Electronics, Communications and Networks
A.J. Tallón-Ballesteros et al. (Eds.)

doi:10.3233/FAIA231175

Analytics for AI Related Applications of Multidimensional Multitape Finite Automata

Samvel SHOUKOURIAN[a,1], Tigran GRIGORYAN[a] and Sevak AMIRKHANYAN[a]
[a] *IT Educational and Research Center, Yerevan State University, Armenia*

Abstract. A special binary representation/coding of an element in a free partially commutative monoid initially introduced in 1980 has been successfully used as a basis for the effective/polynomial solution of the following long standing open problems: functional equivalence of program schemata with non-degenerate operators, equivalence of deterministic multitape finite automata (MFA), equivalence of deterministic multidimensional multitape finite automata (MMFA), regular expressions for MFA, systems of equations for MFA regular sets. In addition, the consideration of the coding leads to an alternative characterization of commutation classes in free partially commutative monoids, which, in comparison with the already known characterization implying from the projection lemma, brings to a better efficiency when checking the equality of traces with lengths longer than a certain number or with alphabets containing more than two symbols. Regular expressions for languages of MFA and MMFA are also defined based on the mentioned coding. A brief overview of relevant AI applications concludes considerations.

Keywords. Multidimensional multitape finite automata, Regular languages, Regular expressions

1. Introduction

In 1959 M. O. Rabin and D. S. Scott introduced deterministic multitape finite automata (DMFA) and the problem of equivalence for these automata [1].

The proof of solvability for the equivalence problem of deterministic two-tape finite automata in 1973 performed by M. Bird [2] was the first step in the direction. In 1991 T. Harju and J. Karhumäki proved the solvability of the equivalence problem for DMFA without any restriction on the number of tapes [3] via a purely algebraic technique. The suggested in [4, 5] algorithmic solution of DMFA equivalence problem is like the solution suggested by M. Bird, but instead of a source automata transformation to a commutative diagram the algorithm interprets a special multidimensional tape that codes execution traces [4, 5, 6] and reflects commutativity assumptions. A special binary representation of an element in a free partially commutative monoid [4, 5, 6] is a basis of the proof.

Two main new results are under consideration.

[1] Corresponding Author, Samvel Shoukourian, IT Educational and Research Center, Yerevan State University, 1 Alek Manukyan St, 0025 Yerevan, Armenia; E-mail: samshouk@sci.am.

First, an alternative characterization for commutation classes of free partially commutative monoids, also called Cartier-Foata commutation monoids [7], or trace monoids after Mazurkiewicz [8] and his followers in computer science [9, 10, 11]. It is based on the above-mentioned special binary representation.

For the problem of trace equality comparison of complexities is performed for two algorithms solving the problem: an algorithm based on the alternative characterization and an algorithm based on the characterization implying from the projection lemma [11].

The second result is based on the solution of the equivalence problem of multidimensional multitape automata [12]. Definition of a regular expression for a multidimensional multitape finite automaton via a set of newly introduced macros that represent a movement direction of the tape heads for a given multidimensional multitape automata brings to a solution of the following problems. The synthesis problem, aka, the construction of a multidimensional multitape automaton accepting the language described by a given regular expression, and the analysis problem, aka, the construction of a regular expression representing the same language as a given multidimensional multitape automata, are solvable for the defined regular expressions.

A brief overview of relevant AI applications for the built analytic representation which is also new concludes considerations.

2. Preliminaries

Recall some definitions from [4, 5, 12, 13].

Let Y be an alphabet. Denote the set of the empty word all words in the alphabet Y by Y^*, and the set of all n-element tuples of words by $(Y^*)^n$.

Let G be a monoid, generated by the set of generators $Y = \{y_1, y_2, \dots, y_n\}$. G is called free partially commutative monoid, if it is defined by a finite set of relations R of type $y_i y_j = y_j y_i$ [14]. In the future we will use the notation $G = \langle Y \mid R \rangle$.

Denote by $I_Y \subseteq \mathrm{Y} \times \mathrm{Y}$ the set of pairs of commutative letters and by $D_Y \subseteq \mathrm{Y} \times \mathrm{Y}$ the set of pairs of non-commutative letters.

