Pseudo-automata for generalized regular expressions

B. F. Melnikov, A. A. Melnikova

***Abstract*—In this paper, we introduce a new formalism which is intended for representing a special extensions of finite automata. We call them generalized nondeterministic finite pseudo-automata. This formalism gives not only the equivalence between two classes of finite automata, i.e., ordinary nondeterministic finite automata and pseudo-automata introduced by us, which also define all the regular languages. This formalism also gives an opportunity of defining the complement operation (and, therefore, generalized regular expressions) in a way similar to the usual "automata" methods.**

We use the term "pseudo-automata", because, unlike usual automata constructions (ordinary nondeterministic finite automata, push-down automata, Turing machines etc.), we do not indicate the concrete paths for defining the considered word of the given regular language; the introduced formalism gives only the algorithm for answering the question, whether the given word belongs to the considered language.

In the paper, we firstly give definition of the pseudo-automata and their languages. After that, we consider the diagrams allowing to visualize them and give some examples. Then we consider some properties of introduced formalism.

***Keywords*—nondeterministic finite automata, regular languages, generalized regular expressions, extension of automata.**

I. Introduction and motivation

In this paper, we propose a formalism used for representing a special class of extensions of finite automata; we call them generalized nondeterministic finite pseudo-automata. The considered algorithm of constructing such automata gives not only the equivalence between the classes of such automata and ordinary finite automata (because such equivalence is obvious a priori), but also the possibility of defining the complement operation (and, generally, the *generalized regular expressions*) by usual "automata" methods. We also consider a method of constructing the *concrete* generalized nondeterministic finite pseudo-automaton which defines the given generalized regular expression. [1]

This paper is supposed to be the first in a series of papers, and we intend to publish the properties of pseudo-automata in several subsequent publications. In this paper, we give the most general definitions and facts only.

In many papers published before, many extensions of nondeterministic finite automata (NFAs) were considered. [2] However, the authors do not know papers, where automata with the complement operation are considered, although such complement operation does not give the loss of regularity. The first author already proposed another approach for the same problem, see [6]. However, in the current paper we define an analog of the complement operation in a different way, which, apparently, is much more successful. [3] And, consequently, in this paper, we use completely other notation. For instance, we use the term "pseudo" because we usually *are not able to indicate the path* defining the given word (we are able to make this thing for ordinary NFAs, push-down automata, Turing machines, etc.); however, at the same time, this word can belong to the considered regular language.

Thus, we shall consider formalism for defining automata for the generalized regular expressions; these expressions are defined like [7]. In other words, we consider the complement operation by "usual automaton methods".

This paper has the following structure. In Section II, we propose the main definitions, i.e., definitions of generalized nondeterministic finite pseudo-automaton (GNFPA) and its language. Namely, we formally determine the corresponding generalized regular expression by the given pseudo-automaton. Therefore, we can in principle, on the basis of this definition only, solve all the usual problems of the theory of regular languages: we can simply obtain a regular expression for our pseudo-automaton, such that this expression would be equivalent (i.e., would define the same language) to the given generalized regular expression. Certainly, we will rarely apply this approach: we define GNFPAs for more complex problems.

One of such standard problems (for the general formal language theory) is checking whether the given word belongs to the language of the given automaton. For GNFPAs, we consider a trivial (but long working) algorithm for such checking in Section III. We are going to describe a more effective algorithm for this thing in the next paper.

In Section IV, we present a method of depicting such automata in the form of graphs and consider a detailed example. In Section V, we present the algorithm of construction of GNFPA by the given generalized regular expression.

In Section VI, we give some alternative models of pseudo-automata. Namely, we formulate: another definition of the language of the given GNFPA; a model of pseudo-automaton that can be regarded as an analog of canonical NFA; two other definitions of GNFPA (a simplified one and, vice versa, a complicated one). In that section and in Conclusion, we

Received December 27, 2017.

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

Aleksandra A. Melnikova, National Research Nuclear University "MEPhI" (email: super-avahi@yandex.ru).

[1] Remark that like ordinary nondeterministic finite automata, it can also define some other equivalent generalized regular expressions.

[2] Among these articles, there is important to mention papers of D. Kirsten, related to the topic of the present work. See [1], [2], etc.

Let us also mention three papers of the authors [3], [4], [5]. In them, *three different* extensions of NFA class were considered; in two of them, the edges were labeled by regular expressions.

[3] Let us remark in advance, that the main definition of the current paper (given in Section II) will describe a more general case than the approach given in [6], and, vice versa, one of the alternative definitions (a definition of Section VI) will describe a particular case of that approach.

also briefly formulate directions for further work on this topic.

II. The main definitions

Firstly let us define a generalized regular expression like [7] etc.; simultaneously, let us determine corresponding regular languages.

Definition 1: A *generalized regular expression (GRE)* on the given finite alphabet Σ is defined in the following way:

1) GRE $\varnothing$ defines regular language $\varnothing$;
2) GRE $\boldsymbol{\varepsilon}$ defines regular language $\{\varepsilon\}$;[4]
3) for each letter $a \in \Sigma$, GRE $\boldsymbol{a}$ defines regular language $\{a\}$.

Further, let p and r be two GRE[5] defining regular languages P and R respectively. Then:

4) GRE $\boldsymbol{(p+r)}$ defines regular language $P \cup R$;
5) GRE $\boldsymbol{(p \cdot r)}$ defines regular language $P \cdot R$;
6) GRE $\boldsymbol{(p^*)}$ defines regular language P^*;

(as for ordinary regular expression). Besides:

7) GRE $\boldsymbol{(\overline{p})}$ defines regular language $\overline{P}$, i.e., language $\Sigma^* \setminus P$.

Nothing more is a GRE.

