A Myhill-Nerode Theorem for Finite State Matrix Automata and Finite Matrix Languages

Abhisek Midya[1(✉)] and D. G. Thomas[2]

[1] CMR Institute of Technology, Bengaluru, India
abhisekmidyacse@gmail.com
[2] Department of Mathematics, Madras Christian College, Chennai, India

Abstract. We propose a deterministic version of *finite state matrix automaton (DFSMA)* which recognizes *finite matrix languages (FML)*. Our main result is a generalization of the classical Myhill-Nerode theorem for *DFSMA*. Our generalization requires the use of two relations to capture the additional structure of *DFSMA*. *Vertical equivalence* $\equiv_v$ captures that words sharing the same vertical location, *horizontal equivalence* $\equiv_h$ captures that words sharing the same horizontal location. A finite matrix language is defined to be regular if relations $\equiv_v$ and $\equiv_h$ exist that satisfy certain conditions, in particular, they have finite index. We show that the language associated to a *DFSMA* is regular, and we construct, for each finite matrix language, a *DFSMA* that accepts this language. Our result provides a foundation for learning algorithms for *DFSMA*.

Keywords: Myhill-Nerode equivalence · Deterministic finite state matrix automata · Finite matrix languages

1 Introduction

Grammatical inference is the realistic common area of research between machine learning and formal language theory. The concept of Grammatical inference deals with the automatic learning of grammars, automata and other language describing devices. We attempt to satisfy both (machine Learning and formal Language Theory) parts of the potential readership of this paper, as it has been shown that the inter-dependencies between both areas are strong. The basic motivation for investigating the learning of *DFSMA*, is to investigate the connection between matrix languages and *automata learning*.

1.1 Learning Aspects

It has been investigated that the passive learning problem of finding a minimal *deterministic finite automata (DFA)* is NP-hard, and it is compatible with a finite set of positive and negative examples, in [16]. In spite of this, many

R. P. Barneva et al. (Eds.): IWCIA 2022, LNCS 13348, pp. 154–170, 2023.
https://doi.org/10.1007/978-3-031-23612-9_10

DFA identification algorithms have been developed. [2] presented an efficient algorithm for active learning a regular language L, which assumes a minimally adequate teacher (MAT) that answers two types of queries about L, with a membership query, the algorithm asks whether or not a given word w is in L, and with an equivalence query it asks whether or not the language L_H of an hypothesized DFA H is equal to L. If L_H and L are not same, a word which is in the symmetric difference of the two languages gets returned. Also there are alternative versions of algorithms for learning regular languages in the MAT model appeared in [7,8,17,30,40]. The limits of the model were investigated in [3,5]. There was an interesting question arose, whether it can be extended to the supersets of regular sets.

In [29] Radhakrishnan and Nagaraja proposed a method for the inference of *even linear languages* from positive examples, also the proposed method can be used in a hierarchical manner to infer grammars for complex pictures. The interesting work of [36] and [34] established the reduction technique of the learning of even linear languages (introduced in [1]) to the learning of regular languages. Also, the usefulness of the concept of control languages (originating from [15]) was shown in the reduction of the learning problem of languages through controlled fixed grammars in [20,21,36,38,39]. In particular, Takada used this concept to develop an efficient learning algorithm, called "even equal matrix languages" [37,39]. Also, in [23,24] polynomial time learning algorithms are proposed for interesting subclasses of contextual array and string languages respectively. Also, in [25], a two dimensional automaton had been defined for array languages. In this way, we realize the importance of learning matrix languages and in this paper we deal with *finite matrix languages*.

In this article, we propose a deterministic version of finite state matrix automata ($DFSMA$) which can recognize finite matrix languages (FML). More importantly, we establish a Myhill-Nerode theorem for $DFSMA$ and FML. We know that the Myhill-Nerode theorem refers to a single equivalence relation on words, and constructs a DFA in which states are equivalence classes, our generalization requires the use of two relations to capture the additional structure of $DFSMA$. The Myhill-Nerode theorem makes the platform to develope a learning algorithm for $DFSMA$ using query learning model [2].

Myhill-Nerode theorems are of pivotal importance for learning algorithms. Angluin's classical L^* algorithm for active learning of regular languages, as well as improvements such as [11,19,30], use an observation table to approximate the Myhill-Nerode congruence. Maler and Steiger [22] established a Myhill-Nerode theorem for ω-languages that serves as a basis for a learning algorithm described in [4]. The SL^* algorithm for active learning of register automata of Cassel et al. [10] is directly based on a generalization of the classical Myhill-Nerode theorem to a setting of data languages and register automata (extended finite state machines).

1.2 Formal Language Aspects

Syntactic approaches, on account of their structure-handling capability, have played an important role in the problem of description of picture patterns

considered as connected digitized, finite arrays of symbols. Pioneering work in suggesting and applying a linguistic model for the solution of nontrivial problems in picture processing was presented in [27]. Using the techniques of formal string language theory, various types of picture or array grammars have been introduced and investigated in [9,13,14,31,32]. Most of the array grammars are based on Chomskian string grammars. Some recent results on picture languages can be found in [6,12,26].

A picture can be represented as a $m \times n$ matrix in which each entry is a_{ij} where $1 \leq i \leq m, 1 \leq j \leq n$. By an operation on a digitized picture is meant a function which transforms a given picture matrix into another one. Programming languages have types and a function may have an argument, which is of type matrix, and it is not trivial to handle computationally. For practical purposes it is desirable to work with operations on digitized pictures which can be defined in terms of functions having considerably fewer arguments.

In this paper we deal with a linguistic model for the generation of matrices (rectangular arrays of terminals) by the substitution of *regular sets* [18] into well-known families of formal languages. In formal language theory the substitution operator operates on 'string languages' (languages made up of strings of terminals). Here the substitution operator operates on a 'string language' and the resultant is a 'matrix language' (language whose sentences are matrices, i.e., $m \times n$ arrays of terminals). In particular, we recall finite/regular matrix languages and we propose the corresponding deterministic version of automaton, called deterministic finite state matrix automata ($DFSMA$). Matrix grammar refers to a grammar in which the production rules are applied together in fixed sets. There are several variants where the rewriting rules are regular, context-free or context-sensitive with arrays of terminals in the place of strings of terminals. Furthermore, in order to obtain richer families, restrictions are imposed on the use of production rules in well known families of grammars. Several such studies are available in the literature [33]. In this paper, our focus is on FML where the rewriting rules are regular. Some interesting classes of pictures including certain letters of the alphabet, kolam, (traditional picture patterns used to decorate the floor in South Indian homes) and wall paper designs (repetitive patterns) can be generated by finite matrix grammars.