Let $K: Y^* \to (\{0,1\}^*)^n, n = |Y|$ be a homomorphism over the set Y^*, which maps words from Y^* to n-element tuples of binary words. The homomorphism K over the set of generators Y is defined by the following equation:

$$K(y_i) = (a_{1i}, \dots, a_{ni}), \text{ where } a_{ij} = \begin{cases} 1, & i = j \\ e, & (y_i, y_j) \in \mathrm{I_Y} \\ 0, & (y_i, y_j) \in \mathrm{D_Y} \end{cases}$$

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

$K(y_i y_j)$ can be defined as a left or as a right concatenation [6]. The left concatenation leads to the representation of the elements of a free partially commutative monoid by points in n-dimensional Euclidean space, i.e., by integer vectors which is presented below in Lemma 3. This representation was used essentially during consideration of the mentioned above algorithmic solution for equivalence problem of deterministic multitape finite automata. Due to that, the left concatenation is used further:

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

***Lemma 1.* [5]** *Let y_i, y_j be generators of G. Then $y_i y_j = y_j y_i \Leftrightarrow K(y_i y_j) = K(y_j y_i)$.*

From Lemma 1, the homomorphism K can be considered as a mapping over the free partially commutative monoid G.

An equivalence relation ρ over Y* is defined in the following way: words $w_1, w_2 \in Y^*$, then $w_1 \rho w_2$ if and only if w_1 and w_2 are representations of the same element in G. The disjoint classes obtained from the partitioning of Y* by the relation ρ are called, classes of commutation.

***Lemma 2.* [5]** *Any free partially commutative monoid with n generators is isomorphic to some sub-monoid of Cartesian product of n free monoids with two generators.*

Lemma 2 allows us to consider the binary codes of elements of the monoid G, instead of the elements themselves. Thus, for any element $g \in G$, its binary code can be considered, instead of the commutation class of representations of element g. The binary code is an n-element tuple of binary words, which is called BC (binary coded) canonical form and is denoted by c_g.

We denote the i-th element of the tuple of binary words $K(g)$ by $K(g)[i]$.

Let $G = \langle Y \mid R \rangle$ be a free partially commutative monoid generated by the set $Y = \{y_1, y_2, \dots, y_n\}$. As K is a homomorphism, $K(G)$ itself is a monoid with the set of generators $K(Y)$ and with the identity $\Big(\underbrace{e, \dots, e}_{n \text{ times}}\Big)$. It follows from Lemma 2 that any element in $K(G)$ is representable in the form $k_{i_1} k_{i_2} \dots k_{i_m}$ where $k_{i_j} \in K(Y)\ \forall j = 1, \dots, m$, and this representation is not unique. The equivalence relation $\rho_{K(Y)}$ partitions $K(Y)^*$, i.e., the set of all words in $K(Y)$, into commutation classes. Denote the class containing the element p by $[p]$. Obviously, $[(e, \dots, e)] = \{(e, \dots, e)\}$ and $[k_i] = \{k_i\}$, $\forall k_i \in K(Y)$. The classes of commutation are used to define regular sets and regular expressions on a free partially commutative monoid [13] in a similar manner as the regular sets and regular expressions for one-tape automata are defined [15].

To conclude: regular expressions can be used for description of some patterns in n-dimensional space.

3. An Alternative Characterization of Traces

In trace theory, the commutation classes gotten from the partitioning of Y^* by the relation ρ are called traces [11]. The quotient monoid is a free partially commutative monoid induced by the relation I_Y, it is called trace monoid and denoted by $M(Y, I_Y)$. The elements of a trace monoid are traces. Recall the projection lemma for trace monoids.

Lemma 4 (Projection Lemma). [11] *For traces u and v of $M(Y, I_Y)$, $u = v$ if and only if $\pi_{y,y'}(u) = \pi_{y,y'}(v)$ for all $(y, y') \in D_Y$, where π_A is a canonical homomorphism erasing from a trace all the letters not belonging to A.*

For a letter y and word w, define $Ocr_y(w)$ as the occurrence number of y in w.

For $Z \subset Y$, define $Ocr_Z(w) := \sum_{y \in Z} Ocr_y(w)$.

Define $\sigma_{y,i,j}(w), i < j$ as the sub-word of w such that the sub-word starts with the i-th occurrence of y in w and ends with j-th occurrence of y in w. If $i = 0$, then $\sigma_{y,i,j}(w)$ is a prefix of w, and if $j > Ocr_y(w)$, then $\sigma_{y,i,j}(w)$ is a suffix of w.

Let w be a word on a partially commutative alphabet Y and $y \in$ Y. Define the vector of numbers $\delta_y(w) = (\alpha_0, \alpha_1, \dots, \alpha_k)$ such that $\alpha_i = Ocr_{\{y' \in Y | (y,y') \in D_Y\}}\left(\sigma_{y,i,i+1}(w)\right)$, where $i = 0, \dots, k$; $k = Ocr_y(w)$.

For the letters $y, y' \in Y$ such that $(y, y') \in D_Y$ define the vector of numbers $\delta_{y,y'}(w) = (\beta_0, \beta_1, \dots, \beta_k)$ such that $\beta_i = Ocr_{y'}\left(\sigma_{y,i,i+1}(w)\right)$, where $k = Ocr_y(w)$ and $i = 0, \dots, k$.

