PROCEEDINGS OF THE YEREVAN STATE UNIVERSITY

Physical and Mathematical Sciences 2021, **55**(1), p. 72–80

Informatics

MEASUREMENT OF DISTANCE BETWEEN REGULAR EVENTS FOR MULTITAPE AUTOMATA BASED ON A NEW CHARACTERIZATION OF EQUIVALENCE CLASSES

T. A. GRIGORYAN *, M. S. HAYRAPETYAN **

IT Educational and Research Center, YSU, Armenia

In this paper several problems related to the implementation of the method for the approximate calculation of distance between regular events for multitape finite automata are considered and resolved. An algorithm of matching for the considered regular expressions is suggested and results of the algorithm application to some specific regular expressions are adduced. The proposed method can be used not only for the mentioned implementation, but also separately.

https://doi.org/10.46991/PYSU:A/2021.55.1.072

MSC2010: Primary: 68Q45; Secondary: 68W32.

Keywords: regular expressions, distance of regular events, string matching, multitape finite automata.

Introduction. Regular expressions have a broad variety of applications and are supported in many programming languages. It is natural that the consideration of commutativity between some letters may additionally increase the range of application of regular expressions.

Partially commutativity in the input alphabet is naturally related to the multitape finite automata (MFA) [1], for which any two letters of different tapes are commutative with each other, and any two letters of the same tape are not commutative.

A new representation of languages for multitape finite automata (MFA), based on a special binary coding of elements in a free partially commutative semigroup were considered in [2]. This coding was also used for the solution of several problems in the theory of automata, which were previously open [2–4].

Regular expressions and events, a metric space of regular events for MFA, a problems of multitape finite automaton analysis resulting in a corresponding regular expression as well as synthesis of a multitape finite automation from a given regular expression for MFA were considered in [5]. We note that initially the notion of a metric space of regular events for one-tape automata was introduced in [6].

* E-mail: tigran.grigoryan1995@gmail.com
** E-mail: hayrapetyanmurad@gmail.com

It is well-known that the equivalence problem for MFA is unsolvable [1], and it follows from the above that for the introduced metric determination of the distance between two regular expressions is unsolvable too, because otherwise it will be possible to determine the equivalence of two MFA via checking whether the distance between them is equal to 0.

An attempt of proposing an approximate method for calculating the distance between the regular events for a multitape automata was done in [7]. The distance was based on the Euclidean distance, due to an isomorphism between elements of a free partially commutative semigroup and a sub-semigroup of $\mathbb{R}^n$. In this paper, we investigate the approximate method of calculating the distance proposed in [5]. This metric is based on a new characterization of commutation classes of a free partially commutative semigroup.

The implementation in C++ emphasizes the parts of the method, which essentially impact its complexity. A justification of the provided solutions is discussed. Results of some runs of the implementation are adduced too.

Both modules for distance calculation between two given regular expressions and for matching an input string with a given regular expression along with test results on the provided benchmark are available from https://github.com/HayMurad/multitape_regex.

The paper comprises of 5 sections. The definitions of free partially commutative semigroups, regular expressions and regular events over those semigroups, and multitape finite automata are formulated in Section 2. Extensions on multitape case of some already known metrics for languages accepted by one-tape automata, and a new metric are considered in Section 3. The algorithm for calculating the approximate distance for given regular events in a free partially commutative semigroup and the algorithm for string matching for regular expressions are represented in Section 4. Final conclusions are given in Section 5.

Preliminaries. Recall some definitions from [5].

If X is an alphabet, then the set of all words in the alphabet X, including the empty word, will be denoted by X^*, and the set of all n-element tuples of words will be denoted by $(X^n)^*$.

Let G be a semigroup with a unit, generated by the set of generators $Y = \{y_1, y_2, \ldots, y_n\}$. G is called a free partially commutative semigroup, if it is defined by a finite set of relations of type $y_i y_j = y_j y_i$.

Let $K : Y^* \to (\{0,1\}^n)^*$ be a homomorphism on the set F_Y, which maps words from Y^* to n-element vectors in the binary alphabet $\{0,1\}$. The homomorphism K over the set of symbols of the set F_Y is defined by the equation:

$$K(y_i) = (a_{1i}, \ldots, a_{ni}), \quad \text{where} \quad a_{ij} = \begin{cases} 1, & i = j, \\ e, & y_i y_j = y_j y_i, \\ 0, & y_i y_j \neq y_j y_i. \end{cases}$$

At the same time $K(e) = (e, \ldots, e)$.

