\,\big),$$
such that:

- $\mathcal{L}(K') = \mathcal{L}(K)$;
- $\mathcal{SH}(K') = \mathcal{SH}(K)$;
- for each state $q \in Q \setminus \{q_m\}$, conditions
$$\varphi^{in}_{K}(q) = \varphi^{in}_{K'}(q) \quad \text{and} \quad \varphi^{out}_{K}(q) = \varphi^{out}_{K'}(q)$$
hold.

Let us consider *some important comments to the initial conditions*.

- There is convenient to write the second condition as above, that is, as non-performance of (12).
- The non-performance of (12) implies the following fact:
$$\bigcup_{q \in \{q_m\} \cup \hat{T}_m} \hat{\Psi}(q_i) \subseteq \hat{\Psi}_m\,. \tag{13}$$
- The performance of (12) does not guarantee, that the set
$$\{q_m\} \cup \hat{T}_m$$
contains a complete $\genfrac{}{}{0pt}{}{A}{X}$-cyclic state (see [6] for this definition) for the considered states
$$A \in Q_\pi \quad \text{and} \quad X \in Q_\rho.$$

Proof. Let us describe the preliminary equivalent transformation of automaton K, consisting of adding some edges.

For this, let bus consider *any* accepted path of automaton K, passes through the state q_m. Let it be the path
$$\begin{gathered} \to p_1 \xrightarrow[K]{a_1} p_2 \xrightarrow[K]{a_2} \;\dots\; \xrightarrow[K]{a_{k_p-1}} (p_{k_p}{=}p) \\ \xrightarrow[K]{b'} (t'{=}t_1) \xrightarrow[K]{b_1} t_2 \xrightarrow[K]{b_2} \;\dots\; \xrightarrow[K]{b_{k_t-1}} (t_{k_t}{=}t'') \xrightarrow[K]{b''} \\ (r{=}r_1) \xrightarrow[K]{c_1} r_2 \xrightarrow[K]{c_2} \;\dots\; \xrightarrow[K]{c_{k_r-1}} r_{k_r} \to \,, \end{gathered} \tag{14}$$
where:

- the states $t_1,\, t_2,\, \dots t_{k_t}$ (these ones only) belong to the strongly connected component which contains the state q_m;
- from the states $t_1,\, t_2,\, \dots t_{k_t}$, at least one coincides with q_m;
- we allow the possibility $k_p = 0$; in this case, the first *string* of the path (14) is empty, t' is some initial state (it is the first state of this path), i.e. in this case, the edge $\xrightarrow[K]{b'}$ is being replaced for $\to$;
- we also allow the possibility $k_r = 0$ (the comments are similar to those above).

Because $t_1,\, t_2,\, \dots t_{k_t}$ belong to the same strongly connected component, we can consider this sequence as a part of some loop. For instance, we can consider the loop
$$\begin{gathered} (t'{=}t_1) \xrightarrow[K]{b_1} t_2 \xrightarrow[K]{b_2} \;\dots\; \xrightarrow[K]{b_{k_t-1}} (t_{k_t}{=}t'') \\ \xrightarrow[K]{b_{k_t}} t_{k_t+1} \xrightarrow[K]{b_{k_t+1}} \;\dots\; \xrightarrow[K]{b_{k_n-1}} (t_{k_n}{=}t')\,. \end{gathered} \tag{15}$$

Evidently, there exists a corresponding path of automaton $\mathcal{BA}(L)$ (because the automata K and $\mathcal{BA}(L)$ accept the same language); let this path (of automaton $\mathcal{BA}(L)$) be λ. Also evidently, the loops (15) and λ, are, generally speaking, *not simple* loops.

According to the state q_m selection method, there exists *some other* loop of automaton K, which:

- corresponds to the loop λ;
- does not contain the state q_m (because all the states of the set Q_m are important states).[7]

[7] The important states were defined before.

Let us explain this fact in details, i.e., we explain the existing of the loop of automaton K, possessing the properties described here.