It is easy to see that the knowledge of all the vectors $\delta_{y,y'}(w)$ characterizes the commutation class of the word w. Indeed, one can easily obtain the set of projections $\{\pi_{y,y'} | y, y' \in Y\}$ from the set of functions $\{\delta_{y,y'} | y, y' \in Y\}$ and vice versa. Note, that the sets $\{\pi_{y,y'} | y, y' \in Y\}$ and $\{\delta_{y,y'} | y, y' \in Y\}$ are equinumerous.

The result of Lemma 4 allows us to formulate the following theorem, which states that the knowledge of the vectors $\delta_y(w)$ is sufficient to characterize the commutation class of the word w.

Theorem 1. *Let g_1 and g_2 be elements of partially commutative* monoid *G with generators $\{y_1, y_2, \dots, y_n\}$. $g_1 = g_2$ if and only if*

$\delta_{y_i}(g_1) = \delta_{y_i}(g_2)$ for all $i = 1, \dots, n$.

Proof. Let $\delta_{y_i}(g) = (\alpha_0, \alpha_1, \dots, \alpha_k)$. Consider the i-th element of the tuple of binary words $K(g)$, i.e., $K(g)[i]$. From the definition of the left concatenation operation, it is obvious that $\delta_1(K(g)[i]) = (\alpha_k, \dots, \alpha_1, \alpha_0)$. In other words, k is the number of occurences of letter 1 in the binary word $K(g)[i]$, and $\alpha_0, \alpha_1, \dots, \alpha_k$ are the numbers of occurences of zeros between each adjacent occurrences of ones in the binary word $K(g)[i]$ from right to left. The components of $\delta_1(K(g)[i])$ amount to $0^{\alpha_k}10^{\alpha_{k-1}}1 \dots 0^{\alpha_1}10^{\alpha_0}$, hence, it follows from Lemma 2 that $g_1 = g_2$ if and only if $\delta_{y_i}(g_1) = \delta_{y_i}(g_2)$ $\forall i = 1, \dots, n$.

We consider the complexity of checking whether two given traces are equal. Assume that traces are given by any of their representations. The problem transposes to checking whether two given words belong to the same trace, hence we consider the words of Y^* as inputs of the algorithm. For Y and I_Y fixed and a word w, both the projections $\pi_{y,y'}$ [11] and the vectors δ_y are computable in linear time. For two words u and v, given their characterization projections $\pi_{y,y'}$ and characterization vectors δ_y, finding out whether $\pi_{y,y'}(u) = \pi_{y,y'}(v)$ $\forall (y, y') \in D_Y$, as well as checking whether $\delta_y(u) = \delta_y(v)$ $\forall y \in Y$ are both computable in linear time, too. Meantime, it is worthwhile to consider the dependence of the complexity of those algorithms from the size of the alphabet and the number of pairs in D_Y.

The time complexity analysis is done below both for the worst-case and the average-case scenarios [16]. Further, in the paper, by writing complexity we will mean time complexity. By the average-case scenario analysis, the probabilistic analysis is meant [16]. We assume that input words are uniformly distributed, i.e. for a fixed non-negative integer l, all input words of length l can occur with the same probability. This assumption allows us to use the term "average" in the analysis instead of the term "expected value".

Let the cardinality of the alphabet Y be n, i.e., $|Y| = n$, and $|D_Y| = 2k$. Consider the words with a length l and let w be such a word. The average of occurrences for a given letter in the words of length l is $\frac{l}{n}$, hence, for fixed y and y' the average length of the projection $\pi_{y,y'}(w)$ is $\frac{2l}{n}$. The average-case complexity of computing the set $\{\pi_{y,y'}(w) \mid (y,y') \in D_Y\}$ is $\Theta\left(\frac{kl}{n}\right)$.

$\forall i = 1, \ldots, n$ let k_i be cardinality of the set $\{(y_i, y_j) | (y_i, y_j) \in D_Y, j = 1, \ldots, n, j \neq i\}$, in other words, k_i is the number of letters with which y_i does not commute. Obviously $\sum_{i=1}^{n} k_i = 2k$ and the average value for k_i is $\frac{2k}{n}$. Hence, the average-case complexity of computing the set $\{\delta_y \mid y \in Y\}$, is also $\Theta\left(\frac{kl}{n}\right)$, as, for each letter in w, we do $\frac{2k}{n}$ increments on average. The worst-case complexity scenario is when $w = y_i^l$ and y_i is non-commutative with all letters in Y. Then, the complexity of both cases is $\Theta(nl)$.