The remainder of this paper is organized as follows. Section 2 recalls the definition of FML and Subsect. 2.1 presents examples of FML for better understanding. Section 3 proposes the definition of $DFSMA$ and examples are discussed in Subsect. 3.1 Section 4 presents some of the important results about FML. In Sect. 5, we discuss the Myhill-Nerode equivalence and establish the Myhill-Nerode theorem for $DFSMA$ with illustrations with examples in Subsect. 5.1 and 5.2. Section 6 concludes the work and shows a future direction of work.

2 Finite Matrix Language (FML)

We recall the definition of FML [35] based on *right linear grammar* [18].

Definition 1 (Finite Matrix Language (FML)). *A Finite matrix grammar (FMG) is a pair* $G = (G_1, G_2)$*, where* $G_1 = (V_1, I_1, P_1, S)$ *is a right linear grammar with* V_1*, a finite set of horizontal non-terminals,* I_1*, a finite set of intermediates (i.e.,* $I_1 = \{S_1, S_2, ..., S_k\}$*),* P_1 *is a finite set of right linear grammar production rules called* horizontal production rules*, and* S*, the start symbol where* $S \in V_1$ *and* $V_1 \cap I_1 = \phi$*. We define* $G_2 = (\bigcup_{i=1}^{k} G_{2i})$ *where* $G_{2i} = (V_{2i}, I_{2i}, P_{2i}, S_i)$ *is a right linear grammar,* V_{2i} *is a finite set of vertical non terminals,* I_{2i} *is a finite set of vertical terminals,* S_i *is the start symbol,* P_{2i} *is a finite set of vertical production rules,* $V_{2i} \cap V_{2j} = \phi$*, if* $i \neq j$*. The horizontal derivations and vertical derivations are denoted as* $\underset{h}{\Rightarrow}, \underset{v}{\Rightarrow}$ *respectively. The derivations are obtained by first applying horizontal production rules and then the vertical production rules. Firstly a horizontal string* $S_1S_2...S_k \in {I_1}^*$ *has been generated using horizontal production rules* P_1 *in* G_1*, i.e.,* $S \overset{*G_1}{\underset{h}{\Longrightarrow}} S_1S_2...S_n$*. A vertical derivation has been defined as follows : if there are rules* $S_i \downarrow a_{1i}A_i, A_i \downarrow a_{2i}B_{3i}, B_{ji} \downarrow a_{ji}B_{j+1}i, B_{ri} \downarrow a_{ri}, 3 \leq j \leq r-1$ *in* G_{2i}*, where* $i \in \{1, ..., k\}$ *for* $i = 1, ..., n$*, then the matrices will be generated in the following way :*

$$S \underset{h}{\overset{*}{\Rightarrow}} \begin{bmatrix} S_1 & \dots & S_n \end{bmatrix} \underset{v}{\Rightarrow} \begin{bmatrix} a_{11} & \dots & a_{1n} \\ A_1 & \dots & A_n \end{bmatrix} \underset{v}{\Rightarrow} \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \cdot & \dots & \cdot \\ a_{(r-1)1} & \dots & a_{(r-1)n} \\ B_{r_1} & & B_{r_n} \end{bmatrix}$$

$$\underset{v}{\Rightarrow} \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \cdot & \dots & \cdot \\ a_{(r-1)1} & \dots & a_{(r-1)n} \\ a_{r1} & & a_{rn} \end{bmatrix}$$

Here $\underset{v}{\overset{*}{\Rightarrow}}$ *is the transitive closure of* $\Rightarrow$*. The vertical derivation gets terminated if* $B_{ri} \rightarrow a_{ri}$ *are all terminal rules in* G_{2i} *where* $i = 1, ..., n$*.*

The set of all matrices is defined as follows:

$$L(G) = \{r \times n \; arrays \; [a_{ij}] \mid i = 1, ..., r, j = 1, ..., n, r, n \geq 1, S \overset{*G_1}{\underset{h}{\Longrightarrow}} S_1S_2...S_n \overset{*G_2}{\underset{v}{\Longrightarrow}} [a_{ij}]\}$$

Remark 1. A single non terminal is produced in each column as the rules are in the form of $A \rightarrow aB, A \rightarrow a$ where $a \in I_2$.

Remark 2. No cell in any column is blank or empty as a rule from one of G_{2i} where $i = 1, ..., k$ is supposed to be ϵ free.

Remark 3. In the definition of finite matrix grammar, the production rules are applied in a simultaneous fashion. In that sense, the grammars are matrix grammars. Moreover, the definition is more general in that the set of rules applied at one stage is not fixed but restricted by the horizontal string generated at the first stage. The name matrix grammar is retained to refer to this generalization also. Importantly, it should be noted that in this paper, the matrix grammars generate

matrix languages whose sentences are matrices ($m \times n$ rectangular arrays). On the other hand, in the formal language theory, the matrix languages are considered to be string languages where sentences are strings-generated by grammars written in the form of a matrix.

Remark 4 (Notation). If L' is the language generated by G_1, and $R_1, ..., R_k$ (the subsets of) the regular sets corresponding to G_{2i} where $i = 1, ..., k$, then we can write $L(G) = (L') : \ : (R_1, ..., R_k)$

2.1 FML - Examples

Example 1. Let $G = (G_1, G_2)$ where $G_1 = (\{S, S'\}, \{S_1, S_2\}, \{S \to S_1S', S' \to S_2S', S' \to S_2\}, S), G_2 = G_{21} \cup G_{22}, G_{21} = (\{S_1, A\}, \{X\}, \{S_1 \to XA, A \to XA, A \to X\}, S_1), G_{22} = (\{S_2, A\}, \{., X\}, \{S_2 \to .A, A \to .A, A \to X\}, S_2)$, then $L = \{S_1S_2^n \mid n \geq 1\}, R_1 = \{X^m \mid m \geq 1\}, R_2 = \{(.)^{m-1}X \mid m \geq 1\}$ and $L(G) = (L') : \ : (R_1, R_2)$. $L(G)$ is a finite matrix language and consists of $m \times n$ arrays $(m > 1, n > 1)$ describing the token L.