$K(y_i y_j)$, $i \neq j$, is defined as a left concatenation:

$$K(y_i y_j) = (a_{1j} a_{1i}, \ldots, a_{nj} a_{ni}).$$

The homomorphism K can be considered as a mapping not only on Y^*, but also on the free partially commutative semigroup G.

Any element in $K(G)$ can be represented in the form $k_{i_1}\dots k_{i_m}$, where $\forall j$ $k_{i_j} \in K(Y)$. As an element may have multiple such representations, we define an equivalence relation ρ_K on $K(Y)^*$, namely, if $p,q \in K(Y)^*$, then $p\rho_K q$ iff p and q are the representations of the same element in $K(G)$. The equivalence class of p is denoted by $[p]$.

For a free partially commutative semigroup G with the set of generators Y, a regular event and a regular expression over $K(Y)$ are defined as follows:

1. $\emptyset$ (empty set) is a regular event in $K(Y)$.
2. $\tilde{E} = \{[e,\dots,e]\}$ is a regular event in $K(Y)$.
3. $\forall y \in Y, \{[K(y)]\}$ is a regular event in $K(Y)$.
4. If P and Q are regular events in $K(Y)$, then so are:
   (a) $P+Q = P \cup Q$;
   (b) $PQ = \{[s] \,|\, s = pq, [p] \in P, [q] \in Q\}$, where pq is the concatenation of p and q as defined above;
   (c) $P^* = \bigcup\limits_{n\geq 0} P^n$, where $P^0 = \tilde{E}$, $P^n = PP^{n-1}$ for $n \geq 1$.
5. There are no any other regular events in $K(Y)$.

1. $\emptyset$ is a regular expression, denoting the regular event $\emptyset$.
2. $K(e) = (e,\dots,e)$ is a regular expression, denoting the regular event $\tilde{E}$.
3. $\forall y \in Y, K(y)$ is a regular expression, denoting the regular event $\{[K(y)]\}$.
4. If p and q are regular expressions in $K(Y)$, denoting regular events P and Q correspondingly, then so are
   (a) $p+q$, denoting the regular event $P+Q$;
   (b) pq, denoting the regular event PQ;
   (c) $p^* = \bigcup\limits_{n\geq 0} p^n$, where $p^0 = K(e)$, $p^n = pp^{n-1}$ for $n \geq 1$ denoting the regular event P^*.
5. There are no any other regular expressions in $K(Y)$.

This definition of regular event differs from the known definition of regular events by the definition of "concatenation" operation, and by being defined as a set of equivalence classes. Further we will simply say a word p belongs to the regular event S, if $[p] \in S$.

Two models of multitape finite automata are necessary for consideration.

Let Q be a finite set of states, X be an input alphabet, $\delta : Q \times X \to 2^Q$ be the transition function, $q_0 \in Q$ be the initial state and $F \subseteq Q$ is the set of final states.

If X can be divided into disjoint, ordered subsets $X = X_1 \cup \dots \cup X_n$ such that $X_i \bigcap\limits_{i\neq j} X_j = \emptyset$ and $\forall x,x'$ $(x \in X_i, x' \in X_j (i \neq j), xx' = x'x)$, then it is called a partially commutative alphabet. Each subset X_i corresponds to i-th tape. Further, in this paper we will only consider partially commutative alphabets.

Definition 1. (Rabin-Scott model [1]). *Let $T : Q \to \{1,\ldots,n\}$ be a tape function, which associates each state from Q with a certain tape. An n-tape automaton is called a tuple $A = (Q,T,X,\delta,q_0,F)$, where $Q = \bigcup_{i=1}^{n} Q_i$ such that $Q_i = \{q | q \in Q, T(q) = i\}$ $\forall i = 1,\ldots,n$.*

Definition 2. (Mixed-state model [8]). *An n-tape automaton is called a tuple $A = (Q,X,\delta,q_0,F)$.*

The mixed-state model differs from the classical Rabin-Scott model by not bounding each state to a specific tape.