Next, suppose the projections $\pi_{y,y'}(u)$ and $\pi_{y,y'}(v)$ are given for the words u and v, and the length of the word u is l and it is shorter than or equal to the length of v. On average, the projections $\pi_{y,y'}(u)$ are of length $\frac{2l}{n}$, hence the complexity of comparing all the projections $\pi_{y,y'}(u)$ and $\pi_{y,y'}(v)$ has time complexity $\Theta(kl/n)$. The worst-case scenario is when $k = n(n-1)/2$, then the complexity is $\Theta(nl)$.

On the other hand, the sum of dimensions of vectors $\delta_y(u)$ is $(l + n)$: denote $l_i = |\delta_{y_i}(u)|, \forall i = 1, \ldots n$, then $\sum_{i=1}^{n} l_i = l + n$. Thus, the computation complexity of the equality of all vectors $\delta_y(u)$ and all vectors $\delta_y(v)$ is $\Theta(l + n)$.

The summary of this analysis is presented in Table 1.

Table 1. Characterization set's and equivalence problem's computation time complexity comparison for $\{\pi_{y,y'}(w) \mid (y,y') \in D_Y\}$ and $\{\delta_y \mid y \in Y\}$ characterizations

	Computing Characterization Set		**Computing Equivalence**	
Characterization	**Average-case Complexity**	**Worst-case Complexity**	**Average-case Complexity**	**Worst-case Complexity**
$\boldsymbol{\pi_{y,y'}}$	$\Theta\left(\frac{kl}{n}\right)$	$\Theta(nl)$	$\Theta\left(\frac{kl}{n}\right)$	$\Theta(nl)$
$\boldsymbol{\delta_y}$	$\Theta\left(\frac{kl}{n}\right)$	$\Theta(nl)$	$\Theta(l+n)$	$\Theta(l+n)$

It is natural to consider the application of the obtained result to multitape finite automata - we are implicitly tied to it with the introduced alternative coding [5, 12]. In the case of mutitape automata, for each pair of symbols from one tape, these symbols are not commutative, while for each pair of symbols from different tapes, these symbols are commutative. The alternative characterization becomes more advantageous along with the growing number of letters in each tape. It can be argued that if the alphabet of a tape consists of more than 2 symbols, it can be encoded via 2 symbols (if all alphabets contain 2 symbols or less, this is the worst case for the alternative characterization), but in this case the encoding will also result in additional complexity for the automaton.

Definition (Multitape Finite Automata) [1]. *A tuple $A = (Q, T, Y, \delta, q_0, F)$, where Q is a finite set of states; $T\colon Q \to \{1, \ldots, m\}$ is a tape function; Y is an input alphabet such that $Y = Y_1 \cup Y_2 \cup \ldots \cup Y_m$, $Y_i \cap_{i \neq j} Y_j = \emptyset$ and $\forall y, y(y \in Y_i, y' \in Y_j (i \neq j), yy' =$*

$y'y)$; $\delta: Q \times Y \to 2^Q$ is the transition function: $\forall q \in Q, \forall y \in Y$ $\delta(q,y)$ is defined if and only if $\exists i \in \{1,\dots,m\}$ such that $q \in Q_i$ and $y \in Y_i$; $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states, is called an n-tape automaton.

In the MFA definition the subset Y_i corresponds to the tape i $(i = 1,\dots,m)$.

Free partially commutative monoids generated by the alphabets and commutativity relations of multitape finite automata can be considered as special cases of trace monoids, where Y is divided into disjoint subsets Y_i each corresponding to a specific tape. In this case $I_Y = \{(y,y') | y \in Y_i, y' \in Y_j, i \neq j\}$ and $D_Y = \{(y,y') | \exists i \; s.t.\, y, y' \in Y_i\}$. It is easy to see that the cardinality of D_Y is equal to $|D_Y| = \sum_{i=1}^{m}(|Y_i| \cdot (|Y_i| - 1))$. This yields to the following corollary:

Corollary. The case when all the tape alphabets consist of two letters $(|Y_i| = 2 \; \forall i = 1,\dots,m)$ is a boundary case: that is, if we add one or more letters to any number of subsets Y_i of the alphabet, it will be faster to solve the equality problem for words having length $l > cn^2/(2k - n)$ with the characterization vectors introduced in Theorem 1.

4. Relation between Regular Expressions and MMFA

In this section we introduce regular expressions for multidimensional multitape automata and solve the synthesis and analysis problems.

Recall some definitions from [12].

The set N^r, where r is a positive integer and $N = \{0,1,\dots\}$, is called an r-dimensional tape. An element $(a_1,\dots,a_r) \in N^r$ is called a cell of the with coordinates $a_1,\dots,a_r$.