G generates $m \times n$ matrices $(m > 1, n > 1)$ which describe the token L. We illustrate by generating a 6×5 matrix from G.

$$S \underset{h}{\overset{*G_1}{\Longrightarrow}} \begin{bmatrix} S_1 & S_2 & S_2 & S_2 & S_2 \end{bmatrix} \underset{v}{\overset{G_2}{\Longrightarrow}} \begin{bmatrix} X & . & . & . & . \\ A & A & A & A & A \end{bmatrix} \underset{v}{\overset{G_2}{\Longrightarrow}} \begin{bmatrix} X & . & . & . & . \\ X & . & . & . & . \\ A & A & A & A & A \end{bmatrix}$$

$$\underset{v}{\overset{G_2}{\Longrightarrow}} \begin{bmatrix} X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ A & A & A & A & A \end{bmatrix} \underset{v}{\overset{G_2}{\Longrightarrow}} \begin{bmatrix} X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ A & A & A & A & A \end{bmatrix} \underset{v}{\overset{G_2}{\Longrightarrow}} \begin{bmatrix} X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ A & A & A & A & A \end{bmatrix}$$

$$\underset{v}{\overset{G_2}{\Longrightarrow}} \begin{bmatrix} X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ X & . & . & . & . \\ X & X & X & X & X \end{bmatrix} \in L(G)$$

Example 2. Let $G = (G_1, G_2)$ where $G_1 = (\{S, S'\}, \{S_1, S_2\}, \{S \to S'S, S \to S'S_1, S' \to S_1S_2S_2S_2\}, S), G_2 = G_{21} \cup G_{22}, G_{21} = (\{S_1, A\}, \{X\}, \{S_1 \to XA, A \to XA, A \to X\}, S_1), G_{22} = (\{S_2, S'_2\}, \{., X\}, \{S_2 \to S'_2S_2, S_2 \to S'_2X, S'_2 \to X..\}, S_2)$, then $L = \{S_1S_2S_2S_2^nS_1 \mid n \geq 1\}, R_1 = \{X^{m_1} \mid m_1 \geq 1\}, R_2 = \{(X..)^{m_2}X \mid m_2 \geq 1\}$ and $L(G) = (L') : \ : (R_1, R_2)$. $L(G)$ is regular and describes rectangular grids made up of $r \times s$ rectangles $(r = 1, 2, \ldots, s = 1, 2, \ldots)$ of the same size.

G generates the 2×4 grid $\in L(G)$.

$$S \underset{h}{\overset{*G_1}{\Longrightarrow}} \begin{bmatrix} S_1 & S_2 & S_2 & S_2 & S_1 & S_2 & S_2 & S_2 & S_1 & S_2 & S_2 & S_2 & S_1 & S_2 & S_2 & S_2 & S_1 \end{bmatrix}$$

$$\xRightarrow[h,v]{G_1,G_2} \begin{bmatrix} X & S'_2 & S'_2 & S'_2 & X & S'_2 & S'_2 & S'_2 & X & S'_2 & S'_2 & S'_2 & X & S'_2 & S'_2 & S'_2 & X \\ A & S_2 & S_2 & S_2 & A & S_2 & S_2 & S_2 & A & S_2 & S_2 & S_2 & A & S_2 & S_2 & S_2 & A \end{bmatrix}$$

$$\xRightarrow[v]{*G_1,*G_2} \begin{bmatrix} X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X \\ X & . & . & . & X & . & . & . & X & . & . & . & X & . & . & . & X \\ X & . & . & . & X & . & . & . & X & . & . & . & X & . & . & . & X \\ X & \dot{X} & \dot{X} & \dot{X} & X & \dot{X} & \dot{X} & \dot{X} & X & \dot{X} & \dot{X} & \dot{X} & X & \dot{X} & \dot{X} & \dot{X} & X \\ X & . & . & . & X & . & . & . & X & . & . & . & X & . & . & . & X \\ X & . & . & . & X & . & . & . & X & . & . & . & X & . & . & . & X \\ X & \dot{X} & \dot{X} & \dot{X} & X & \dot{X} & \dot{X} & \dot{X} & X & \dot{X} & \dot{X} & \dot{X} & X & \dot{X} & \dot{X} & \dot{X} & X \end{bmatrix} \in L(G)$$

In the next section we define *deterministic finite state matrix automata (DFSMA)*, to correspond to families of matrices (FML).

3 Deterministic Finite State Matrix Automata (DFSMA)

We define a deterministic version of finite state matrix automata.

Definition 2 (Deterministic finite state matrix automaton(DFSMA)). *A* deterministic finite state matrix automaton *is defined as a* 9 *tuple* $DFSMA = (Q, I, T, \delta, \delta', S, F', F, \$)$ *where*

- T *: Set of horizontal symbols and* $|T|$ *is the number of horizontal symbols and it denotes the number of horizontal states also.*
- I *: Set of vertical symbols.*
- $Q = (\bigcup_{i=1}^{k} Q_i) \cup Q'$ *is the finite set of states where* Q_i *is the finite set of vertical states corresponding to each horizontal state* $S_i \in Q'$*,* Q' *is the finite set of horizontal states where* $(\bigcup_{i=1}^{k} Q_i) \cap Q' = \phi$ *and* $|Q'| = |T| = k$*.*
- *Vertical transition function* $\delta : (Q_i \cup Q') \times I \rightarrow Q_i$*. A* vertical transition *of* $DFSMA$ *is of the form :* $\delta(q_i, x) = q_j$ *where* $q_i, q_j \in (Q_i \cup Q')$*, if* $q_i = S_j \in Q'$ *where* $i = 0$*, then it is the first transition of the automata which starts from the start state* $S_0 \in Q'$*, and if* $i \neq 0$ *then it is the first transition from any other horizontal state* $S_j \in Q'$*. The vertical transition function* δ *can be extended to* $\hat{\delta}$ *that operates on states and strings (as opposed to states and symbols), such that,* $\hat{\delta}(q_i, \epsilon) = q_i, \hat{\delta}(q_i, xa) = \delta(\hat{\delta}(q_i, x), a)$*.*

$$Q_i = \{q_k \mid (\hat{\delta}(q_j, xa) = q_k) \wedge (q_j \in Q_i \vee q_j = S_j)\}.$$

- *Horizontal transition function* $\delta' : F' \times \{\$\} \rightarrow Q'$*, then the horizontal transition is of the form* $\delta'(f_i, \$) = S_j$*.* $(F' \cap Q_i)$ *is a singleton set which contains only* f_i*,* $x_{*,i}$ *denotes the* i*th column vector.*

$$Q' = \{S_j \mid \delta'(f_i, \$) = S_j \wedge (f_i \in F') : \delta'(\hat{\delta}(S_i, x_{*,i}), \$) = S_j\}$$

- $S_0 \in Q'$ *: Initial state*
- *We define finite set of vertically accepting states* $F' = (\bigcup_{i=1}^{k} f_i)$*, such that,*

$$F' = \{f_i \mid \hat{\delta}(q_i, x_{*,i}) = f_i \wedge \delta'(f_i, \$) = S_j \wedge (1 \leq i \leq k)\}$$

- $F = f_k$ *denotes the final state of DFSMA, such that,* $\hat{\delta}(S_k, x_{*,k}) = f_k$ *where* $f_k \in Q_k$ *and* k *is the number of horizontal states.*

– $ *is the end marker where* $\$ \notin I$

The (standard) semantics of DFSMA is defined as follows.