An automaton is called deterministic in Rabin-Scott model (DMFA), if $\forall i,\ \forall q \in Q_i,\ x \in X_i\ |\delta(q,x)| \leq 1$, otherwise, it is called nondeterministic (NMFA).

An automaton in Mixed-state model is called deterministic (DMFA-MSM), if $\forall q \in Q,\ x \in X\ |\delta(q,x)| \leq 1$, and there does not exist $x_i \in X_i,\ x_j \in X_j,\ i \neq j$, such that $|\delta(q,x_i)| > 0$ and $\left|\delta(q,x_j)\right| > 0$ simultaneously. Otherwise, it is called nondeterministic (NMFA-MSM).

Metric Spaces for Free Partially Commutative Semigroups. Below, we will consider extensions of some known metrics on regular expressions for regular languages over free partially commutative semigroup. First we consider the metric proposed by Bodnarchuk [6]. For a word p its length is denoted by $d(p)$. Let E_1 and E_2 be regular events in the classical algebra of events. The Bodnarchuk distance between E_1 and E_2 is defined as the number $\rho(E_1,E_2) = 2^{-d(E_1 \Delta E_2)}$, where $d(E) = \min\{d(r) | r \in E\}$ and $E_1 \Delta E_2$ is the symmetric set difference of the regular events E_1 and E_2.

The Bodnarchuk metric is a complete metric for regular languages over free partially commutative semigroup, but it does not tell much about the distance between two regular events and it is not a good suit for measuring difference between them.

Next, we consider the edit distance. In [9] it was extended to the distance of two regular languages, and similarly can be extended over a free partially commutative semigroup. To be more specific, Levenshtein distance will be used [10].

Let $Y = \{y_1,y_2,\ldots,y_m\}$ be a partially commutative alphabet ($Y = Y_1 \cup \ldots \cup Y_n$), and G be a free partially commutative semigroup generated from Y.

The longest sub-word $y_{m_1} y_{m_2} \ldots y_{m_k}$ of the word $g \in G$, where $y_{m_j} \in Y_i, \forall j$, is called the projection of the word g on the subset Y_i, and will be denoted as $\sigma_{Y_i}(g)$.

For $g_1,g_2 \in G$ we will denote $ED(g_1,g_2) = \sqrt{n_1^2 + n_2^2 + \ldots + n_k^2}$ as the edit distance of the words g_1 and g_2, where n_i is the edit distance of the words $\sigma_{Y_i}(g_1)$ and $\sigma_{Y_i}(g_2)$.

Let E_1 and E_2 be regular events over a free partially commutative semigroup G. $ED(E_1,E_2) = \min\limits_{r_1 \in E_1, r_2 \in E_2} (ED(r_1,r_2))$ is called the edit distance between E_1 and E_2.

The downside of the edit distance is that it disregards the actual difference of the regular events and focuses on the two closest words.

A new metric was introduced in [5] to address these drawbacks. Let G be a free partially commutative semigroup with generators $Y = y_1,y_2,\ldots,y_k$ and $g_1,g_2 \in G$.

$C_{y_i}(g_j)$ is the vector of occurrences of letter y_i in g_j [5]. As these vectors may have different lengths, denote by $C'_{y_i}(g_1)$ and $C'_{y_i}(g_2)$ the same vectors with -1-s added to the one having smaller length. Define the distance between $g_1, g_2 \in G$ as follows:

$$d(g_1,g_2) = \left\| \left(\left\| C'_{y_1}(g_1) - C'_{y_1}(g_2) \right\|_2, \ldots, \left\| C'_{y_k}(g_1) - C'_{y_k}(g_2) \right\|_2 \right) \right\|_2,$$

where $\|v\|_2$ is the L_2 norm of the vector.

Let E_1 and E_2 be regular events over a free partially commutative semigroup. d_h distance of E_1 and E_2 is defined by

$$d_H(E_1,E_2) = \max\{ \sup_{r_1 \in E_1} \inf_{r_2 \in E_2} d(r_1,r_2), \sup_{r_2 \in E_2} \inf_{r_1 \in E_1} d(r_1,r_2) \}.$$

The problem of equivalence for multitape finite automata is generally not solvable [1], hence the problem of whether $d_H(E_1,E_2) = 0$ is not solvable. This leads to the fact, that the calculation of d_H is unsolvable, as well.

In the next section an algorithm for the approximate calculation of d_H is given.