The signature of MMA is defined as the set $S = \{(n_1,m_1),\dots,(n_k,m_k)\}$, where n_i, m_i $(1 \le i \le k)$ are natural numbers and for all $1 \le \; i,j \le \; k, n_i = n_j \; \Leftrightarrow \; i = j$. $\forall i$ it defines the number of tapes (m_i) with the arity n_i.

Further, in the paper, we assume that $= \{(i,m_i) \mid 1 \le i \le c\}$, where $c \ge 2, m_i \ge 0, m = m_1 + \dots + m_c > 0$.

Definition (Multidimensional Multitape Automaton) [14]. *A tuple* $A = \langle Q,T,X,q_0,F,\phi,\psi \rangle$, where Q is the set of states; $T: Q \to \{1,\dots,m\}$ is a tape function; X is the input alphabet; $q_0 \in Q$ *is the initial state;* $F \subseteq Q$ *is the set of final states;* $\phi : Q \times X \to Q$ is the transition function; $\psi : Q \times X \to \{1,\dots,c\}$ is the movement direction function, is called a c-dimensional multitape automaton with signature S.

We write $A(w) = 1$ if an automaton A accepts the word $w \in X^*$, and $A(w) = 0$, if it does not.

Two MMFAs A_1 and A_2 with the same signature are called equivalent $(A_1 \sim A_2)$, if and only if, $\forall w \in X^* \; A_1(w) \; = \; A_2(w)$ and the coordinates of the heads on all tapes are equal.

In [12], an algorithm for constructing a multitape finite automaton from a given multidimensional multitape automaton is brought. The main step of the algorithm is the transformation of the c-dimensional automaton A with the signature S to a $(c-1)$-dimensional automaton A' with $(m + m_c)$ tapes and the signature S', where:

$S' = S \setminus \{(c, m_c), (c-1, m_{c-1}), (1, m_1)\} \cup S_1,$
$S_1 = \{(c-1, m_{c-1} + m_c), (1, m_1 + m_c)\}$, if c > 2,
$S_1 = \{(1, m_1 + 2m_2)\}$, if $c = 2$.

The transformation from A to A' for the transitions $\psi(q,x) = 1$ and $\psi(q,x) > 1$, are shown in Figure 3 in [12] (Figure 1 in this paper).

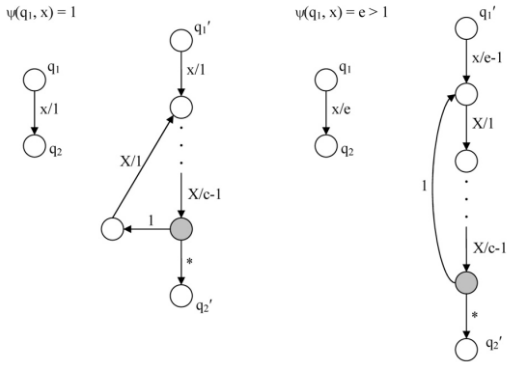

Figure 1. [12] Transformation of a transition in n-dimensional multitape finite automaton to a transition in $(n-1)$-dimensional multitape finite automata.

These transformations are used to keep the movements of heads of multidimensional tapes. For the definition of regular expressions for multidimensional multitape automata, we also need to express the movement of heads of multidimensional tapes.

For that, we introduce a group of symbols: $\left\{\overset{k}{\rightarrow}\right\}$, where $k \in \mathbb{N}$.

Now, we can extend the definition of regular expressions by adding the introduced macros to the alphabet letters. Let $G = \langle Y \mid R \rangle$ be a free partially commutative monoid, generated by the set of generators $Y = \{y_1, y_2, \dots, y_n\}$ and the set of relations R of type $y_i y_j = y_j y_i$. Suppose that $Y = Y_1 \cup \dots \cup Y_m$, such that $Y_i \bigcap_{i \neq j} Y_j = \emptyset$ and $\forall y, y' \left(y \in Y_i, y' \in Y_j (i \neq j), yy' = y'y\right)$. Each subset Y_i corresponds to i-th tape. Consider the following extension of the free partially commutative semigroup G:

1. $Y_i' = \left\{y \overset{k}{\rightarrow} \,\middle|\, y \in Y_i;\ k \in \mathbb{N},\ 0 < k \leq c_i; where\ c_i\ is\ the\ arity\ of\ i-th\ tape\right\}$
2. $Y' = Y_1' \cup \dots \cup Y_m'$,
3. $R' = \left\{y \overset{k}{\rightarrow} y' \overset{k'}{\rightarrow} = y' \overset{k'}{\rightarrow} y \overset{k}{\rightarrow} \,\middle|\, y \in Y_i, y' \in Y_j, i \neq j;\ 0 < k \leq c_i, 0 < k' \leq c_j\right\}$
4. $G' = \langle Y' \mid R' \rangle$

The notation $y \overset{k}{\rightarrow}$, where $y \in Y_i$, means, that after reading the symbol y, the head of the i-th tape moves one cell in k-th direction.