A *vertical run of* $DFSMA$ *over* a vertical word $w_v = x_0 \cdots x_n$, is a sequence of steps of $DFSMA$:

$$S_i \xrightarrow{x_0} q_1 \quad \ldots \quad q_n \xrightarrow{x_n} q_{n+1}$$

We say a vertical run is *accepting* if $q_{n+1} = f_i \in F'$. It is *rejecting* if $q_{n+1} \notin F'$. Vertical word w_v is *accepted (rejected)* if $DFSMA$ has an accepting (rejecting) run over w_v. There must be an accepting vertical run corresponding each intermediate state $S_j \in Q'$ where $1 \leq j \leq k$.

A *horizontal run of* $DFSMA$ *over* a horizontal word $w_h = x_{*,0}x_{*,1} \cdots x_{*,n}$ where $x_{*,i}$ denotes the ith column of the matrix, a horizontal word w_h is a sequence of column vectors where each column vector is followed by the end marker $\$ \notin I$, such that, $w_h = x_{*,0}\$\ x_{*,1}\$ \cdots x_{*,n}$, is a sequence of steps of $DFSMA$:

$$S_0 \xrightarrow{x_{*,0}} f_0 \xrightarrow{\$} S_1 \quad \ldots \quad S_n \xrightarrow{x_{*,n}} f_n.$$

We say a horizontal run is *accepting* if $f_n = f_k \in F'$ where $|T| = k$. Horizontal word w_h is *accepted (rejected)* if $DFSMA$ has an accepting (rejecting) run over w_h. The *language* of $DFSMA$, notation $L(DFSMA)$, is the set of all horizontal words or images that are accepted by $DFSMA$.

Our proposed DFSMA has single initial state S_0. The automaton starts reading from the first column of the input matrix. All the vertical and horizontal moves are unique. It reaches f_i and then using enmarker $\$_i$ goes to another column corresponding to some S_j. If $i = j$ then it will create a loop, otherwise it will go to new horizontal state. If there exist an input matrix $m \times n$ then the automaton reads till the nth column, there will not be any endmarker $\$_n$ followed by the nth column, so the last horizontal move is based on the endmarker $\$_{n-1}$ which takes the automaton to S_n. The automaton will finish the reading with the nth set of vertical moves corresponding to S_n, and it ends up with $f_n = F$, it has single final state (Fig. 1).

3.1 DFSMA - Examples

Example 3. We define $DFSMA = (Q, I, T, \delta, \delta', S, F, F', \$)$ which can accept the language of Example 2.1 (L token).

– $Q = \{Q_1 \cup Q_2 \cup Q'\}$ where $Q_1 = \{q_{1_1}\}, Q_2 = \{q_{2_1}, q_{2_2}\}$ and $F' = \{q_{1_1}, q_{2_2}\}$ where $q_{1_1} = f_1, q_{2_2} = f_2$ and $Q' = \{S_1, S_2\}$,
– $I = \{., x\}$,
– $T = \{S_1, S_2\}$,
– $F' = \{f_1 = q_{1_1}, f_2 = q_{2_2}\}$, and $f_2 = q_{2_2}$ is the final state.
– S_1 is the initial state.
– $ is the end marker where $\$ \notin I$
– Vertical transitions (δ) and horizontal transitions (δ') are given below.

1. $\delta(S_1, x) = q_{1_1}$ where $q_{1_1} = f_1$
2. $\delta(q_{1_1}, x) = q_{1_1}$
3. $\delta'(q_{1_1}, \$) = S_2$
4. $\delta(S_2, .) = q_{2_1}$
5. $\delta(q_{2_1}, .) = q_{2_1}$
6. $\delta(q_{2_1}, x) = q_{2_2}$
7. $\delta'(q_{2_2}, \$) = S_2$

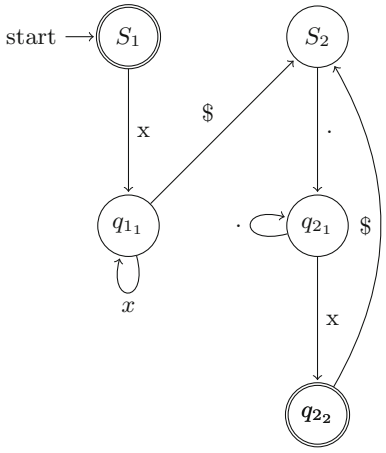

Fig. 1. Deterministic finite state matrix automaton.

Example 4. We define $DFSMA = (Q, I, T, \delta, \delta', S, F, F', \$)$ which can accept the language of Example 2.2 (L token).

- $Q = \{Q_1 \cup Q_2 \cup Q_3 \cup Q_4 \cup Q'\}$ where $Q_1 = \{q_{1_1}\}, Q_2 = \{q^1_{2_1}, q^1_{2_2}, q^1_{2_3}\}, , Q_2 = \{q^1_{2_1}, q^1_{2_2}, q^1_{2_3}\}, Q_3 = \{q^2_{2_1}, q^2_{2_2}, q^2_{2_3}\}, Q_4 = \{q^3_{2_1}, q^3_{2_2}, q^3_{2_3}\}$ and $F' = \{q_{1_1}, q^1_{2_3}, q^2_{2_3}, q^3_{2_3}\}$ where $q_{1_1} = f_1, q^1_{2_3} = f_2, q^2_{2_3} = f_3, q^3_{2_3} = f_4$ and $Q' = \{S_1, S^2_1, S^2_2, S^2_3\}$,
- $I = \{., x\}$,
- $T = \{S_1, S^2_1, S^2_2, S^2_3\}$,
- $F' = \{f_1 = q_{1_1}, f_2 = q^2_{2_3}, f_3 = q^2_{2_3}, f_4 = q^3_{2_3}\}$, and $f_1 = q_{1_1}$ is the final state.
- S_1 is the initial state.
- \$ is the end marker where $\$ \notin I$
- Vertical transitions (δ) and horizontal transitions (δ') are given below (Fig. 2).

1. $\delta(S_1, x) = q_{1_1}$ where $q_{1_1} = f_1$
2. $\delta(q_{1_1}, x) = q_{1_1}$
3. $\delta'(q_{1_1}, \$) = S^2_1$
4. $\delta(S^2_1, X) = q^1_{2_1}$

5. $\delta(q_{2_1}^1, .) = q_{2_2}^1$
6. $\delta(q_{2_2}^1, .) = S_1^2$
7. $\delta(q_{2_2}^1, X) = q_{2_3}^1$
8. $\delta'(q_{2_3}^1, \$) = S_2^2$
9. $\delta(S_2^2, X) = q_{2_1}^2$
10. $\delta(q_{2_1}^2, .) = q_{2_2}^2$
11. $\delta(q_{2_2}^2, .) = S_2^2$
12. $\delta(q_{2_2}^2, X) = q_{2_3}^2$
13. $\delta'(q_{2_3}^2, \$) = S_3^2$
14. $\delta(S_3^2, X) = q_{2_1}^3$
15. $\delta(q_{2_1}^3, .) = q_{2_2}^3$
16. $\delta(q_{2_2}^3, .) = S_3^2$
17. $\delta(q_{2_2}^3, X) = q_{2_3}^3$
18. $\delta'(q_{2_3}^3, \$) = S_1$