Algorithm for Approximate Distance Calculation, Implementation and Benchmark for Matching Algorithm. In this section we present the implementation of the method for approximate calculation of the difference between two regular events over a free partially commutative semigroup proposed in [5], mainly focusing on parts, which require a careful approach.

Recall the approximate method for calculating d_H. Let R_1 and R_2 be regular expressions over a partially commutative alphabet $X = X_0 \cup \ldots \cup X_n$, denoting regular events E_1 and E_2 correspondingly.

1. Construct DMFAs from R_1 and R_2:
   - (a) construct NMFA-εs from R_1 and R_2 using "Thompson's construction" [11];
   - (b) transform these NMFA-εs into NMFA-MSMs (A_1 and A_2) using "Subset construction" [11];
   - (c) transform A_1 and A_2 into DMFAs using a new tape $X_\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\}$. Denote these DMFAs by $A_1^{(D)}$ and $A_2^{(D)}$, and their starting states by q_{10} and q_{20} respectively.
2. Run "Congruence Builder" algorithm [4] passing the union of $A_1^{(D)}$ and $A_2^{(D)}$ as an input automaton.
   - (a) If q_{10} and q_{20} are in the same equivalence class, then $d_H = 0$.
   - (b) Otherwise, denote by $P_1^{(D)}$ and $P_2^{(D)}$ the finite sets of words accepted by $A_1^{(D)}$ and $A_2^{(D)}$, obtained during the execution of "Congruence Builder". Continue to the next step.
3. Delete the last words from each tuple in $P_1^{(D)}$ and $P_2^{(D)}$, getting sets of n-tuples of words P_1 and P_2 correspondingly.
4. For every $p \in P_1 \setminus P_2$, if p is accepted by A_2, add the tuple p to P_2.
5. For every $p \in P_2 \setminus P_1$, if p is accepted by A_1, add the tuple p to P_1.
6. Calculate the distance $d_H(P_1,P_2)$.

Items 4 and 5 need a more detailed approach. In particular, the implementation of these steps requires an algorithm for matching a word in a given NMFA-MSM.

The implementation of word matching is given in Algorithm 1. The procedure $MATCH\_WORD$ takes as an argument an NMFA-MSM, a state from which it starts the matching (initially q_0) and an $n-tuple$ word.

Algorithm 1 Word matching in NMFA-MSM

```
1:  procedure MATCH_WORD(A, q, t_w)
2:      A = (Q, X, δ, q_0, F)
3:      if ∀t_w^i ∈ t_w t_w is empty then
4:          if q ∈ F then
5:              return True
6:          else
7:              return False
8:      tapes ← {i | δ(q,x) ≠ ∅, x ∈ X_i}
9:      found ← False
10:     for i in tapes do
11:         if t_w^i not empty then
12:             t_w^i = w_{i_1} w_{i_2} ... w_{i_r}
13:             if δ(q, w_{i_1}) ≠ ∅ then
14:                 found ← MATCH_WORD(A, δ(q, w_{i_1}),
15:                                    (t_w^1, ..., t_w^i, w_{i_2} ... w_{i_r}, ... t_w^n))
16:             if found = True then
17:                 return True
18:     return False
```

Note, that the sequence of steps 1 (a), 1 (b) and Algorithm 1 is an algorithm for matching a given word in a partially commutative alphabet with a pattern given as a regular expression over a free partially commutative semigroup. This leads to a separate algorithm for a string matching in a multitape case.

Let R be a regular expression over a free partially commutative semigroup and, r be the total number of operators and operands in R. The worst case time of constructing an NMFA-MSM from R is $O(r^2 2^r)$, however typically the number of states s in the constructed NMFA-MSM is about r, so it takes $O(r^3)$ time (similar to the construction of DFA [11]).

The worst case complexity of checking whether an NMFA-MSM accepts an input string with a length x, by Algorithm 1 is $O(n^x)$, where n is the number of tapes. To see, whether the typical results run faster, a benchmark has been created based on http://openresty.org/misc/re/bench/. The later one is a benchmark for regular expressions for one tape automata.

An implementation of the algorithm has been written in C++. It finds all maximal words accepted by an NMFA-MSM from a given string or stream of characters. The algorithm has been tested on three different files having sizes of 25 Mb, 10 Mb and 10 Kb. For every test the number of matches and algorithm elapsed time are recorded. The results of benchmark can be viewed in https://github.com/HayMurad/multitape_regex/tree/master/test.