For generalized regular expression $\mathcal{E}$, its language will be denoted by $\mathcal{L}(\mathcal{E})$

Also like [7] etc., we shall sometimes omit unnecessary parentheses and symbols "$\cdot$". $\square$

We shall not consider examples, they are well-known, see [7] etc.

Let us now define generalized nondeterministic finite automata and their languages.

Definition 2: A *generalized nondeterministic finite pseudo-automaton (GNFPA)* is a tuple

$$\boldsymbol{G} \;=\; \left(\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta}, \boldsymbol{S}, \boldsymbol{F}, \boldsymbol{T}, \boldsymbol{\zeta^{in}}, \boldsymbol{\zeta^{out}}\right), \qquad (1)$$

where:

- $\boldsymbol{Q}$ is the finite set of states;
- $\boldsymbol{\Sigma}$ is the considered alphabet;
- $\boldsymbol{\delta}$ is the transition function of the type

$$\delta : Q \times \left(\Sigma \cup \{\varepsilon\}\right) \to \mathcal{P}(Q)$$

  (where $\mathcal{P}(Q)$ means the superset for Q);
- $\boldsymbol{S} \subseteq Q$ is the set of initial states;
- $\boldsymbol{F} \subseteq Q$ is the set of final states

(as for ordinary nondeterministic finite automaton). Besides:

- $\boldsymbol{T} \subseteq \mathbb{N}$ is a finite set; [6]
- $\boldsymbol{\zeta^{in}}$: $T \to Q \times \mathcal{P}(Q)$ (the *push-function*); for some its element

$$\zeta^{in}(i) = (s_i, S_i),$$

  we shall write $s_i = \theta^{in}(i)$ and $S_i = \Theta^{in}(i)$;
- $\boldsymbol{\zeta^{out}}$: $T \to \mathcal{P}(Q) \times Q$ (the *pop-function*); for some its element

$$\zeta^{out}(j) = (F_j, f_j),$$

  we shall write $F_j = \Theta^{out}(j)$ and $f_j = \theta^{out}(j)$. $\square$

[4] Similar to most books on the formal languages theory, we shall consider this item, although expressions ε and defined below ($\varnothing^*$) are equivalent (under natural interpretations).

[5] Remark that they are *not* two ordinary regular expressions.

[6] We mean that $\mathbb{N}$ does not contain 0, i.e., $\mathbb{N} = \{1, 2, \dots\}$.

The set T and functions ζ^{in} and ζ^{out} define complement languages; below, we shall define such languages and, generally, language of a GNFPA. For this thing, let us firstly consider some ancillary definitions.

Definition 3: For the given NFA $K = (Q, \Sigma, \delta, S, F)$ and some subsets $S', F' \subseteq Q$, let us denote nondeterministic finite automaton $(Q, \Sigma, \delta, S', F')$ by $\boldsymbol{\mathcal{K}(K, S', F')}$. $\square$

Definition 4: For the given GNFPA G (1), let us denote corresponding ordinary nondeterministic finite automaton $(Q, \Sigma, \delta, S, F)$ by $\boldsymbol{\mathcal{K}(G)}$. $\square$

Definition 5: For the given GNFPA G defined by (1) and some subsets $S', F' \subseteq Q$, let us denote the GNFPA $(Q, \Sigma, \delta, S', F', T, \zeta^{in}, \zeta^{out})$ by $\boldsymbol{\mathcal{G}(G, S', F')}$. $\square$

Definition 6: Let G be a GNFPA (1), where $T \neq \emptyset$. For some $k \in T$, let us define a generalized nondeterministic finite automaton

$$G_{-k} = \left(Q, \Sigma, \delta, S, F, T_{-k}, \zeta^{in}_{-k}, \zeta^{out}_{-k}\right),$$

where:

- $T_{-k} = T \setminus \{k\}$;
- $\zeta^{in}_{-k}(i) = \zeta^{in}(i)$ for each $i \in T_{-k}$;
- $\zeta^{out}_{-k}(i) = \zeta^{out}(i)$ for each $i \in T_{-k}$.[7]

Below, we shall sometimes use functor names ζ^{in} and ζ^{out} instead of ζ^{in}_{-k} and ζ^{out}_{-k} respectively. $\square$

Let us present now the main definition of this paper (it is the recurrent definition).

Main Definition 1: Let the GNFPA G defined by (1) be given. Then we define its language $\mathcal{L}(G)$ in the following way.

- Let $L_1 = \mathcal{L}(\mathcal{K}(G))$.[8]
- If $T = \varnothing$, then we define $L_2 = \varnothing$.
- Otherwise, for each $k \in T$, let us consider three the following GNFPAs:
  1) $G_{\to k} = \mathcal{G}\left(G_{-k}, S, \{\theta^{in}(k)\}\right)$;
  2) $G_{[k]} = \mathcal{G}\left(G_{-k}, \Theta^{in}(k), \Theta^{out}(k)\right)$;
  3) and $G_{k\to} = \mathcal{G}\left(G_{-k}, \{\theta^{out}(k)\}, F\right)$.

  Then we define

$$L_2 = \bigcup_{i \in T} \left(\mathcal{L}(G_{\to i}) \cdot \overline{\mathcal{L}(G_{[i]})} \cdot \mathcal{L}(G_{i\to})\right). \qquad (2)$$

- The defined language $\mathcal{L}(G) = L_1 \cup L_2$. $\square$

Remark that this definition is correct: the cardinality of the set T for each of three GNFPAs $G_{\to k}$, $G_{[k]}$ and $G_{k\to}$ is less than the cardinality of the set T for GNFPA G. Therefore, we can assume that we *a priori* know the languages of these three GNFPAs, because we also defined the languages of all GNFPAs having $T = \varnothing$.