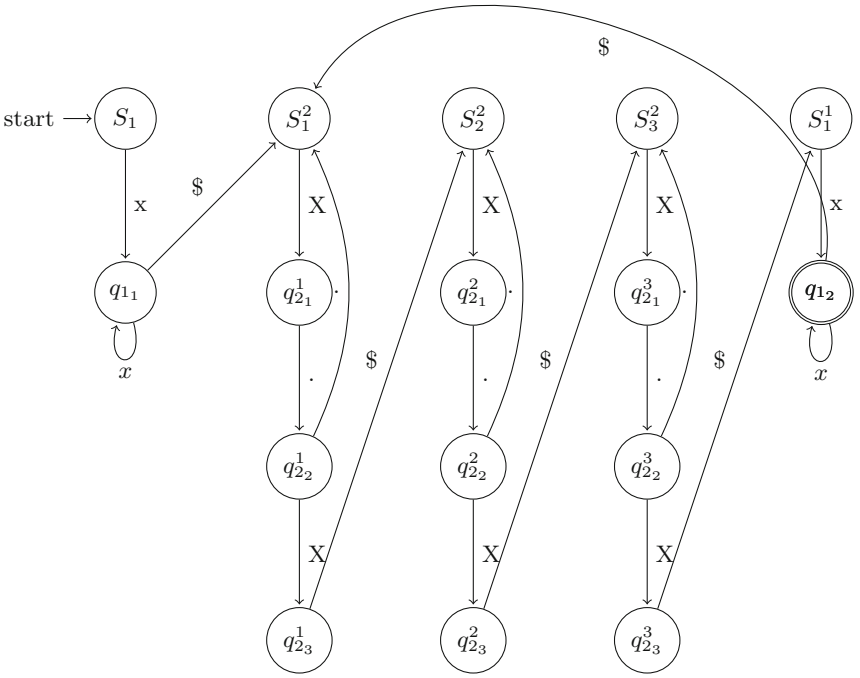

Fig. 2. Deterministic finite state matrix automaton for 2×4 grid

In the next section, we show some of the important results of FML.

4 Properties of Finite Matrix Languages

In this section, we summarize some of the important closure properties of FML in Table 1. Also, we present some of the decidable results of FML in Table 2. The following important results had been eastablished in [28].

Table 1. Closure property results

Union	Closed
Concatenation	Closed
Kleene Closure	Closed
Complementation	Closed
Intersection	Closed

Table 2. Decidability results

$L = \phi$	Decidable
$L = \Sigma^*$	Decidable
$L_1 = L_2$	Decidable
$w \in L$	Decidable

As the membership problem, $(w \in L)$, is decidable, it would be possible to apply MAT model to learn $DFSMA$. In order to apply MAT model, very importantly we must establish the important Myhill - Nerode theorem for $DFSMA$ and FML.

In the next section, we discuss the Myhill - Nerode equivalence of $DFSMA$ and FML.

5 Myhill - Nerode Equivalence

The Myhill-Nerode equivalence [2] considers two words w and w' of a language L equivalent if there does not exist a suffix u that distinguishes them, that is, only one of the words wu and $w'u$ is in L. The Myhill-Nerode theorem states that L is regular if and only if this equivalence relation has a finite index, and moreover that the number of states in the smallest deterministic finite automaton (DFA) recognizing L is equal to the number of equivalence classes. In this section, we present a Myhill-Nerode theorem for DFSMA and FML. In string languages, Myhill and Nerode only needs a single equivalence relation on words to capture DFAs, we need two relations $\equiv_v$, $\equiv_h$ on words to capture the richer structure of DFSMA.

Here, first we define *Right invariant vertical equivalence relation* and *Right invariant horizontal equivalence relation* in order to establish the Myhill - Nerode theorem for DFSMA and FML.

Definition 3 (Right invariant vertical equivalence relation). *A vertical equivalence relation $\equiv_v$ on I^* is said to be right invariant if, for $x, y, z \in I^*$, $x \equiv_v y \implies \forall z(xz \equiv_v yz)$.*

Example 5. Suppose $L = (L') : \quad : (R_1, ..., R_k)$ be a language over I^{**} where each $R_i, i \geq 1$ be a language over I. If there exist an equivalence relation $\equiv_R$ on

I^* then it is a right invariant equivalence relation on I^*. We define $x \equiv_R y$ if and only if $\forall z(xz \in R \iff yz \in R)$. It can be easily cross-checked that $x \equiv_R y$ is an equivalence relation as it satisfies reflexive, symmetric, and transitive properties. We assume that $x \equiv_R y$ where $x, y \in I^*$ and $z \in I^*$ be an arbitrary, now our claim is $xz \equiv_R yz$, that is, $(\forall w)(xzw \in R \iff yzw \in R)$. For any arbitrary $w \in I^*$, we write $u = zw$, now since $x \equiv_R y$, we have $xu \in R \iff yu \in R$, so $xzw \in R \iff yzw \in R$.

Definition 4 (Right invariant horizontal equivalence relation). *Horizontal equivalence relation $\equiv_h$ on I^{**} is said to be right invariant if, for $x, y, z \in I^{**}$, $x \equiv_h y \implies \forall z(xz \equiv_h yz)$.*

Example 6. Suppose $L = (L') : \ : (R_1, ..., R_k)$ be a language over I^{**}. If there exist an equivalence relation $\equiv_{L'}$ on I^{**} then it is a right invariant equivalence relation on I^{**}. We define $x \equiv_{L'} y$ if and only if $\forall z(xz \in L' \iff yz \in L')$ where $x, y, z \in I^{**}$. It can be easily verified that $x \equiv_{L'} y$ is an equivalence relation as it satisfies reflexive, symmetric, and transitive properties. We assume that $x \equiv_{L'} y$ where $x, y \in I^{**}$ and $z \in I^{**}$ be an arbitrary, Now we claim $xz \equiv_{L'} yz$, that is, $(\forall w)(xzw \in L' \iff yzw \in L')$. For any arbitrary $w \in I^{**}$, we write $u = zw$, now since $x \equiv_{L'} y$, we have $xu \in L' \iff yu \in L'$, so $xzw \in L' \iff yzw \in L'$.