Items 1 (c), 3 and 6 are straightforward and do not require further details, and item 2 is already given in [4].

Algorithm 2 Calculate Approximate d_H

```
1:  procedure CALCULATE_APPROXIMATE_DISTANCE(R_1, R_2, X)
2:      A_1^(D) ← BUILD_DMFA(R_1)
3:      A_2^(D) ← BUILD_DMFA(R_2)
4:      Eq ← CONGRUENCE_BUILDER(A_1^(D) ∪ A_2^(D))
5:      if EQ_SET(Eq, q_10) = EQ_SET(Eq, q_20) then
6:          return 0
7:      else
8:          P_1 ← EQ_SET(Eq, q_10).Words
9:          P_2 ← EQ_SET(Eq, q_20).Words
10:         TRIM(P_1)
11:         TRIM(P_2)
12:         for p in P_1 \ P_2 do
13:             if MATCH_WORD(A_2, q_20, p) then
14:                 P_2 ← P_2 ∪ {p}
15:         for p in P_2 \ P_1 do
16:             if MATCH_WORD(A_1, q_10, p) then
17:                 P_1 ← P_1 ∪ {p}
18:         return d_H(P_1, P_2)
```

We proceed to full pseudocode of the algorithm (Algorithm 2) for finding the approximate distance between regular expressions over a free partially commutative semigroup. Let R_1 and R_2 be regular expressions over a partially commutative alphabet.

Here in line 2 we construct DMFA $A_1^{(D)}$ for R_1 according to steps 1 (a), 1 (b), 1 (c). Similarly we construct DMFA $A_2^{(D)}$ for R_2 in line 3. In line 4, according to step 2, we run "Congruence Builder" algorithm for the union of $A_1^{(D)}$ and $A_2^{(D)}$ and store the resulting equivalence classes in Eq. We check if q_{10} and q_{20} states are in the same equivalence class in line 5, and if they are we return 0 as the distance of regular expressions because they are equivalent. Otherwise in line 8 from Eq. equivalence classes we retrieve the class that q_{10} belongs to and store the words that were obtained during the execution of "Congruence Builder" for that class in P_1. Similarly we store words of equivalence class that q_{20} belongs in P_2 in line 9. The $n+1$-tuples of words are trimmed (lines 10, 11) by removing the last words corresponding to X_ε tape. This results in n-tuples of words. Afterwards, we loop over all the words in $P_1 \setminus P_2$ and check if given word is accepted by A_2 automaton (line 13). If there is a match we add given word into P_2 (line 14). We do similar check for all the words in $P_2 \setminus P_1$ in a loop and add them to P_1, if they are accepted by A_1 in (line 15). Finally, we calculate the d_H distance between P_1 and P_2 in line 18.

The implementation of the Algorithm 2 in C++ is given in https://github.com/HayMurad/multitape_regex/tree/master/src.

Conclusion. In this paper the distance between regular expressions over a free partially commutative semigroup is considered.

An algorithm is described for the calculation of the approximate distance between two regular expressions, and as a part of it, an algorithm is proposed for finding the matches of a given regular expression over a partially commutative alphabet in the input text. The implementation of the algorithm is done in C++ and the performance of matching algorithm is checked on a defined benchmark.