It is also very important to note the following fact. There is no analogy with the accepting path of an ordinary automaton: in the case of our automata, such path, generally speaking, simply does not exist. That is, we have not to define the language by an accepting path (as we do for ordinary NFAs, and also for push-down automata and Turing machines, [8] etc.); we do it in a different method. In this method, we *do not know* the order of the use of the elements of T, and, therefore, we have to assume the possibility of any order of their application as an initial definition.

[7] Remark that we define *all the necessary values* for $\zeta^{in}_{-k}(i)$ and $\zeta^{out}_{-k}(i)$.

[8] I.e., we consider in this case the ordinary nondeterministic finite automaton and the usual definition of its language.

As we already said in Introduction, in principle we can, on the basis of this definition only, solve all the usual problems of the theory of regular languages (because we can simply obtain a regular expression for our pseudo-automata, such that this expression would define the same language). Certainly, we will rarely apply this approach: we define GNFPAs for more complex problems. Some of such problems are considered in the remainder of the paper.

III. Algorithm for checking whether the word belongs to the language of GNFA

In this section, we give an obvious recursive algorithm for checking, whether the given word belongs to the language of GNFPA. It is clear that without the algorithm answering this question, it hardly makes sense to consider pseudo-automata: the method, briefly described at the end of previous section, connected with the reformulation this question for ordinary automata and ordinary regular expressions, can hardly be called acceptable.

We shall not consider the complexity of this algorithm because of the following. Firstly, the main its purpose is the fundamental possibility of the corresponding problem. Secondly, we propose to introduce significantly more efficient algorithm in the following papers.

Algorithm 1: Checking whether the word belongs to the language of GNFPA.

Input:

- GNFPA G given by Definition 2;
- word $u \in \Sigma^*$.

Output:

- `true`, if $u \in \mathcal{L}(G)$;
- `false`, if $u \notin \mathcal{L}(G)$.

We shall denote the result of calling of this algorithm (`true` or `false`) by `Checking`(G, u); here, we apply a recursive call. In addition, like Pascal language, we shall use notation `Checking` (without arguments) on the left side of the assignment operator, to return a value to the running algorithm. However, like C++ language (and unlike Pascal), we believe that the use of such an assignment operator leads to the *immediate completion* of the execution of the branch of the recursive algorithm.

Method:

- if $u \in \mathcal{L}(\mathcal{K}(G))$ then `Checking:=true`;
- if $T = \varnothing$ then `Checking:=false`;
- for each representation of the word u in the form $u = xyz$, we consider each $k \in T$; if for at least one of these variants, we obtain simultaneously:
  - $x \in G_{\to k}$, i.e., `Checking`$(G_{\to k}, x)$=`true`;
  - $y \in G_{[k]}$, i.e., `Checking`$(G_{[k]}, y)$=`true`;
  - and $z \in G_{k\to}$, i.e., `Checking`$(G_{k\to}, z)$=`true`

  (in these subitems, we make the recursive calls), then `Checking:=true`;
- otherwise, `Checking:=false`.

End of the description of Algorithm 1. □

The correctness of this algorithm immediately follows from Main Definition 1.

IV. Diagrams of GNFPAs and examples

In this section, we present two methods of depicting our pseudo-automata in the form of graphs and consider some examples. Like ordinary NFA, a GNFPA can be described by an oriented graph; however, the edges are here labeled by both letters of Σ and some integers. The sets Q, S, F and δ are represented like ones for ordinary NFA. Further, if $q \in \Theta^{in}(i)$, then we add the edge from $\theta^{in}(i)$ to q labeled by $-i$; similarly, if $q \in \Theta^{out}(i)$, then we add the edge from q to $\theta^{out}(i)$ labeled by $+i$ (we shall never omit the sign "+").

Remark also, that we can omit the set T (it consists of all the marks of corresponding edges), but, unlike ordinary NFAs, *we have to designate considered alphabet* Σ.

In our previous papers [3], [4], [6], [9] etc., we denote the states of ordinary automata (NFAs) by double circles, like, for example, [10]. In this paper, we shall use the same agreement (i.e., use double circles for NFAs which are *not* GNFPAs) and use single circles for "proper" GNFPAs.

We shall also use the following agreement distinguishing ordinary NFAs and GNFPAs: ordinary lines will show the ordinary edges (marked by letters of Σ, usually a and b), and dotted lines will show the elements of ζ^{in} and ζ^{out}.

Thus, for alphabet $\Sigma = \{a, b\}$, let us consider the following pseudo-automaton, depicted on Fig. 1 below. For this pseudo-automaton (let it be 1), we have:

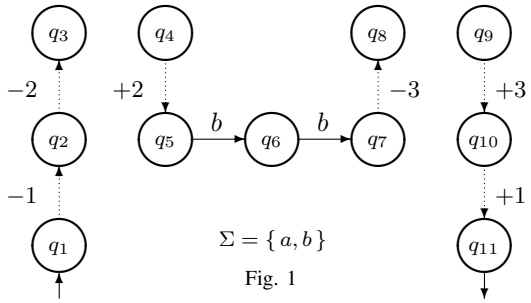


Fig. 1

- $Q = \{ q_1, \ldots, q_{11} \}$;
- $\Sigma = \{ a, b \}$ (remark once more, that we have to indicate the alphabet on the figure);
- $\delta = \left\{ q_5 \xrightarrow{b} q_6,\ q_6 \xrightarrow{b} q_7 \right\}$;
- $S = \{q_1\}$;
- $F = \{q_{11}\}$;
- $T = \{ 1, 2, 3 \}$;
- $\zeta^{in}(1) = (q_1, \{ q_2 \})$, $\zeta^{out}(1) = (\{ q_{10} \}, q_{11})$;
- $\zeta^{in}(2) = (q_2, \{ q_3 \})$, $\zeta^{out}(1) = (\{ q_4 \}, q_5)$;
- $\zeta^{in}(3) = (q_7, \{ q_8 \})$, $\zeta^{out}(1) = (\{ q_9 \}, q_{10})$.