Lemma 1. *Suppose $DFSMA = (Q, I, T, \delta, \delta', S, F, F', \$)$. There exist a vertical equivalence $\equiv_{DFSMA_v}$ and it is right invariant.*

Proof. We define $x \equiv_{DFSMA_v} y$ if and only if $\hat{\delta}(s_0, x) = \hat{\delta}(s_0, y)$ where $x, y \in I^*$. It is trivial that $x \equiv_{DFSMA_v} y$ is an equivalence relation as it satisfies reflexive, symmetric and transitive properties. We consider $x \equiv_{DFSMA_v} y$ that is $\hat{\delta}(s_0, x) = \hat{\delta}(s_0, y)$, for $z \in I^*$, $\hat{\delta}(s_0, xz) = \hat{\delta}(\hat{\delta}(s_0, x), z) = \hat{\delta}(\hat{\delta}(s_0, y), z) = \hat{\delta}(s_0, yz)$ as we know already that $\hat{\delta}(s_0, x) = \hat{\delta}(s_0, y)$. (See Definition 3)

Lemma 2. *Suppose $DFSMA = (Q, I, T, \delta, \delta', S, F, F', \$)$. There is a horizontal relation $\equiv_{DFSMA_h}$ on I^{**}, and it is right invariant.*

Proof. Suppose $w = x_{*,0} \cdots x_{*,n}$ and $w' = y_{*,0} \cdots y_{*,n'}$ then $w \equiv_{DFSMA_h} w'$ if and only if -

$$\delta'(\hat{\delta}(s_0, x_{*,0}), \$) = \delta'(\hat{\delta}(s_0, y_{*,0}), \$)$$
$$\delta'(\hat{\delta}(s_1, x_{*,1}), \$) = \delta'(\hat{\delta}(s_1, y_{*,1}), \$)$$
$$\vdots$$
$$\delta'(\hat{\delta}(s_n, x_{*,n}), \$) = \delta'(\hat{\delta}(s_{n'}, y_{*,n'}), \$)$$

Now it can be easily understood that $\equiv_h$ is right invariant equivalence relation if, for all $z \in I^{**}$, wz and $w'z$ leads $DFSMA$ to same state. (See Definition 4)

We can now state and prove the celebrated result of Myhill & Nerode.

Theorem 1. *Suppose there is a DFSMA. Then* $L(DFSMA) = (L') :: (R_1, ..., R_k)$.

Proof. Assume $(L') : : (R_1, ..., R_k)$ *is recognized by* $DFSMA = (Q, I, T, \delta, \delta', S, F, F', \$)$,

- *For* $(x \in R_i)$, *if* $\hat{\delta}(s_0, x) = p$, *then*

$$[x]_i = \{y \in R_i \mid \hat{\delta}(s_0, y) = p\}$$

(All those strings member of R_i, *if we put them in the initial state* S_0 *and if they reach* p, *they are equivalent to* x*).*
- *That is given,* $(q \in Q_i)$, *we define -*

$$C_q = \{x \mid (x \in R_i) \wedge (R_i \subseteq I^*) \wedge \hat{\delta}(S_0, x) = q\}$$

(All those strings if we put them in the initial state S_0, *if they reach* q, *then those strings are in equivalence class* C_q. C_q *is possibly empty if* q *is reachable. So corresponding to each state there is an equivalence class of* $\equiv_{R_i}$ *and it is finite index.)*
- *The vertical equivalence classes corresponding to each* R_i *are completely determined by the vertical states of DFSMA. More over the number of vertical equivalence classes of* $\equiv_{R_i}$ *for each* R_i *is less than or equal to the number of vertical states of DFSMA for each* R_i. *As we know that for each* i, $|Q_i|$ *is finite, we can conclude that the number of vertical equivalence classes of* $\equiv_{R_i}$ *for each* R_i *is finite index.*
-

$$\begin{aligned} R_i &= \{x \in I^* \mid \hat{\delta}(S_0, x) \in F'\} \\ &= \bigcup_{p_v \in F'} \{x \in I^* \mid \hat{\delta}(S_0, x) = p_v\} \\ &= \bigcup_{p_v \in F'} C_{p_v} \end{aligned}$$

(C_{p_v} *is a vertical equivalence class corresponding to state* p_v, R_i *is union of all* C_{p_v} *for* $p_v \in F'$. *Some of the intermediate final states may not be reachable, in that case the set is empty) (See Lemma 1)*
- *Similarly it can be shown that the horizontal equivalence classes are completely determined by the the horizontal states of DFSMA, a horizontal word* $w \in I^{**}$ *is consisting of multiple column vectors, such that,* $w = x_{*,0}\ x_{*,1} \cdots x_{*,j}$, *we define,* $s_0 \xrightarrow{x_{*,0}\$\ x_{*,1}\$\cdots x_{*,j-1}\$} s_j$ *using a sequence of steps of DFSMA:*

$$s_0 \xrightarrow{x_{*,0}} f_0 \xrightarrow{\$} s_1 \quad \ldots \quad s_{j-1} \xrightarrow{x_{*,j-1}} f_{j-1} \xrightarrow{\$} s_j,$$

We define $[s_j]$, *if* $s_0 \xrightarrow{x_{*,0}\$\ x_{*,1}\$\cdots x_{*,j-1}\$} s_j$, *then,*

$$[x_{*,0}\ x_{*,1} \cdots x_{*,j-1}] = \{y_{*,0}\ y_{*,1} \cdots y_{*,k} \in I^{**} \mid s_0 \xrightarrow{y_{*,0}\$\ y_{*,1}\$\cdots y_{*,k}\$} s_j\}$$

– *More over the number of horizontal equivalence classes of* $\equiv_{L'}$ *is less than or equal to the number of horizontal states of* $DFSMA$.

$$C_{s_j} = \{x_{*,0}\ x_{*,1} \cdots x_{*,n'} \in I^{**} \mid s_0 \xrightarrow{x_{*,0}\$\ x_{*,1}\$ \cdots x_{*,n'}\$} s_j\},$$

is an equivalence class of $\equiv_{L'}$ *and finite index.*

–

$$\begin{aligned} L' &= \{x \in I^{**} \mid \delta'(\hat{\delta}(s_0, w), \$) \in S_j\} \\ &= \bigcup_{p_h \in S_j} \{x \in I^{**} \mid \delta'(\hat{\delta}(s_0, w), \$) = p_h\} \\ &= \bigcup_{p_h \in S_j} C_{p_h} \end{aligned}$$

(C_{p_h} is a horizontal equivalence class corresponding to state p_h, L is union of all C_{p_h} for $p_h \in F$. Some of the final states may not be reachable, in that case the set is empty)(See Lemma 2)

Example 7. In the Example 1 of Subsect. 3.1, the vertical equivalence classes are following:

– $C_{q_{11}} = \{X^m \mid (\hat{\delta}(s_1, X^m) = q_{11}, m \geq 1\}$
– $C_{q_{21}} = \{(.)^m \mid (\hat{\delta}(s_2, (.)^m) = q_{21}, m \geq 1\}$
– $C_{q_{22}} = \{(.)^m X \mid (\hat{\delta}(s_2, (.)^m X) = q_{22}, m \geq 1\}$

Horizontal equivalence class is given below.

$$C_{s_1} = \{\epsilon \mid s_1 \xrightarrow{\epsilon} s_1\}$$

S_1 is the start state.

$$C_{s_2} = \{((.)^m X)_0 \cdots ((.)^m X)_n \mid s_2 \xrightarrow{((.)^m X)_0 \cdots ((.)^m X)_n} s_2\}$$

Theorem 2. *Suppose* $L = (L') : \ : (R_1, ..., R_k)$ *is an* FML *over* I^{**}. *Then there exist a* $DFSMA$ *such that* $L = L(DFSMA)$.