Next, we consider the problem of constructing multidimensional multitape finite automaton from a given regular expression and vice-versa. The two problems are referred by Glushkov as the synthesis and the analysis problems of automata, correspondingly [15].

The problem of constructing MFA from the given regular expression is solved in [17]. We use a similar approach for MMFA.

When considering these problems for MMFAs, we always assume that the free partially commutative monoids on which regular expressions are defined, are generated from alphabets for multitape automata complemented with $\overset{k}{\rightarrow}$ symbols. Otherwise, the alphabets will not be applicable for multidimensional multitape finite automata.

Theorem 2. *There is an algorithm, which constructs an ε-MMFA A for a given regular expression R on a free partially commutative monoid, such that* $L(A) = L(R)$.

Proof. We use Thompson's construction [18] with one key difference: any transition of type $\left(q_1, x \overset{k}{\rightarrow}, q_2\right)$ is transformed to a transition $(q_1, x/k, q_2)$.

Theorem 3. *There is an algorithm, which finds a regular expression R on a free partially commutative monoid for a given ε-MMFA A, such that* $L(A) = L(R)$.

Proof. One such algorithm is the technique of state elimination [19], with a single difference: all transitions $(q_1, x/k, q_2)$, first, need to be transformed to the transitions $\left(q_1, x \overset{k}{\rightarrow}, q_2\right)$.

5. System of Equations on Regular Expressions

Next, we consider systems of equations on regular expressions for multidimensional multitape finite automata.

Let S_{ij} and P_i be given regular expressions ($\forall i, j = 1, \dots, n$) and X_i be unknown regular expressions ($\forall i = 1, \dots, n$). Consider the following system of equations:

$$\begin{cases} X_1 = X_1 S_{11} + X_2 S_{21} + \cdots + X_n S_{n1} + P_1 \\ X_2 = X_1 S_{12} + X_2 S_{22} + \cdots + X_n S_{n2} + P_2 \\ \vdots \\ X_n = X_1 S_{1n} + X_2 S_{2n} + \cdots + X_n S_{nn} + P_n \end{cases} \tag{1}$$

Systems of equations of regular expressions for one-tape finite automata and its solutions were considered in [20] by V. Bodnarchuk. Similarly, in [13] the solution of systems of equations of regular expressions for multitape finite automata was given.

In both cases it is shown that the solution of the equation $X = XS + P$ is $X = P(S)^*$, and, that the minimal solution of the system of equations (1) can be found by a successive elimination of unknown variables.

For the case of multidimensional multitape finite automata the same approach does not work in the general case, as regular expressions also express the coordinates of multidimensional tapes. We consider only regular expressions (R) obtained from deterministic multidimensional multitape finite automata (A) using the method suggested in the proof of Theorem 3. Note, that A and R accept the same language with the same coordinates for each word of that language $\left(L(R) = L(A)\right)$. For a given regular expression R satisfying the aforementioned condition, the automaton A can be constructed by making the opposite steps suggested in the proof of Theorem 3.

In order to find the solution of the system of equations (1) for the case of multidimensional multitape finite automata, the transformation represented in Figure 1 is used. Define a mapping of $\phi: \mathcal{R}_m \rightarrow \mathcal{R}$, where $\mathcal{R}_m$ is the set of all multidimensional regular expressions obtained from deterministic multidimensional multitape finite automata using the proof of Theorem 3 and $\mathcal{R}$ is the set of all regular expressions on a free partially commutative monoid (regular expressions for one-dimensional multitape finite automata):

1. $\phi(e) = e$,
2. $\phi\left(y \overset{1}{\rightarrow}\right) = y \overset{1}{\rightarrow} \left(Y \overset{2}{\rightarrow} \ldots Y \overset{c-1}{\longrightarrow}\right)\left(* \overset{1}{\rightarrow} + 1 \overset{1}{\rightarrow} Y \overset{1}{\rightarrow} \left(Y \overset{2}{\rightarrow} \ldots Y \overset{c-1}{\longrightarrow}\right)\right)$, where c is the dimension of the tape y belongs to, and Y in the regular expression means any symbol from the alphabet Y,
3. $\phi\left(y \overset{k}{\rightarrow}\right) = y \overset{k-1}{\longrightarrow} \left(Y \overset{1}{\rightarrow} Y \overset{2}{\rightarrow} \ldots Y \overset{c-1}{\longrightarrow}\right)\left(* \overset{1}{\rightarrow} + 1 \overset{1}{\rightarrow} \left(Y \overset{1}{\rightarrow} Y \overset{2}{\rightarrow} \ldots Y \overset{c-1}{\longrightarrow}\right)\right)$, where $1 < k \leq c$,
4. $\phi(r_1 r_2) = \delta(r_1)\delta(r_2)$,
5. $\phi(r_1 + r_2) = \phi(r_1) + \phi(r_2)$,
6. $\phi(r^*) = \left(\phi(r)\right)^*$.