By our suppositions (13), the condition $\hat{\Psi}(q_m) \subseteq \hat{\Psi}_m$ holds. Considering arbitrary pair $(A,X) \in \hat{\Psi}(q_m)$, we obtain that for some $l < m$, the state q_l is a complete $\genfrac{}{}{0pt}{}{A}{X}$-cyclic state.

Then, also by (13), *for each* loop of automaton $\mathcal{BA}(L)$ passing throw state $\genfrac{}{}{0pt}{}{A}{X}$, there exists a *corresponding* loop of automaton K, which *does not* contain other complete $\genfrac{}{}{0pt}{}{A}{X}$-cyclic states. (I.e., the such state is the already selected q_l.)

(We again used some notation of [6].)

Let this loop of automaton K be

$$(h'{=}h_1) \xrightarrow[K]{b_1} h_2 \xrightarrow[K]{b_2} \;\ldots\; \xrightarrow[K]{b_{k_h-1}} (h_{k_t}{=}h'') \xrightarrow[K]{b_{k_t}} h_{k_t+1} \xrightarrow[K]{b_{k_t+1}} \;\ldots\; \xrightarrow[K]{b_{k_n-1}} (h_{k_n}{=}h')\,.$$

Then the path

$$(h'{=}h_1) \xrightarrow[K]{b_1} h_2 \xrightarrow[K]{b_2} \;\ldots\; \xrightarrow[K]{b_{k_h-1}} (h_{k_t}{=}h'') \qquad (16)$$

(it also is the part of the last loop) also does not contain the state q_m.

For both states h' and h'', the conditions of Proposition 8 hold. Then we can add the edges

$$p \xrightarrow[K]{b'} h' \quad \text{and} \quad h'' \xrightarrow[K]{b''} r$$

without changing the accepted language and the star-height of the considered automaton. Thus, we obtain the following accepting path of the *obtained* automaton:

$$\to p_1 \xrightarrow[K]{a_1} p_2 \xrightarrow[K]{a_2} \;\ldots\; \xrightarrow[K]{a_{k_p-1}} (p_{k_p}{=}p) \xrightarrow[K]{b'} (h'{=}h_1) \xrightarrow[K]{b_1} h_2 \xrightarrow[K]{b_2} \;\ldots\; \xrightarrow[K]{b_{k_h-1}} (h_{k_t}{=}h'') \xrightarrow[K]{b''} (r{=}r_1) \xrightarrow[K]{c_1} r_2 \xrightarrow[K]{c_2} \;\ldots\; \xrightarrow[K]{c_{k_r-1}} r_{k_r} \to$$

(like before, the first and / or the last strings can be empty), which accepts the same word, like the path (14).

Evidently, we can make such equivalent transformation (which also does not change the value $\mathcal{SH}$ of the considered automaton) *for each possible pairs* t' and t''.[8] Then *for each* word of the considered language, there exists corresponding accepting path, which *does not* contain the set q_m.

Thus, we obtain

$$\mathcal{L}(K_{-q_m}) = \mathcal{L}(K).$$

Moreover, for each state $q \in Q \setminus \{q_m\}$, equalities

$$\varphi^{in}_K(q) = \varphi^{in}_{K'}(q) \quad \text{and} \quad \varphi^{out}_K(q) = \varphi^{out}_{K'}(q)$$

are evident. □

XII. Some examples for the deleting algorithm

Like Part I, we shall continue to consider examples where the sets Ψ (and, therefore, $\hat{\Psi}$ etc.) consist of one element. I.e., by [6, Def. 2],

$$|\varphi^{in}_K(q)| = |\varphi^{out}_K(q)| = 1\,;$$

the examples for the sets $\varphi^{in}_K(q)$ and / or $\varphi^{out}_K(q)$ consisting of 2 or more elements are complicated. Also like [6] and Part I, all the examples are special modifications of automaton $\mathcal{BA}(L)$ for the language of regular expression (5).

First, let us repeat automaton of [1, Fig. 1], see Figure 8. Now, unlike [1], we shall consider the other automaton:

- without an input present in Figure 8 (i.e., the state ${}^{A}_{Y}1$ is now *not* the initial one);
- without an edge present in Figure 8 (i.e., the edge
$$ {}^{B}_{X}2 \xrightarrow[\mathcal{BA}(L)]{b} {}^{A}_{Y}2 $$
is now *absent*);[9]
- the other order function τ.