Proof. We define $DFSMA = (Q, I, T, \delta, \delta', S, F, F', \$)$ *where*

– $Q = Q'' \cup Q'$ *such that* $Q'' = \bigcup_{i=1}^{k} Q_i$ *where* $\forall i\ Q_i = \{[x]_i \mid (x \in R_i \wedge R_i \subseteq I^*)\}$ *is a finite set of vertical states corresponds to* S_i *and* $Q' = \{[x_{*,i}] \mid x_{*,i} \in I^{**}\}$ *where* $x_{*,i}$ *is the ith column vector and it is followed by the ith end marker* $\$_i$, *then it goes to another horizontal state* S_j. $\forall i\ Q_i$ *is the set of equivalence classes of* $\equiv_{R_i}$ *and* Q' *is the set of equivalence classes of* $\equiv_{L'}$.
(We consider that for each horizontal state $S_i, i \geq 0$, *there is a finite set* Q_i *of vertical states, all these vertical moves are same as DFA. In case of horizontal moves, the automaton needs to read atleast one column, if it reads the ith column vector* $x_{*,i}$, *then it encounters with the ith end marker* $\$_i$, *finally it goes to some another horizontal state* S_j, *then it is called horizontal move.)*

- $S_0 = [\epsilon]$
  (Since we assume that L is nonempty, $\epsilon \in Pref(L)$. Here S_0 is the first horizontal state, always the first horizontal state will be considered as the initial state of the automata. Here S_0 will have Q_0 which will contain a finite set of vertical states and using them the automata will read the corresponding 0th column vector $x_{*,0}$, followed by $\$_0$, then it goes to some S_j where $j \geq 0$.)
- $F = \{[w] \mid w \in L\}$ *where* $w = x_{*,0} \cdots x_{*,n}$.
  ($w = x_{*,0} \cdots x_{*,n}$ where $x_{*,0}$ represents the 0th column, $x_{*,1}$ represents 1th column so on. So 0th to nth column vector, that i, an array of n columns is getting accepted. Every column is followed by an end marker \$ apart from the last column $x_{*,n}$, in case of the last column, it reaches f_n which is the final state. There is no more column, so there does not exist more horizontal state.)
- $F' = \{[x_{*,i}] \in Q_i \mid [x_{*,i}] \in R_i \wedge (1 \leq i \leq k)\}$.
  (Each column vector $x_{*,i}$ takes the automata to vertical final state $f_i \in F'$. After reaching the vertical final state, the automaton encounters with end marker \$, then it goes to another horizontal state.)
- *Vertical transition :* $\delta([x], a) = [xa]$ *and there exist* Q_i, *such that,* $[x] \in Q_i$.
- *Horizontal transition :* $\delta'([x_{*,i}], \$) = [S_i S_j]$ *where* S_i, S_j *corresponds to* $x_{*,i}, x_{*,j}$ *respectively,* $[x_{*,i}] = f_i \in F', [S_i S_j] \in Q'$.

Example 8. Example 1 in Sect. 2.1 is suggested to refer. Now, we show that $L(DFSMA) = R_i$.

Corollary 1. $(R_i \subseteq L(DFSMA) \wedge L(DFSMA) \subseteq R_i) \implies (R_i = L(DFSMA))$

Proof. First we will show that $\forall i\ R_i \subseteq L(DFSMA)$, $w_v = x_0 \cdots x_n$, *if* $w_v = x_0 \cdots x_n$ *is an arbitrary element, then the vertical run of DFSMA:*

$$S_i \xrightarrow{x_0} q_1 \quad \ldots \quad q_n \xrightarrow{x_n} q_{n+1}$$

Here the run will be accepted if $q_{n+1} \in F'$, *so any* $w_v \in R_i$, *that will be accepted by DFSMA, that is,* $\forall i\ R_i \subseteq L(DFSMA)$.

Now, we need to show that $L(DFSMA) \subseteq R_i$, *using contrapositive we can write* $\neg R_i \nsubseteq \neg L(DFSMA)$, *then the run of DFSMA:*

$$S_i \xrightarrow{x_0} q_1 \quad \ldots \quad q_n \xrightarrow{x_n} q_{n+1}$$

Here the run of DFSMA will be rejected as $q_{n+1} \notin F'$.

Here, we show that $L(DFSMA) = L$.

Corollary 2. $(L \subseteq L(DFSMA) \wedge L(DFSMA) \subseteq L) \implies (L = L(DFSMA))$

Proof. First we will show that $L \subseteq L(DFSMA)$, *if* $w = x_{*,0}\$\ x_{*,1}\$ \cdots x_{*,n}$ *is an arbitrary element, then the run of DFSMA:*

$$S_0 \xrightarrow{x_{*,0}} f_0 \xrightarrow{\$} S_1 \quad \ldots \quad S_n \xrightarrow{x_{*,n}} f_n.$$

Here the run is accepting if $f_n \in F$, so any $w \in L$, that will be accepted by $DFSMA$, that is, $L \subseteq L(DFSMA)$.

Now, we need to show that $L(DFSMA) \subseteq L$, using contrapositive we can write $\neg L \nsubseteq \neg L(DFSMA)$, then the run of $DFSMA$:

$$S_0 \xrightarrow{x_{*,0}} f_0 \xrightarrow{\$} S_1 \quad \ldots \quad S_n \xrightarrow{x_{*,n}} f_n.$$

Here the run of $DFSMA$ is will be rejected as $f_n \notin F$.

6 Conclusion and Future Work

In this paper we define deterministic finite state matrix automata, $DFSMA$, which can recognize finite matrix languages, FML. $DFSMA$ has single initial state and single final state. More importantly, we established the Myhill-Nerode theorem for $DFSMA$ and FML. Unlike the classical Myhill-Nerode theorem, here we need two equivalence relations, called vertical equivalence relation $\equiv_v$ and horizontal equivalence relation $\equiv_h$ to capture the behaviour of $DFSMA$. Now as we have Myhill-Nerode theorem for $DFSMA$ and FML, we can come up with a learning algorithm for $DFSMA$.

So, in the form of future work, it could be the immediate step of this work to develop an efficient learning algorithm for $DFSMA$, it could be interesting to explore query learning model [2] in this context.