We apply the transformation ϕ on the regular expressions S_{ij} and P_h as many times, as it is necessary to transform them to one-dimensional regular expressions $\left(S'_{ij}, P'_h\right)$, i.e. for all symbols $\overset{i}{\rightarrow}$ in regular expressions $i = 1$. Obviously, these regular expressions are regular expressions for multitape finite automata, so we can omit $\overset{1}{\rightarrow}$ symbols in their final notation. After replacing all S_{ij} and P_h regular expressions with their corresponding S'_{ij} and P'_h regular expressions, we can solve the new system of equations by a successive elimination of unknown variables [13].

It is important to note, that the solution of the system of equations obtained in this way is a regular expression for multitape finite automata that may contain additional tapes compared to automata corresponding to coefficients of the considered system of equations. Moreover, the regular language corresponding to the solution might not be representable by a deterministic multidimensional multitape finite automata.

6. Examples of Possible Applications

Several directions of our further investigations based on the suggested analytics are outlined below with a hope to receive feedback on hidden difficulties when exploring these directions as well as to come to new proposals relating to other possible directions of application not listed currently.

6.1. Application to Learning

Learning involves effective compression of data via leveraging regularities. Based on the suggested apparatus the following approach can be used among others.

As a practical matter, simpler computational devices like MFA or MMFA can be embedded into trainable systems like neural nets. Assume that each embedded device represents some analytics reflecting an approximate compression for a part of the expected data. Such a device can serve as a "sensor" for a neural net with a role to fix a given distance deviations from a permissible range. If the number of such violations exceeds the arranged beforehand limit, then the embedded device should be extended/tuned to immerse these violations into the current analytics. The data can come from different sources (multiple tapes) and can be of different types (multiple dimensions). This process should be continued either till saturation, when no violations exceeding the distance or the limit are observed, or till an infinite set of non-immersive violations is observed.

The first case means that no more devices should be embedded in the neural net for observing regularities.

The latter case means that power-wise the MFA/MMFA model is not sufficient for further considerations and a new more powerful model should be suggested for observing these regularities.

As an example, such devices can work in accordance with neural nets for forest smoke recognition [21, 22], and for smoke estimation of smoky vehicles [23].

6.2. Monitoring Behavior of Cellular and Self-Replicating Automata

Again, MFA/MMFA as less powerful devices can be embedded, where necessary, in cellular and self-replicating automata for monitoring, early diagnosing and avoiding different non-desirable continuations of automata functioning. For some published cases of such automata behavior, the considered here models are sufficient for monitoring the behavior, for others there is a need to have reversible automata. Meantime, there are also cases where MMFA are not sufficient for monitoring the behavior and there is a need to use more powerful automata models.

We plan very next to consider extensions of the technique suggested in [24] for reversible one tape MMFA for reducing the consideration of analytics for such automata to the case considered in the current publication.

6.3. Monitoring of Moving Geometric Objects in Multidimensional Space

A novel approach can be developed for efficient representation of approximations for moving in n-dimensional space of some geometric objects or their most distinguishable parts by means of the introduced regular expressions and their corresponding finite automata. A fundamental advantage of regular expressions over context free or other grammars is that practical questions regarding regular languages can be answered algorithmically in polynomial time while for other grammars that is not possible. The approach should be oriented to dealing only with those representation methods which are suitable for direct machine processing.

6.4. Validation of AI Generated Language Models Basing on Regular Expressions

Large language models [25, 26] (e.g., GPT-3, GPT-4), are broadly used for many natural language processing tasks. Meantime there are multiple concerns around negative effects of the models: data memorization, bias, inappropriate language, etc. A system/engine is suggested in [25] for validating these models via standard regular expressions despite the difficulties connected with their complexity and generation capacities. The suggested analytics will allow us to consider different sources for the generation as well as different directions of generation within the same source, i.e., to use a more powerful model for validation.

6.5. Regular Languages in Protein Structures Computing

We believe that there is a chance to extend the use scope for regular languages in protein structures computing (see, e.g., [27, 28]) if to apply the suggested analytics instead of standard regular expressions.

References

[1] Rabin MO, Scott DS. Finite automata and their decision problems. IBM Journal of Research and Development, 1959;3(2):114–125.

[2] Bird M. The equivalence problem for deterministic two-tape automata. Journal Computer and System Sciences, 1973;7(2):218–236.