For the new automaton and order function, see Figure 9.

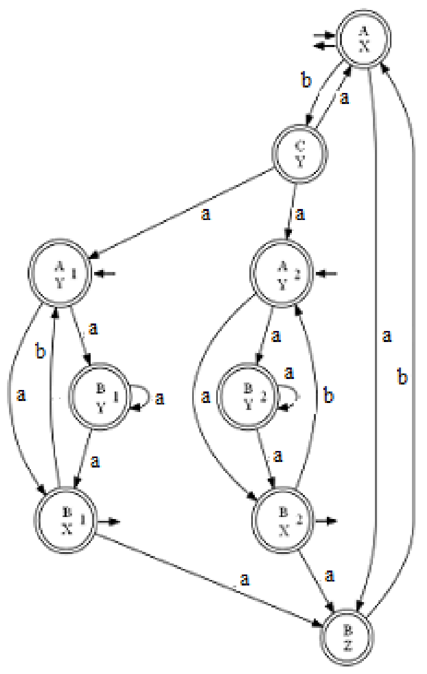

Figure 8. The *previous* automaton (for the language $(a + ab + ba)^*$)

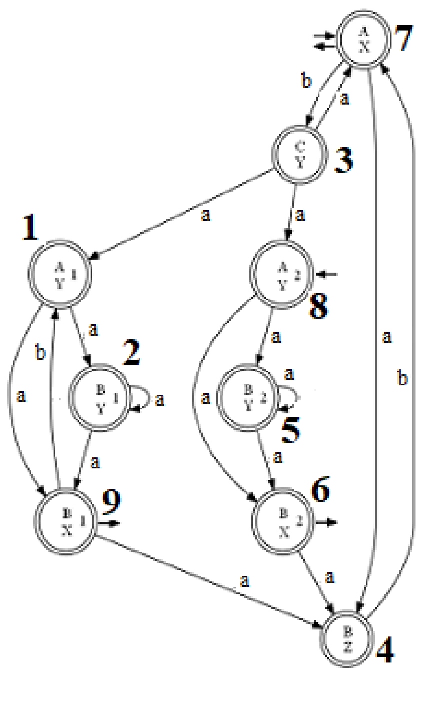

Figure 9. The *given* automaton (for the language $(a + ab + ba)^*$)

We should show, that the values of the two state-marking functions for all states coincide with those shown in the figure. We can do it in the following two ways:

- either complete the procedure of canonization for this automaton (like [13]);
- or for each state for both state-marking functions, strictly prove equality by specifying the "necessary"

[8] Generally speaking, there exist infinitely many accepting paths of the type (14), but the finite number of such states t' and t''.

[9] We especially designate the transition function as belonging to the automaton $\mathcal{BA}(L)$, although this is not entirely correct.

input (output) words and additionally proving that there are no "extra" input (output) words.

However, both these ways need a very long time, then we shall apply the following simple method. [10]

The inputs, outputs and edges of the automaton of Figure 9 are *the own subsets* of corresponding sets of the automaton of Figure 8; then both input and output words for all states of Figure 9 are also the own subsets for such ones of Figure 8. We know the state-marking functions for the first automaton; all their values consist of one element. Besides, the automaton of Figure 9 have neither useless nor inaccessible states, then each value of its state-marking function is not empty. Therefore, for each value we have the only possibility, which should be the same as shown in the figure.

We shall look at the last example quite briefly. Deleting the edge

$$ {}^{B}_{X}1 \xrightarrow[\mathcal{BA}(L)]{b} {}^{B}_{Z} $$

leads to almost the same comments that we had for the previous automaton. However, in this case, we can also consider a *new inter-component edge*

$$ {}^{B}_{X}1 \xrightarrow[\mathcal{BA}(L)]{b} {}^{A}_{Y}2, $$

instead of

$$ {}^{B}_{X}2 \xrightarrow[\mathcal{BA}(L)]{b} {}^{A}_{Y}2 $$

(the last one was already deleted for automaton on Figure 9). The last automaton for this language is depicted on Figure 10; its detailed consideration is beyond the scope of this paper.