Received 07.04.2021
Reviewed 20.04.2021
Accepted 28.04.2021

REFERENCES

1. Rabin M.O., Scott D.S. Finite Automata and Their Decision Problems. *IBM J. Res. Dev.* **3** : 2 (1959),114–125. https://doi.org/10.1147/rd.32.0114
2. Letichevsky A.A, Shoukourian A.S., Shoukourian S.K. *The Equivalence Problem of Deterministic Multitape Finite Automata: A New Proof of Solvability Using a Multidimensional Tape* (eds. A.-H. Dediu, H. Fernau, C. Martín-Vide). " Language and Automata Theory and Applications" 4th International Conference, LATA 2010. Trier, Germany, May 24–28 (2010). *Proceedings of Lecture Notes in Computer Science* **6031** (2010), 392–402. Springer (2010). https://doi.org/10.1007/978-3-642-13089-2_33
3. Grigoryan H.A., Shoukourian S.K. The Equivalence Problem of Multidimensional Multitape Automata. *J. Comput. Syst. Sci.* **74** : 7 (2008), 1131–1138. https://doi.org/10.1016/j.jcss.2008.02.006
4. Grigoryan H.A., Shoukourian S.K. Polynomial Algorithm for Equivalence Problem of Deterministic Multitape Finite Automata. *Theor. Comput. Sci.* **833** (2020), 120–132. https://doi.org/10.1016/j.tcs.2020.05.044
5. Godlevsky A.B., Grigoryan H.A., Grigoryan T.A., Shoukourian S.K. Some Results on Regular Events for Multitape Finite Automata: A Preliminary Report. *Bull. EATCS* **133** (2021). http://bulletin.eatcs.org/index.php/beatcs/article/view/645
6. Bodnarchuk V.G. The Metrical Space of Events. Part I. *Cybernetics* **1** : 1 (1965), 20–24. https://doi.org/10.1007/BF01071439
7. Grigoryan T.A. An Approximate Method for Calculating the Distance Between Regular Languages for Multitape Finite Automata. *Mathematical Problems of Computer Science* **54** (2020), 69–79. https://doi.org/10.51408/1963-0060
8. Tamm H. *On Minimality and Size Reduction of One-Tape and Multitape Finite Automata.* Ph.D. Thesis. Department of Computer Science, University of Helsinki, Finland (2004).

9. Bunke H. *Edit Distance of Regular Languages. In 5th Annual Symposium on Document.* Analysis and Information Retrieval, Las Vegas, Nevada, April 15–17 (1996). Proceedings (1996), 113–124.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.468.8776
10. Levenshtein V.I. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Sov. Phys. Dokl.* **163** : 4 (1965), 845–848.
11. Aho A.V., Sethi R., Ullman J.D. *Compilers: Principles, Techniques, and Tools.* World Student Series Edition. Ser. Addison-Wesley Series in Computer Science (1986).

Տ. Ա. ԳՐԻԳՈՐՅԱՆ, Մ. Ս. ՀԱՅՐԱՊԵՏՅԱՆ

ԿԱՆՈՆԱՎՈՐ ՊԱՏԱՀՈՒՅԹՆԵՐԻ ՄԻՋԵՎ ՀԵՌԱՎՈՐՈՒԹՅԱՆ ՉԱՓՈՒՄ ԲԱԶՄԱԺԱՊԱՎԵՆ ԱՎՏՈՄԱՏՆԵՐԻ ՀԱՄԱՐ՝ ՀԱՄԱՐԺԵՔՈՒԹՅԱՆ ԴԱՍԵՐԻ ՆՈՐ ԲՆՈՒԹԱԳՐԻ ՀԻՄԱՆ ՎՐԱ

Հոդվածում քննարկվում և լուծվում են կանոնավոր պատահույթների միջև հեռավորությունը մոտավոր հաշվարկման մեթոդի իրականացման հետ կապված որոշ խնդիրներ՝ բազմաժապավեն ավտոմատների համար: Առաջարկվում է քննարկված կանոնավոր արտահայտությունների բառի համապատասխանության ալգորիթմ և բերվում են որոշ հատուկ կանոնավոր արտահայտությունների վրա ալգորիթմի կիրառման արդյունքներ: Առաջարկվող մեթոդը կարող է օգտագործվել ոչ միայն նշված իրականացման համար, այլ նաև առանձին:

Т. А. ГРИГОРЯН, М. С. АЙРАПЕТЯН

ИЗМЕРЕНИЕ РАССТОЯНИЯ МЕЖДУ РЕГУЛЯРНЫМИ СОБЫТИЯМИ ДЛЯ МНОГОЛЕНТОЧНЫХ АВТОМАТОВ НА ОСНОВЕ НОВОЙ ХАРАКТЕРИСТИКИ КЛАССОВ ЭКВИВАЛЕНТНОСТИ

В статье рассмотрены и решены несколько проблем, связанных с реализацией метода приближенного вычисления расстояния между регулярными событиями для многоленточных конечных автоматов. Предлагается алгоритм подбора слов рассматриваемых регулярных выражений и приведены результаты применения алгоритма к некоторым конкретным регулярным выражениям. Предложенный метод можно использовать не только для указанной реализации, но и отдельно.