[3] Harju T, Karhumaki J. The equivalence problem of multitape finite automata. Theoretical Computer Science, 1991; 78(2):347–355.

[4] Letichevsky A, Shoukourian A, Shoukourian S. The Equivalence Problem of Deterministic Multitape Finite Automata: A New Proof of Solvability Using a Multidimensional Tape. In: Dediu A, Fernau H, Martin-Vide C (eds.), Language and Automata Theory and Applications, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings, volume 6031 of Lecture Notes in Computer Science. Springer, 2010 pp. 392–402.

[5] Godlevskii AB, Letichevskii AA, Shukuryan SK. Reducibility of program-scheme functional equivalence on a nondegenerate basis of rank unity to the equivalence of automata with multidimensional tapes. Cybernetics, 1980;16(6):793–799.

[6] Grigoryan H., Shoukourian S. Polynomial algorithm for equivalence problem of deterministic multitape finite automata. Theoretical Computer Science, 2020. 833:120–132.

[7] Cartier P, Foata D. Problemes combinatoires de commutation et rearrangements, volume 85 of Lecture Notes in Mathematics. Springer-Verlag Berlin Heidelberg, 1969.

[8] Mazurkiewicz A. Concurrent Program Schemes and their Interpretations. DAIMI Report Series, 1977. 6(78).

[9] Diekert V. Combinatorics on Traces, volume 454 of Lecture Notes in Computer Science. Springer, 1990.

[10] Mazurkiewicz AW. Introduction to Trace Theory. In: Diekert V, Rozenberg G (eds.), The Book of Traces, pp. 3–41. World Scientific, 1995.

[11] Diekert V, Metivier Y. Partial Commutation and Traces. In: Rozenberg G, Salomaa A (eds.), Handbook of Formal Languages, Volume 3: Beyond Words, pp. 457–533. Springer, 1997.

[12] Grigorian H., Shoukourian S. The equivalence problem of multidimensional multitape automata. Journal of Computer and System Sciences. 2008 Nov 1;74(7):1131-8.

[13] Grigoryan T. Some results on regular expressions for multitape finite automata. Proceedings of the YSU A: Physical and Mathematical Sciences, 2019. 53(2):82–90.

[14] Clifford AH, Preston GB. The Algebraic Theory of Semigroups, Volume I, volume 7.1 of Mathematical Surveys and Monographs. American Mathematical Society, 2 Edition, 1961.

[15] Glushkov VM. The abstract theory of automata. Russian Math. Surveys, 1961. 16(5):3-62.

[16] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms, 3rd Edition, MIT Press 2009.

[17] Godlevsky A, Grigoryan HA, Grigoryan T, Shoukourian SK. Some Results on Regular sets for Multitape Finite Automata: A Preliminary Report. Bull. EATCS 133 (2021).

[18] Aho AV, Sethi R, Ullman JD. Compilers: Principles, Techniques, and Tools. Addison-Wesley series in computer science / World student series edition. Addison-Wesley, 1986.

[19] Brzozowski JA, McCluskey EJ. Signal ow graph techniques for sequential circuit state diagrams. IEEE Trans. Electron. Comput., 1963. 12(2):67-76.

[20] Bodnarchuk VG. Systems of equations in the algebra of events. USSR Computational Mathematics and Mathematical Physics, 1963. 3(6):1470-1487.

[21] Tao H, Duan Q, Lu M, Hu Z. Learning discriminative feature representation with pixel-level supervision for forest smoke recognition. Pattern Recognition, 2023. 143:109761.

[22] Tao H. A label-relevance multi-direction interaction network with enhanced deformable convolution for forest smoke recognition. Expert Systems with Applications, 2024. 236:121383.

[23] Tao H, Duan Q. Learning discriminative feature representation for estimating smoke density of smoky vehicle rear. IEEE Transactions on Intelligent Transportation Systems, 2022. 23(12):23136-47.

[24] Shukuryan SK. Comparison of two-sided automata with respect to operating time and decidability in the problem of functional equivalence in a certain class of discrete transducers and program schemes over memory. Cybernetics, 1977. 13:152–159.

[25] Kuchnik M, Smith V, Amvrosiadis G. Validating Large Language Models with ReLM, Carnegie Mellon University, arXiv:2211.15458v1 [cs.LG], 21 Nov 2022.
[26] Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D. Language models are few-shot learners. In Advances in Neural Information Processing Systems, 2020. 33:1877– 1901.
[27] Rozenberg G, Salomaa A. DNA Computing: New Ideas and Paradigms. ICALP 1999: pp. 106-118.
[28] Wang Z. DNA Computing: Modelling in Formal Languages and Combinatorics on Words, and Complexity Estimation, PhD thesis in Computer Science, University of Waterloo, Ontario, Canada, 2022.