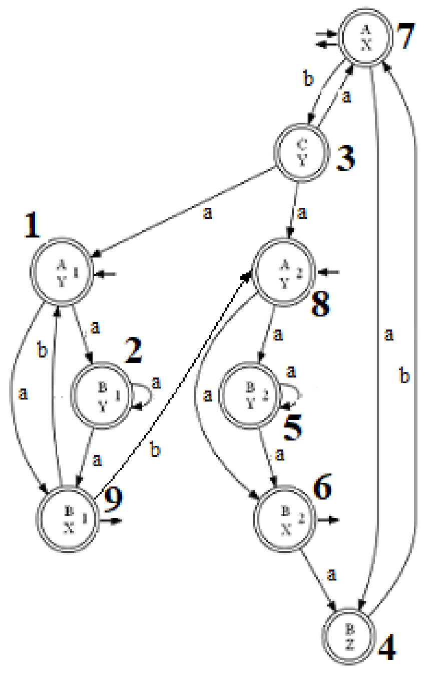

Figure 10. The *last* automaton (for the language $(a + ab + ba)^*$)

[10] Sometimes, we use this method in the computer programs, for example, in heuristic algorithms checking the equivalence of two automata, obtained by transformation of basis automata. However, this topic is beyond the scope of this paper.

On the Part III

In Part III, we are going to describe the algorithm for *special adding* a state. This algorithm will also have the same feature of transformations, i.e. the values of star-height for the obtained automata will be no more than such value for the given automaton.

Let us very briefly say, *why* we shall consider this case. We shall add *not only edges* (like combining case of Part I), but *also states*. The meaning of this addition is *not* the "urgent" combining states; we "improve the structure" of the automaton under consideration; this will help to subsequently apply one of the algorithms discussed in the first two parts. In other words, even the "adding" algorithm *does not "complicate"* the considered automaton; the details will be described later.

References

[1] Melnikov B. An approach to the classification of the loops of finite automata. Part I: Long corresponding loops // International Journal of Open Information Technologies. 2019, vol. 7, no. 4, pp. 1–5.

[2] Melnikov B., Melnikova A. Some properties of the basis finite automaton // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). 2002, vol. 9, no. 1. pp. 135–150.

[3] Melnikov B., Sayfullina M. On some algorithms for the equivalent transformation of nondeterministic finite automata // Izvestiya of Higher Educational Institutions. Mathematics. 2009, no. 4, pp. 67–72. (in Russian, `https://elibrary.ru/item.asp?id=11749888`)

[4] Melnikov B. Extended nondeterministic finite automata // Fundamenta Informaticae. 2010, vol. 104, no. 3, pp. 255–265.

[5] Melnikov B. The star-height of a finite automaton and some related questions // International Journal of Open Information Technologies. 2018, vol. 6, no. 7, pp. 1–5.

[6] Melnikov B., Melnikova A. An approach to the classification of the loops of finite automata. Part II: The classification of the states based on the loops // International Journal of Open Information Technologies. 2018, vol. 6, no. 11, pp. 1–6.

[7] Melnikov B., Melnikova A. Pseudo-automata for generalized regular expressions // International Journal of Open Information Technologies. 2018, vol. 6, no. 1, pp. 1–8.

[8] Harary F. *Graph Theory*. Addison Wesley (Boston), 1969, 274 p.

[9] Aho A., Ullman J. *The Theory of Parsing, Translation, and Compiling, Vol. 1: Parsing*. Prentice Hall (NJ), 1972, 560 p.

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

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

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

[13] 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.

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

Boris Feliksovich MELNIKOV,
Professor of Shenzhen MSU – BIT University, China
(`http://szmsubit.ru/`),
Professor of Russian State Social University
(`http://www.rgsu.net/`),
email: `bf-melnikov@yandex.ru`,
mathnet.ru: `personid=27967`,
elibrary.ru: `authorid=15715`,
scopus.com: `authorId=55954040300`,
ORCID: `orcidID=0000-0002-6765-6800`.