References

1. Amar, V., Putzolu, G.: On a family of linear grammars. Inf. Control **7**(3), 283–291 (1964)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
3. Angluin, D.: Negative results for equivalence queries. Mach. Learn. **5**(2), 121–150 (1990)
4. Angluin, D., Fisman, D.: Learning regular omega languages. Theoret. Comput. Sci. **650**, 57–72 (2016)
5. Angluin, D., Kharitonov, M.: When won't membership queries help? In: Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, pp. 444–454 (1991)
6. Anselmo, M., Giammarresi, D., Madonia, M.: A common framework to recognize two-dimensional languages. Fund. Inform. **171**(1–4), 1–17 (2020)
7. Bergadano, F., Varricchio, S.: Learning behaviors of automata from shortest counterexamples. In: Vitányi, P. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 380–391. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59119-2_193
8. Birkendorf, A., Böker, A., Simon, H.U.: Learning deterministic finite automata from smallest counterexamples. SIAM J. Discret. Math. **13**(4), 465–491 (2000)
9. Bunke, H., Sanfeliu, A.: Syntactic and Structural Pattern Recognition: Theory and Applications, vol. 7. World Scientific, Singapore (1990)

10. Cassel, S., Howar, F., Jonsson, B., Steffen, B.: Active learning for extended finite state machines. Formal Aspects Comput. **28**(2), 233–263 (2016). https://doi.org/10.1007/s00165-016-0355-5
11. Cavalcanti, A., Dams, D.: FM 2009: Formal Methods: Second World Congress, Eindhoven, The Netherlands, November 2–6, 2009, Proceedings, vol. 5850. Springer (2009). https://doi.org/10.1007/978-3-642-05089-3
12. Fernau, H., Paramasivan, M., Schmid, M.L., et al.: Simple picture processing based on finite automata and regular grammars. J. Comput. Syst. Sci. **95**, 232–258 (2018)
13. Firschein, O.: Syntactic pattern recognition and applications. Proc. IEEE **71**(10), 1231–1231 (1983)
14. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 215–267. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_4
15. Ginsburg, S., Spanier, E.H.: Control sets on grammars. Math. Syst. Theory **2**(2), 159–177 (1968)
16. Gold, E.M.: Language identification in the limit. Inf. Control **10**(5), 447–474 (1967)
17. Habarra, O., Jiang, T.: Learning regular languages from counterexamples. J. Comput. Syst. Sci. **43**(2), 299–316 (1991)
18. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. ACM SIGACT News **32**(1), 60–65 (2001)
19. Isberner, M., Howar, F., Steffen, B.: The TTT algorithm: a redundancy-free approach to active automata learning. In: Bonakdarpour, B., Smolka, S.A. (eds.) RV 2014. LNCS, vol. 8734, pp. 307–322. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11164-3_26
20. Koshiba, T., Mäkinen, E., Takada, Y.: Learning deterministic even linear languages from positive examples. Theoret. Comput. Sci. **185**(1), 63–79 (1997)
21. Mäkinen, E.: A note on the grammatical inference problem for even linear languages. Fund. Inform. **25**(2), 175–182 (1996)
22. Maler, O., Staiger, L.: On syntactic congruences for ω-languages. Theoret. Comput. Sci. **183**(1), 93–112 (1997)
23. Midya, A., Thomas, D., Malik, S., Pani, A.K.: Polynomial time learner for inferring subclasses of internal contextual grammars with local maximum selectors. In: Hung, D., Kapur, D. (eds.) Theoretical Aspects of Computing- ICTAC 2017. Lecture Notes in Computer Science, vol. 10580, pp. 174–191. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67729-3_11
24. Midya, A., Thomas, D.G., Pani, A.K., Malik, S., Bhatnagar, S.: Polynomial time algorithm for inferring subclasses of parallel internal column contextual array languages. In: Brimkov, V.E., Barneva, R.P. (eds.) IWCIA 2017. LNCS, vol. 10256, pp. 156–169. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59108-7_13
25. Midya, A., Vaandrager, F., Thomas, D.G., Ghosh, C.: Simulating parallel internal column contextual array grammars using two-dimensional parallel restarting automata with multiple windows. In: Lukić, T., Barneva, R.P., Brimkov, V.E., Čomić, L., Sladoje, N. (eds.) IWCIA 2020. LNCS, vol. 12148, pp. 106–122. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51002-2_8
26. Mráz, F., Průša, D., Wehar, M.: Two-dimensional pattern matching against basic picture languages. In: Hospodár, M., Jirásková, G. (eds.) CIAA 2019. LNCS, vol. 11601, pp. 209–221. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23679-3_17
27. Narasimhan, R.: Labeling schemata and syntactic descriptions of pictures. Inf. Control **7**(2), 151–179 (1964)

28. Radhakrishnan, V., Chakravarthy, V., Krithivasan, K.: Some properties of matrix grammars- parallel image analysis. In: International Work shop on Parallel Image Processing and Analysis-Theory and Applications, pp. 213–225 (1999)
29. Radhakrishnan, V., Nagaraja, G.: Inference of even linear grammars and its application to picture description languages. Pattern Recogn. **21**(1), 55–62 (1988)
30. Rivest, R.L., Schapire, R.E.: Inference of finite automata using homing sequences. Inf. Comput. **103**(2), 299–347 (1993)
31. Rosenfeld, A.: Picture Languages: Formal Models for Picture Recognition. Academic Press, Cambridge (2014)
32. Rosenfeld, A., Siromoney, R.: Picture languages-a survey. Lang. Des. **1**(3), 229–245 (1993)
33. Salomaa, A.: On grammars with restricted use of productions. Suomalainen Tiedeakatemia (1969)
34. Sempere, J.M., García, P.: A characterization of even linear languages and its application to the learning problem. In: Carrasco, R.C., Oncina, J. (eds.) ICGI 1994. LNCS, vol. 862, pp. 38–44. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58473-0_135
35. Stromoney, G., Siromoney, R., Krithivasan, K.: Abstract families of matrices and picture languages. Comput. Graph. Image Process. **1**(3), 284–307 (1972)
36. Takada, Y.: Grammatical inference for even linear languages based on control sets. Inf. Process. Lett. **28**(4), 193–199 (1988)
37. Takada, Y.: Algorithmic learning theory of formal languages and its applications. Ph.D. thesis, International Institute for Advanced Study of Social Information Science ... (1992)
38. Takada, Y.: A hierarchy of language families learnable by regular language learning. Inf. Comput. **123**(1), 138–145 (1995)
39. Takada, Y.: Learning formal languages based on control sets. In: Jantke, K.P., Lange, S. (eds.) Algorithmic Learning for Knowledge-Based Systems. LNCS, vol. 961, pp. 316–339. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60217-8_15
40. Yokomori, T.: Learning non-deterministic finite automata from queries and counterexamples. In: Machine Intelligence, vol. 13, pp. 169–189 (1992)