(Below, we shall not use such detailed applications of descriptions.)

Now, we shall use Main Definition 1. Evidently, we have here $\mathcal{L}(\mathcal{K}(G)) = \varnothing$, then for defining $\mathcal{L}(G)$, we have to consider the following automata. More precisely, denoting the sequences in *square* brackets (in order not to confuse them with the notation of formulas), we will consider 6 possible sequences of applying 3 elements of the set T:

$$[1,2,3],\ [1,3,2],\ [2,1,3],\ [2,3,1],\ [3,1,2],\ [3,2,1].$$

We shall use the possible sequences of applying these elements in the title of subsections. We shall also apply some incomplete sequences for this thing, e.g., $[1]$, $[1,2]$, etc.

Sequence [1]

- We have $i = 1$, then $\theta^{in}(1) = q_1$, then pseudo-automaton $G_{\to 1}$ is depicted on the following Fig. 2:

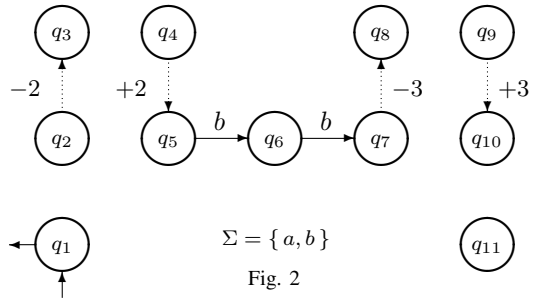


Fig. 2

There is evident, that $\mathcal{L}(G_{\to 1}) = \{\varepsilon\}$.

- Similarly, $\theta^{out}(1) = q_{11}$, then for automaton $G_{1\to}$, we also obtain $\mathcal{L}(G_{1\to}) = \{\varepsilon\}$.
- Therefore, we have to consider the following pseudo-automaton $G_{[1]}$ (Fig. 3 below), which have the only initial state q_2 (because for G, we have $\Theta^{in}(1) = \{q_2\}$):

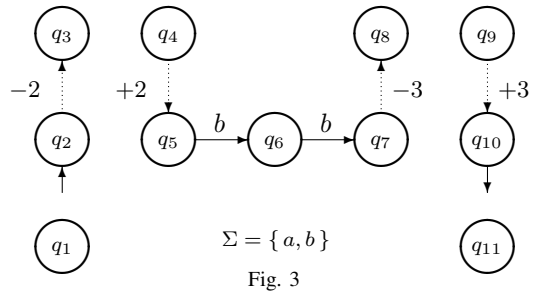


Fig. 3

We have to use the complement of its language as *the subset* of the language of the given pseudo-automaton G.

Further, for related figures (i.e., for sequences $[1, 2]$ and $[1, 3]$), we shall not depict states q_1 and q_{11}.

We shall continue to consider this case (i.e., sequence $[1]$) after studying its subcases (sequences $[1, 2]$, $[1, 2, 3]$, $[1, 3]$ and $[1, 3, 2]$).

Sequence [1,2]

- For this sequence, we have now $i = 2$, then $\theta^{in}(2) = q_2$, then pseudo-automaton $(G_{[1]})_{\to 2}$ is depicted on the following Fig. 4:

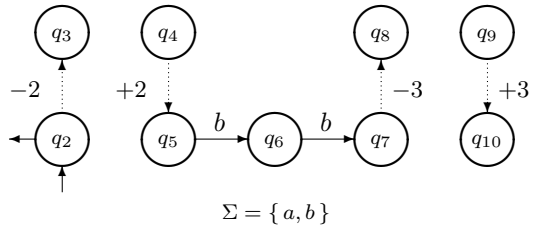


Fig. 4

There is evident, that $\mathcal{L}((G_{[1]})_{\to 2}) = \{\varepsilon\}$.

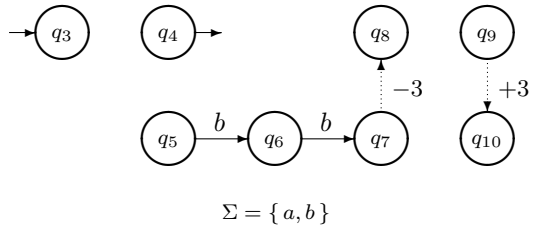


Fig. 5

- $\Theta^{in}(2) = \{q_3\}$, $\Theta^{out}(2) = \{q_4\}$, then pseudo-automaton $(G_{[1]})_{[2]}$ is depicted on Fig. 5 given before. (We omit state q_2. Remark that we also could omit some other states.)

  Evidently, the language of this automaton is $\varnothing$, then the complement of this language (used in our definitions) is Σ^*.
- $\theta^{out}(2) = q_5$, then automaton $(G_{[1]})_{2\to}$ is depicted on Fig. 6 below; like before, we omit some states, not affecting the language.

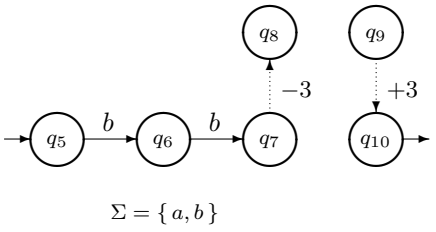


Fig. 6

The languages of this pseudo-automaton will be obtained below, by considering sequence $[1, 2, 3]$.

Sequence [1,2,3]

Let the considered automaton (Fig. 6) be G'.

- Evidently, we obtain language $\mathcal{L}(G'_{\to 3}) = \{bb\}$.
- Like pseudo-automaton $(G_{[1]})_{[2]}$, we obtain that $\mathcal{L}(G'_{[3]}) = \{\varepsilon\}$, and the complement of this language is Σ^*.
- And, evidently, $\mathcal{L}(G'_{3\to}) = \{\varepsilon\}$.

Sequences [1,3] and [1,3,2]

These cases are very similar to the discussed above cases for sequences $[1, 3]$ and $[1, 3, 2]$. Moreover, we obtain the same languages. Therefore, we shall not consider these cases.

Sequence [1], the completion

Let us finish considering the language which obtained where we apply $i = 1$ in Main Definition 1 (i.e., in (2)).

For pseudo-automaton $G_{[1]}$ (Fig. 3) we obtained, that its language is

$$\mathcal{L}(G_{[1]}) = \Sigma^* \cdot b \cdot b \cdot \Sigma^*;$$

then

$$\mathcal{L}(G) \supseteq \{\varepsilon\} \cdot \overline{\Sigma^* \cdot b \cdot b \cdot \Sigma^*} \cdot \{\varepsilon\} = \overline{\Sigma^* \cdot b \cdot b \cdot \Sigma^*}. \qquad (3)$$

Sequences [2] and [3]

Let us consider the case of sequence $[2]$ (the case of sequence $[3]$ is similar). For it, there is sufficient to consider pseudo-automaton $G_{\to 2}$ only. It is the following one:

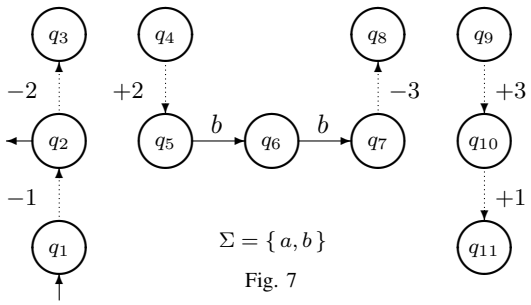


Fig. 7

We can simply obtain, that its language is empty: states q_1 and q_2 have no connecting paths, they have "connecting" ζ^{in}-element only. From here, it is easy to obtain, that all the languages resulting for sequences [2] and [3] are $\varnothing$, and, therefore, in (3) we can change the sign "$\supseteq$" for "$=$".

Thus, the obtained language for considered pseudo-automaton G is

$$\overline{\Sigma^* \cdot b \cdot b \cdot \Sigma^*}. \tag{4}$$

It is the language including all the words over alphabet $\{a, b\}$ that *do not include* two characters b in a row in neighboring positions. We can see that *for this language*, the ordinary automaton (a NFA defining it) is simpler (Fig. 8):

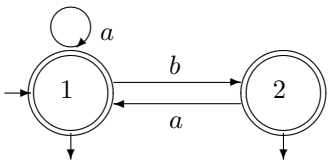


Fig. 8

However, certainly, there exist languages, for which a GRE is simpler then a regular expression.

V. The algorithm of construction of GNFPA by the given generalized regular expression

In this section, we believe the generalized regular expression be given and present the algorithm constructing corresponding pseudo-automaton, i.e., a GNFPA which language coincides with the language of the given GRE.

For ordinary nondeterministic finite automata and regular expressions, a similar version of building NFA is known since the 1950s; see, for example, the description of the classical version of such construction in [7].

And in this section, we transfer this method for GREs and GNFPAs. We make building in roughly the same way as in [6]. Like the classical construction, the automaton corresponding to the regular expression is constructed recursively according to the items of Definition 1; certainly, we have to add an item corresponding to the complement operation. Thus, we explicitly indicate the automaton (that is, we use the constructive definition for the GNFPA).

As we already noted in Introduction, a different definition of our formalism was used in [6]. However, we are able to use figures of [6], because the formalism defined there uses only the special cases of the above definitions; it also can be considered as a special case of the formalism defined in the present paper. Therefore, we do not repeat the figures from [6] in this paper [9] and consider the figure for automaton describing GRE $\overline{p}$ only.

Like ordinary regular expressions and ordinary NFAs, we have to indicate the specific GNFPA which language coincides with the language of the given GRE. But we cannot use completely unchanged construction of [7] etc., because we have to define a GNFPA, not a NFA; however, this does not complicate the presentation. We also note in advance that in each item there is evident the proof, that the presented GNFPA actually determines the required GRE.

Thus, let us consider the following algorithm.

[9] Let us only remark, that unlike [6], we marked here elements of ζ^{in} by negative numbers and elements of ζ^{out} by positive numbers.

Algorithm 2: Constructing corresponding generalized nondeterministic finite pseudo-automaton by the given generalized regular expression.

Input: GRE $\mathcal{E}$.

Output: a GNFPA G, such that $\mathcal{L}(G) = \mathcal{L}(\mathcal{E})$.

Method: Let us consider items corresponding to Definition 1.

1) GRE $\varnothing$ corresponds to pseudo-automaton

$$\big(\{s, f\}, \Sigma, \varnothing, \{s\}, \{f\}, \varnothing, \varnothing, \varnothing\big)$$

(here and below, as before, we consider functions δ, ζ^{in} and ζ^{out} as the sets of their elements, and, therefore, we use the sign $\varnothing$; set T is also empty);

2) GRE ε corresponds to pseudo-automaton

$$\big(\{s, f\}, \Sigma, \delta, \{s\}, \{f\}, \varnothing, \varnothing, \varnothing\big),$$

where

$$\delta = \Big\{\, s \xrightarrow{\varepsilon} f \,\Big\};$$

3) for each letter $a \in \Sigma$, GRE a corresponds to pseudo-automaton

$$\big(\{s, f\}, \Sigma, \delta, \{s\}, \{f\}, \varnothing, \varnothing, \varnothing\big),$$

where

$$\delta = \Big\{\, s \xrightarrow{a} f \,\Big\}.$$

Further, let p and r be two GRE corresponding to pseudo-automata

$$G_p = \big(Q_p, \Sigma, \delta_p, S_p, F_p, T_p, \zeta_p^{in}, \zeta_p^{out}\big)$$

and

$$G_r = \big(Q_r, \Sigma, \delta_r, S_r, F_r, T_r, \zeta_r^{in}, \zeta_r^{out}\big)$$

respectively. Let us remark, that according to the above definitions, we can assume, that $T_p \cap T_r = \varnothing$. Then:

4) GRE $(p + r)$ corresponds to pseudo-automaton

$$G_{p+r} = \big(Q_p \cup Q_r, \Sigma, \delta_p \cup \delta_r, S_p \cup S_r, F_p \cup F_r, T_p \cup T_r, \zeta_p^{in} \cup \zeta_r^{in}, \zeta_p^{out} \cup \zeta_r^{out}\big);$$

5) GRE $(p \cdot r)$ corresponds to pseudo-automaton

$$G_{p\cdot r} = \big(Q_p \cup Q_r, \Sigma, \delta_p \cup \delta_r \cup \delta_\cup, S_p, F_r, T_p \cup T_r, \zeta_p^{in} \cup \zeta_r^{in}, \zeta_p^{out} \cup \zeta_r^{out}\big),$$

where

$$\delta_\cup = \Big\{\, f_p \xrightarrow{\varepsilon} s_r \,\Big|\, f_p \in F_p,\ s_r \in S_r \,\Big\};$$

6) GRE (p^*) corresponds to pseudo-automaton

$$G_{p^*} = \big(Q_p \cup \{q\}, \Sigma, \delta_p \cup \delta_\cup, \{q\}, \{q\}, T_p, \zeta_p^{in}, \zeta_p^{out}\big),$$

where $q \notin Q$ is a new state, and

$$\delta_\cup = \Big\{\, q \xrightarrow{\varepsilon} s_p \,\Big|\, s_p \in S_p \,\Big\} \cup \Big\{\, f_p \xrightarrow{\varepsilon} q \,\Big|\, f_p \in F_p \,\Big\}$$

(like ordinary regular expression and usual constructions close to the matter of [7] etc. [10]). Besides:

7) GRE $(\overline{p})$ corresponds to pseudo-automaton

$$G_{\overline{p}} = \big(Q_p \cup \{s, f\}, \Sigma, \delta_p, \{s\}, \{f\}, T_p \cup \{t\}, \zeta_p^{in} \cup \zeta_\cup^{in}, \zeta_p^{out} \cup \zeta_\cup^{out}\big),$$

[10] Certainly, we also add three elements of the tuple (1).

where $s, f \notin Q$ are new states, $t \notin T_p$ is a new element, and

$$\zeta^{in}_{\cup}(t) = \big(s, S_p\big), \quad \zeta^{out}_{\cup}(t) = \big(F_p, f\big).$$

(Like before, we consider functions ζ^{in} and ζ^{out} as sets of their elements. Namely, we defined here function $\zeta^{in}_{\overline{p}}$ containing all the elements of function ζ^{in}_p and also the specified here elements for its new argument t (i.e., set $\zeta^{in}_{\cup}$). Similarly for ζ^{out}_p, see Fig. 9.)

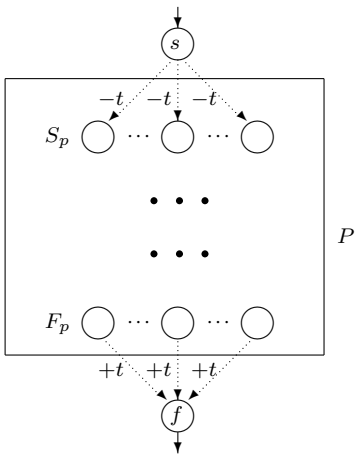


Fig. 9

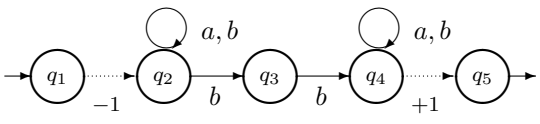
End of the description of Algorithm 2. □

As we already said, the proofs of the correctness of constructions are simple for each item. Let us formulate this fact by the following proposition.

Proposition 1: Using Algorithm 2, we obtain a GNFPA defining the given GRE. [11]

Proof. Items 1, 2, 3 and 6 are evident. In items 4 and 5, sets T of the parts of automata (i.e., sets P and R of automata T_P and T_R) do not affect the languages of the other part of the same automaton. In item 7, we have the only possibility for defining the word: i.e., we have to use t in (2) for defining a word. Then we obtain the complement of language of automaton P. □

Let us remark, that the precise application of Algorithm 2 gives automaton which is significantly different from the above (Fig. 1). Therefore, having at the input GRE (4) considered before, we obtain automaton, which after trivial simplifications becomes the following one (Fig. 10):



$\Sigma = \{\, a, b \,\}$

Fig. 10

[11] That is, we really define this expression, and *not just equivalent to it*, as usually in such proofs. The *informal explanation* of this fact is the following. Firstly, we explicitly indicate the order of the automata used in applying the operation $+$. Secondly, the process of constructing the pseudo-automaton described here explicitly determines the only possible order of appearance of elements of set T (for the obtained automaton).

VI. The alternative models and some directions for further researches

The title of this section includes the word "models", and we do present *four models different from one considered above*. However, each time (i.e., for each model), we mean different things:

1) we consider another definition of the language of the given GNFPA, which gives the equivalent language;
2) we consider the GNFPA which is obtained based on the canonical NFA for the given language;
3) we consider another (simplified) definition of the functions ζ^{in} and ζ^{out} of the given GNFPA (and, consequently, also another definition of the language of the pseudo-automaton);
4) vice versa, for the same functions ζ^{in} and ζ^{out} we also consider another (complicated) definition of the language of the given GNFPA.

The main goal of the section is to give definitions of models slightly different from those introduced above and to formulate the main directions of the most detailed studies. Let us also remark, that an alternative model can be considered, i.e.,

5) the definitions introduced in [6].

1) The alternative definition of the language

In this subsection, we give the alternative definition of the language of GNFPA.

Firstly, let us consider some notation for permutations.

Definition 7: For a permutation $P = (p_1, p_2, \ldots, p_n)$, where $n \geq 2$, let us denote:

- $\alpha(P) = p_1$; i.e., the head considered as an element;
- $\Omega(P) = (p_2, p_3, \ldots, p_n)$; i.e., the tail considered as a sequence, which is not empty. □

Definition 8: Let GNFPA G be considered according to Definition 2. For a permutation $P = (p_1, p_2, \ldots, p_n)$, where $n \geq 1$, let us define language

$$\mathcal{L}_P(G) = \mathcal{L}\big(\mathcal{K}(G)\big) \ \cup \ L_1 \cdot \overline{L_2} \cdot L_3\,,$$

where languages L_1, L_2 and L_3 are defined in the following way:

- if $n = 1$, i.e., $P = (p)$ for simplicity, then

$$L_1 = \mathcal{L}\big(\mathcal{G}\big(G_{-p}, S, \{\theta^{in}(p)\}\big)\big),$$
$$L_2 = \mathcal{L}\big(\mathcal{G}\big(G_{-p}, \Theta^{in}(p), \Theta^{out}(p)\big)\big),$$
$$L_3 = \mathcal{L}\big(\mathcal{G}\big(G_{-p}, \{\theta^{in}(p)\}, F\big)\big);$$

- if $n \geq 2$, then

$$L_1 = \mathcal{L}_{\Omega(P)}\big(\mathcal{G}\big(G_{-\alpha(P)}, S, \{\theta^{in}(\alpha(P))\}\big)\big),$$
$$L_2 = \mathcal{L}_{\Omega(P)}\big(\mathcal{G}\big(G_{-\alpha(P)}, \Theta^{in}(\alpha(P)), \Theta^{out}(\alpha(P))\big)\big),$$
$$L_3 = \mathcal{L}_{\Omega(P)}\big(\mathcal{G}\big(G_{-\alpha(P)}, \{\theta^{in}(\alpha(P))\}, F\big)\big). \quad \square$$

In the next paper, we shall prove that two considered definitions (this one and Main Definition 1) are equivalent, i.e., they define the same language. As we said before, we do not obtain the estimates of the complexity of the algorithms formulated before; we are going to do this thing in the next paper. However, there is evident, that both algorithms corresponding to this model (i.e., algorithm for obtaining corresponding GRE and algorithm checking, whether the word belongs to the GNFPA) are much less complex than algorithms corresponding to Main Definition 1.

2) Pseudo-automaton obtained on the base of the canonical NFA

In this approach, all the above definitions are retained in their original form. We consider *ordinary canonical* NFAs with some elements added for functions ζ^{in} and ζ^{out} (as before, we consider them as the set of their elements). Let us briefly describe the process of constructing such an automaton.

1) We obtain ordinary canonical NFA. But unlike most of our previous papers ([3], [4], [11], [12]), we consider here canonical NFA which *may* have so called "dead" state; see the details in our quoted papers, for instance, [9, Footnote 1]. Let the set of canonical NFA be Q_π, also like [9].
2) We choose an arbitrary function that assigns to different natural numbers (the set of these numbers will be T) all different ordered pairs of states of the constructed automaton.
3) For each such pair (let it be (q, q') for the chosen $t \in T$), we add to function ζ^{in} (it again is considered as the set) element $\zeta^{in}(t) = (q, \{q\})$.
4) For the considered pair, we also add to function ζ^{out} element $\zeta^{out}(t) = (Q_\pi \setminus \{q'\},\, q')$.

As we said before, we are going to continue this model in some next papers. In particular, we prove the correctness of our constructions (i.e., that we really define the regular language we need) and also describe some properties of the obtained automata. For now, we will briefly consider an example.

In this example, we will continue to explore the language considered in the above examples of this paper. We can assume, that automaton of Fig. 8 is a canonical one; and on the following Fig. 11, we give the same automaton with redesignation of its stats by A and B (like our usual notation) and with addition of the "dead" state N:

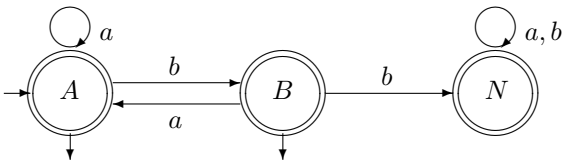


Fig. 11

Among the 9 elements of the set being formed, let us select 2 ones, (A, B) and (B, N); apparently, it is most convenient to denote the corresponding elements of set T by 2 and 6 (because A is the first element, then (A, A) is the first pair, $\ldots$, (N, N) is the last pair). Let us consider the following Fig. 12; that is automaton of Fig. 11 with the addition of the usual notation for functions ζ^{in} and ζ^{out}, corresponding to the said elements 2 and 6 of set T.

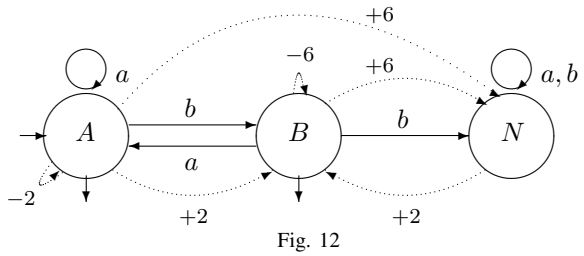


Fig. 12

(We can even assume here, that $T = \{\, 2, 6 \,\}$, not showing in the figure its other elements. Remark also once again, that we denote the states of ordinary NFAs by double circles, and the states of ordinary GNFPAs by single circles.)

3) The simplified definition of pseudo-automaton

Let us distinguish the simplified model from our basic approach.

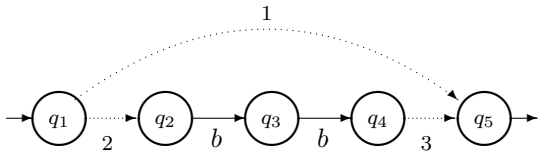
- In Definition 2, we change both functions ζ^{in} and ζ^{out} for functions of the type

$$\zeta^{in}, \zeta^{out} : T \to Q\,;$$

- In Main Definition 1, the three auxiliary pseudo-automata are the following ones:
  1) $G_{\to k} = \mathcal{G}\left(G_{-k}, S, \{\zeta^{in}(k)\}\right)$;
  2) $G_{[k]} = \mathcal{G}\left(G_{-k}, \{\zeta^{in}(k)\}, \{\zeta^{out}(k)\}\right)$;
  3) and $G_{k\to} = \mathcal{G}\left(G_{-k}, \{\zeta^{out}(k)\}, F\right)$.

All other definitions remain as before.

In this model, it is more convenient to use slightly modified principles for arrows and symbols in the pictures. Because values of functions ζ^{in} and ζ^{out} are single states (not set of states), then for each $t \in T$, it is more convenient to write arrows from $\zeta^{in}(t)$ to $\zeta^{out}(t)$ marking simply t (neither $-t$, nor $+t$). Let us consider an example, Fig. 13:



$\Sigma = \{\, a, b \,\}$

Fig. 13

We can prove that for the definition given here, the pseudo-automaton accepts the same language that we considered all the above in this paper. [12]

4) The complicated definition of pseudo-automaton

As the last model, let us very briefly distinguish the complicated model from our basic approach. In fact, there is the only (but extremely important) difference, i.e., in Main Definition 1, the three auxiliary pseudo-automata are the following ones:

1) $G_{\to k}$ is changed for NFA $\mathcal{K}\left(G, S, \{\zeta^{in}(k)\}\right)$;
2) $G_{[k]} = \mathcal{G}\left(G, \{\zeta^{in}(k)\}, \{\zeta^{out}(k)\}\right)$;
3) $G_{k\to}$ is changed for NFA $\mathcal{K}\left(G, \{\zeta^{out}(k)\}, F\right)$.

Remark that the first automaton and the last one are written even easier than ones of the basic model; certainly, their languages are languages of ordinary NFAs. However, changing G_{-k} for G in the second automaton greatly complicates all the corresponding algorithms: we can apply this recursive definition arbitrarily many times, but the definition is correct. In this paper, we shall not consider the resulting languages of this model.

[12] It is very important to note that using this approach, we obtain *non-empty* languages corresponding, for example, to sequence $[2, 1, 3]$ (not only for sequences $[1, 2, 3]$ and $[1, 3, 3]$, as in our main approach). However, for each such sequence, we obtain here the same language.

As we said before, we only described the models in this section. In some next papers, we are going consider them in detail, in particular, to prove the facts briefly formulated in this paper.

VII. Conclusion

We can say that in this paper, we have only described the terminology and proved the simple statements. In new paper, we are going to apply methods, which are similar to those we considered in previous papers:

1) the consideration of another (equivalent) definition of the language of the given GNFPA uses the methods we applied in [5], [6], [13], [14];
2) the consideration of "canonical GNFPA for the given language" uses the methods we applied in [6], [11], [12];
3) the consideration of "simplified definition" uses the methods we applied in [4], [15];
4) and the consideration of "complicated definition" uses the methods we applied in [3], [6], [13].

References

[1] Kirsten D. and Mäurer I. *On the determinization of weighted automata*. Journal of Automata, Languages and Combinatorics. 2005, vol. 10, no. 2/3. pp. 287–312.

[2] Kirsten D. *Distance desert automata and the star height problem*. Informatique Théorique et Applications. 2005, vol. 39, no. 3. pp. 455–509.

[3] Melnikov B. and Vakhitova A.. *Some more on the finite automata*. The Korean Journal of Computational and Applied Mathematics. 1998, vol. 5, no. 3. pp. 495–506.

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

[5] Vylitok A., Zubova M. and Melnikov B. *Ob odnom rasshirenii klassa konechnyh avtomatov ... [About one extension of the class of finite automata for accepting context-free languages]*. Vestnik of Moscow University. Series 15: Computational mathematics and cybernetics. 2013, no. 1. pp. 39–45. (in Russian)

[6] Baumgärtner S. and Melnikov B. *Obobschennye nedeterminirovannye konechnye avtomaty [Generalizing nondeterministic finite automata]*. Izvestiya of Higher Educational Institutions. Volga Region. Phisical and Mathematical Sciences, 2013, no. 2 (26), pp. 64–74. (in Russian)

[7] Salomaa A. *Jewels of Formal Language Theory*. Computer Science Press, 1981, 144 p.

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

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

[10] Brauer W. *Automatentheorie. Eine Einführung in die Theorie endlicher Automaten*. Stuttgart: B. G. Teubner, 1984, 493 S. (in German)

[11] Melnikov B. and Melnikova A. *Edge-minimization of nondeterministic finite automata*. The Korean Journal of Computational and Applied Mathematics. 2001, vol. 8, no. 3. pp. 469–479.

[12] Melnikov B. and Melnikova A. *Mnogoaspektnaya minimizaciya nedeterminirovannyh konechnyh avtomatov ... [Multi-aspect minimization of nondeterministic finite automaton (Part I. Auxiliary facts and algorithms)]*. Izvestiya of Higher Educational Institutions. Volga Region. Phisical and Mathematical Sciences, 2011, no. 4, pp. 59–69. (in Russian)

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

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

[15] Melnikov B. and Sayfullina M. *O nekotoryh algoritmah ekvivalentnogo preobrazovaniya nedeterminirovannyh konechnyh avtomatov [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)